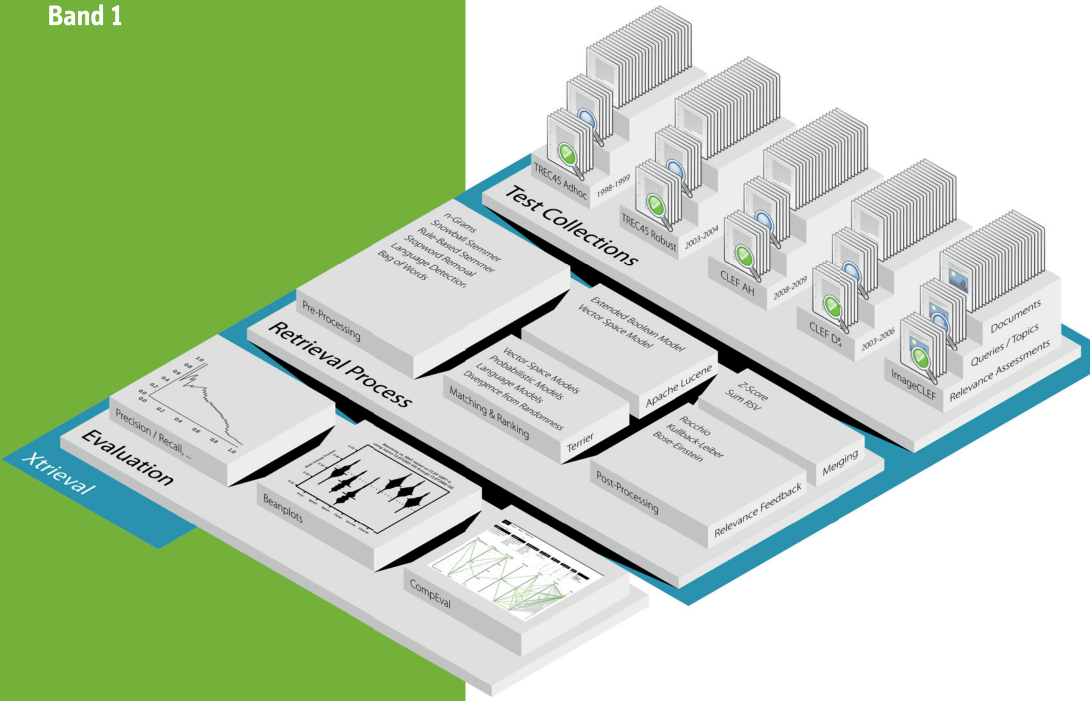
Jens Kürsten

A Generic Approach to Component-Level Evaluation in Information Retrieval

Wissenschaftliche Schriftenreihe
Dissertationen der Medieninformatik
Band 1



TECHNISCHE UNIVERSITÄT
CHEMNITZ

PROFESSUR
MEDIENINFORMATIK

Fakultät für Informatik · Professur Medieninformatik · www.tu-chemnitz.de/cs/mi

Jens Kürsten

A Generic Approach to Component-Level Evaluation in Information Retrieval

Wissenschaftliche Schriftenreihe

Dissertationen der Medieninformatik

Band 1

Prof. Dr. Maximilian Eibl (Hrsg.)

Jens Kürsten

A Generic Approach to Component-Level Evaluation in Information Retrieval

TECHNISCHE UNIVERSITÄT
CHEMNITZ

Universitätsverlag Chemnitz
2012

Impressum

Bibliographische Information der Deutschen Nationalbibliothek

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Angaben sind im Internet über http://dnb.d-nb.de abrufbar.

Zugl.: Chemnitz, Techn. Univ., Diss., 2012

Cover-Gestaltung: Arne Berger
Cover-Illustration: Thomas Wilhelm

Technische Universität Chemnitz/Universitätsbibliothek
Universitätsverlag Chemnitz
09107 Chemnitz
http://www.bibliothek.tu-chemnitz.de/UniVerlag/

Herstellung und Auslieferung
Verlagshaus Monsenstein und Vannerdat OHG
Am Hawerkamp 31
48155 Münster
http://www.mv-verlag.de

ISSN 2195-2574 ISBN 978-3-941003-68-2

http://nbn-resolving.de/urn:nbn:de:bsz:ch1-qucosa-96344

TECHNISCHE UNIVERSITÄT CHEMNITZ

A Generic Approach to Component-Level Evaluation in Information Retrieval

von der Fakultät für Informatik
der Technischen Universität Chemnitz
genehmigte Dissertation
zur Erlangung des akademischen Grades

Doktoringenieur (Dr.-Ing.)

vorgelegt von
Jens Kürsten
geboren am 29. Juli 1980 in Torgau

Tag der Einreichung:	1. März 2012
Tag der Verteidigung:	6. September 2012
Gutachter:	Prof. Dr. Maximilian Eibl, Technische Universität Chemnitz
	Prof. Vivien Petras, PhD, Humboldt-Universität zu Berlin
	Prof. Dr. Harald Kosch, Universität Passau

Danksagung

Ich möchte zuerst die Gelegenheit nutzen, alljenen zu danken, die mich auf dem Weg des Verfassens dieser Dissertation maßgeblich unterstützt haben. In erster Linie gilt dies natürlich für Prof. Maximilian Eibl, der mit seiner konstruktiven und hilfreichen Kritik, sowie mit gezielten Denkanstößen wesentlich dazu beigetragen hat, dass ich meine Forschungstätigkeit in dieser Arbeit darstellen kann.

Bei Prof. Vivien Petras bedanke ich mich für die wertvollen Diskussionen während der Entstehung dieser Arbeit. Ihre wichtigen Kommentare und Hinweise haben die Arbeit wesentlich bereichert.

Meinen Kollegen in der Forschungsinitiative *sachsMedia*: Arne Berger, Katharina Einert, Stephan Heinich, Robert Knauf, Albrecht Kurze, Markus Rickert und Marc Ritter, danke ich für ihre wertvollen Anregungen und die sehr angenehme Zusammenarbeit über mehrere Jahre. Mein besonderer Dank gilt Thomas Wilhelm für seine Unterstützung bei allen technischen Problemen im Zusammenhang mit dieser Arbeit. Ohne seine nützlichen Hinweise wäre ich an mancher Stelle verzweifelt.

Auch die Zusammenarbeit mit den Mitarbeitern der Universitäten Passau und Hildesheim hat wesentlich zum Gelingen dieser Arbeit beigetragen. Für ihre Anregungen zu dieser Arbeit danke ich Prof. Harald Kosch, Prof. Thomas Mandl und Prof. Christa Womser-Hacker, sowie Daniela Becks, Britta Meixner und Peter Schultes.

Zudem danke ich Sue Kirk für ihre hilfreichen Anmerkungen zur Englischen Orthografie und Grammatik, die es mir erlaubt haben diese Arbeit sprachlich weiter zu verbessern.

Meinen Eltern, Ines und Thomas, danke ich herzlichst für ihre Unterstützung, die mir geholfen hat mich selbst und meine Interessen zu finden. Ein Ziel wie diese Arbeit wäre ohne ihre kontinuierliche Förderung für mich unerreichbar gewesen.

Ganz besonders möchte ich mich bei meiner Freundin Petra bedanken. Es ist nahezu unmöglich all die Tätigkeiten aufzuzählen, die sie beigesteuert hat, um mir das Verfassen dieser Arbeit zu erleichtern. Ihr steter Zuspruch hat es mir ermöglicht, das Ziel meiner Tätigkeit im Auge zu behalten. Für die Einschränkungen, die sie auf sich genommen hat, um mich auf dem Weg zu dieser Arbeit zu unterstützen, bin ich ihr zu tiefstem Dank verpflichtet.

Kurzfassung

Das Forschungsgebiet Information Retrieval befasst sich mit Theorien und Modellen, die die Grundlage für jegliche Dienste bilden, die als Antwort auf ein formuliertes Informationsbedürfnis den Zugang zu oder einen Verweis auf entsprechende Elemente einer Dokumentsammlung ermöglichen. Die Qualität von Suchalgorithmen wird im Teilgebiet Information Retrieval Evaluation untersucht. Der klassische Ansatz für den empirischen Vergleich von Retrievalsystemen basiert auf dem Cranfield-Paradigma und nutzt einen spezifischen Korpus mit einer Menge von Beispielanfragen mit zugehörigen Relevanzbewertungen.

Internationale Evaluationskampagnen, wie die Text Retrieval Conference, haben in den vergangenen zwei Jahrzehnten zu großen Fortschritten in der Methodik der empirischen Bewertung von Suchverfahren geführt. Der generelle Fokus dieses systembasierten Ansatzes liegt jedoch nach wie vor auf dem Vergleich der Gesamtsysteme, dass heißt die Systeme werden als Black Box betrachtet. In jüngster Zeit ist diese Evaluationsmethode vor allem aufgrund des Black-Box-Charakters des Untersuchungsgegenstandes in die Kritik geraten. Aktuelle Arbeiten fordern einen differenzierteren Blick in die einzelnen Systemeigenschaften, bzw. ihrer Komponenten. In der vorliegenden Arbeit wird ein generischer Ansatz zur komponentenbasierten Evaluation von Retrievalsystemen vorgestellt und empirisch untersucht.

Der Fokus der vorliegenden Dissertation liegt deshalb auf zentralen Komponenten, die für die Bearbeitung klassischer Ad-Hoc Suchprobleme, wie dem Finden von Büchern zu einem bestimmten Thema in einer Menge von Bibliothekseinträgen, wichtig sind. Ein zentraler Ansatz der Arbeit ist die Weiterentwicklung des Xtrieval Frameworks mittels der Integration weitverbreiteter Retrievalsysteme mit dem Ziel der gegenseitigen Eliminierung systemspezifischer Schwächen. Herausragende Ergebnisse im internationalen Vergleich, für verschiedenste Suchprobleme, verdeutlichen sowohl das Potenzial des Ansatzes als auch die Flexibilität des Xtrieval Frameworks.

Moderne Retrievalsysteme beinhalten zahlreiche Komponenten, die für die Lösung spezifischer Teilaufgaben im gesamten Retrievalprozess wichtig sind. Die hier vorgelegte Arbeit ermöglicht die genaue Betrachtung der einzelnen Komponenten des Ad-hoc Retrievals. Hierfür wird mit Xtrieval ein Framework dargestellt, welches ein breites Spektrum an Verfahren flexibel miteinander kombinieren lässt. Das System ist offen konzipiert und ermöglicht die Integration weiterer Verfahren sowie die Bearbeitung weiterer Retrievalaufgaben jenseits des Ad-hoc Retrieval. Damit wird die bislang in der Forschung verschiedentlich geforderte aber bislang nicht erfolgreich umgesetzte komponentenbasierte Evaluation von Retrievalverfahren ermöglicht.

Mächtigkeit und Bedeutung dieser Evaluationsmöglichkeiten werden anhand ausgewählter Instanzen der Komponenten in einer empirischen Analyse mit über 13.000 Systemkonfigurationen gezeigt. Die Ergebnisse auf den vier untersuchten Ad-Hoc Testkollektionen sind vielfältig. So wurden beispielsweise systematische Fehler bestimmter Ranking-Modelle identifiziert und die theoretischen Zusammenhänge zwischen spezifischen Klassen dieser Modelle anhand empirischer Ergebnisse nachgewiesen. Der Maßstab des durchgeführten Experiments macht eine Analyse aller möglichen Einflüsse und Zusammenhänge zwischen den untersuchten Komponenten unmöglich. Daher werden die erzeugten empirischen Daten für weitere Studien öffentlich bereitgestellt.

Abstract

Research in information retrieval deals with the theories and models that constitute the foundations for any kind of service that provides access or pointers to particular elements of a collection of documents in response to a submitted information need. The specific field of information retrieval evaluation is concerned with the critical assessment of the quality of search systems. Empirical evaluation based on the Cranfield paradigm using a specific collection of test queries in combination with relevance assessments in a laboratory environment is the classic approach to compare the impact of retrieval systems and their underlying models on retrieval effectiveness.

In the past two decades international campaigns, like the Text Retrieval Conference, have led to huge advances in the design of experimental information retrieval evaluations. But in general the focus of this system-driven paradigm remained on the comparison of system results, i.e. retrieval systems are treated as black boxes. This approach to the evaluation of retrieval system has been criticised for treating systems as black boxes. Recent works on this subject have proposed the study of the system configurations and their individual components. This thesis proposes a generic approach to the evaluation of retrieval systems at the component-level.

The focus of the thesis at hand is on the key components that are needed to address typical ad-hoc search tasks, like finding books on a particular topic in a large set of library records. A central approach in this work is the further development of the Xtrieval framework by the integration of widely-used IR toolkits in order to eliminate

the limitations of individual tools. Strong empirical results at international campaigns that provided various types of evaluation tasks confirm both the validity of this approach and the flexibility of the Xtrieval framework.

Modern information retrieval systems contain various components that are important for solving particular subtasks of the retrieval process. This thesis illustrates the detailed analysis of important system components needed to address ad-hoc retrieval tasks. Here, the design and implementation of the Xtrieval framework offers a variety of approaches for flexible system configurations. Xtrieval has been designed as an open system and allows the integration of further components and tools as well as addressing search tasks other than ad-hoc retrieval. This approach ensures that it is possible to conduct automated component-level evaluation of retrieval approaches.

Both the scale and impact of these possibilities for the evaluation of retrieval systems are demonstrated by the design of an empirical experiment that covers more than 13,000 individual system configurations. This experimental set-up is tested on four test collections for ad-hoc search. The results of this experiment are manifold. For instance, particular implementations of ranking models fail systematically on all tested collections. The exploratory analysis of the ranking models empirically confirms the relationships between different implementations of models that share theoretical foundations. The obtained results also suggest that the impact on retrieval effectiveness of most instances of IR system components depends on the test collections that are being used for evaluation. Due to the scale of the designed component-level evaluation experiment, not all possible interactions of the system component under examination could be analysed in this work. For this reason the resulting data set will be made publicly available to the entire research community.

Contents

List of Figures

List of Tables

1 Introduction

Searching for a desired piece of information is an everyday activity in modern life. Due to the increasing amount of digital information which is instantly available by accessing resources like the world wide web or other, possibly private, networks, finding a source of information is also a complex task. The field of *information retrieval* research deals with theories and models that constitute the foundations for any kind of service that provides access or pointers to particular elements of a collection of documents in response to a submitted information need. The term *information retrieval* (IR) was coined by Calvin Mooers in a short paper [134, p. 572] in 1950:

> “The problem of directing a user to stored information, some of which may be unknown to him, is the problem of ’information retrieval’.”

The development of ideas to model the process of information retrieval was motivated by the possibility of constructing mechanised systems for literature search. These systems were the precursors of modern information retrieval systems of which the omnipresent web search engines are the most prominent example nowadays. The specific field of information retrieval evaluation is concerned with the critical assessment of the quality of search systems in general. This thesis focuses on the scientific evaluation of modern information retrieval systems and the underlying system components.

In order to place this work within the field of information retrieval evaluation research, the most essential achievements in the discipline need to be pointed out. Cyril Cleverdon's Cranfield experiments [39] were the first large-scale, scientific investigation of the effectiveness of search and it is considered a major milestone for evaluation in information retrieval. Together with his team he studied how four different library indexing schemes affected the quality of search results over a period of several years in the early 1960's. Because of the enhancements of mainframe and personal computers, the amount of information processed and stored on these machines was growing rapidly during the following three decades. At the same time the size of the collections used for the assessment of new retrieval models remained almost unchanged. As a consequence there existed a large gap between the collections used in research to develop new ideas, and those being used in commercial services at the start of the 1990's.

The formation of the Text Retrieval Conference (TREC) in 1992 was the next milestone for information retrieval research. It served two major goals. First, the assembly of full-text document collections that have sizes comparable to collections used in commercial search engines. Second, the organisers of TREC formulated the evaluation tasks, the test queries, and the relevance assessments, in order to ensure an independent comparison of the different implementations of information retrieval theories and models. The general set-up of a TREC evaluation experiment consists of a search task based on a collection of documents. A set of test queries and corresponding relevance assessments is used to rank participating retrieval systems according to the retrieval effectiveness they achieved on that test set. Although the scientific discipline has shifted its focus of interest towards diversity in search tasks, the general experimental set-up for evaluation remained almost unchanged for scenarios that do not include real system users.

The purpose of system-driven evaluation in the laboratory setting is to investigate and understand the effects of the systems on retrieval effectiveness. A specific problem of this approach is that the systems under examination are usually assembled by different groups that focus on a single, or only a few aspects, of their own system. The general ranking of all contributed systems and their configurations does not account for this.

Instead, it provides a general overview on all applied experimental settings by treating all systems as black boxes. This thesis addresses this limitation by considering the key components of IR systems that are needed to deal with a specific search task. In order to understand how the IR system components contribute to the usefulness of the complete system, each of the components has to be examined individually. Since it is not clear whether an implementation of a component that outperforms other algorithms still produces optimal results when it is plugged into the complete system, the components have to be studied individually, but also in the context of the search task the system is used for. This thesis proposes a generic approach to component-level evaluation of retrieval systems and substantiates this concept based on a large-scale empirical study on small to mid-sized text collections from TREC and other evaluation campaigns.

From here on the acronym *IR*, or the term *retrieval*, will be used to refer to information retrieval. Since IR effectiveness evaluation is the central topic of interest, the terms *search quality*, *retrieval performance*, and *retrieval effectiveness*, are being used as interchangeable references to the subject throughout this thesis.

1.1 Key Challenges

In the past decades IR evaluation research has met a number of challenges, most of which have been related to the scale of the data sets used for empirical experiments. With the turn of the millennium the diversity of search applications and tasks, which can mainly be attributed to the development of the Internet and its social web applications, added another dimension to the problem. Scalability and complexity are the key aspects that distinguish modern IR systems from their antecedents. In experimental IR research it is difficult to account for both simultaneously. However, evaluation campaigns like TREC will have to deal with these issues. Therefore, the current laboratory evaluation paradigm needs a major adjustment. The problem had already been pointed out in the following statement by Stephen Robertson from 2009 [145, p. 4]:

> “The experimental paradigm is that we have a number of alternative systems, and the research question under investigation is: ’Which system is best’. If we take seriously the notion that we are engaged in developing a science of search, then Cranfield would seem to [...] help in the development of models or theories. [...] However, an analysis of the role of empirical knowledge in general and laboratory experiment in particular, in relation to models or theories, reveals some limitations of the Cranfield approach. Despite the huge advances in this experimental paradigm [...], I believe we are only scratching the surface of what experiments can tell us. [...] I believe that what we need now is not so much better systems (though they are always welcome) as better understanding of the phenomena.”

Component-level IR evaluation is one approach in providing a better understanding of the relationships between IR models implemented in various IR systems. Its main purpose is to study the dimensions of the system and task complexity. But the diversity of search tasks is a major problem for this method. If it is not known which components of the IR systems are needed for a particular search task, then it is impossible to define the scope of the experiment. As a result component-level evaluation requires to know which of the system components and configurations are potentially useful for a particular search task. Most of the current approaches to comparisons at component-level are based on fixed architectures. This drawback will be discussed in detail in this thesis. Corresponding solutions to the problem will be discussed hypothetically.

It has already been pointed out that component-level experimentation has to focus on the comparison of individual components in the context of a search task under examination. This is a challenging problem for the typical approach to IR evaluation, because the actual software implementations of the components are only known to the individual participants of the task. As a consequence, recording and comparing the effects of individual components is difficult and can only be realised based on a meta-level discussion of the individual results. And yet not all combinations of IR system components may have been covered due to the lack of interchangeable components or component outputs. The thesis on hand postulates the hypothesis that automated IR evaluation at the component-level can be realised by employing the integrative approach of the Xtrieval framework [108].

The use of publicly available baselines for the critical assessment of new methods is an important issue in IR experimentation in general and component-level evaluation in particular. This includes baseline implementations of IR systems and components as well as reference baselines of results of effectiveness evaluations. A recent study [9] on research publications reporting improved retrieval effectiveness demonstrated that the existing baselines are used only rarely in retrospective evaluation experiments. As a result, although improved retrieval performance is reported frequently, a consistent upward trend could not be found on the most widely-used test collections. In line with the present thesis, widely-used IR software toolkits are discussed in detail and integrated into a common framework named Xtrieval. The purpose of this strategy is to combine the strengths and to eliminate the limitations of individual implementations. This hypothesis is investigated by means of a large-scale experiment that covers many different state-of-the-art components of IR systems.

1.2 Contributions

This thesis is devoted to experimental IR evaluation on component-level. It develops concepts which go beyond the traditional paradigm based on the Cranfield methodology and the corresponding enhancements developed during 20 years of IR evaluation at TREC. A review of current approaches to experimental evaluation of IR system components forms the basis of the approach proposed in this work.

In order to integrate the most significant implementations of IR models and theories a critical discussion of widely-used IR toolkits is provided. The Xtrieval framework is used to conduct a large-scale experiment to study the effect of state-of-the-art IR system components on retrieval effectiveness using a selection of standard ad-hoc test collections. Different implementations of key system components are selected for inclusion in the experiment based on a detailed discussion of the underlying theories and models. The experiment demonstrates that automated component-level evaluation can be realised when existing IR toolkits are combined so that the advantages of corresponding implementations can optimally complement each other. The results also show that the best experiments from the corresponding evaluation campaigns can be improved. This suggests that the performance of the baselines that should be used for reference on these widely-used test collections is in fact higher than the best results obtained from the original evaluations.

The large experiment set created for the validation of the developed approach to component-level IR evaluation is a further contribution of this thesis. The key observations concerning the impact on retrieval effectiveness that are highlighted in this work are only a first step to understanding the relationships between the different IR system components. Further studies can focus on a detailed investigation of potential connections between particular component implementations and specific types of test queries. Others may concentrate on the statistical analysis of the interactions between individual system configurations based on the observation that, for instance, some ranking models favour specific implementations and configurations of the pseudo-relevance feedback component. The developed approach is generally applicable and can be adapted to other search tasks, given that the relevant implementations of system components are publicly available.

The following list provides an overview of the key contributions of this thesis:

- The enhancement of the Xtrieval framework laid the foundations for the empirical evaluation presented in this thesis. The initial version of Xtrieval was based on Lucene as core retrieval library. In the course of the creation of this work, the Xtrieval framework was extended to allow the access to the Terrier toolkit and the Lemur project, which are two academic IR libraries that are commonly used in the IR community. The reasons for the selection of these toolkits and the potential benefits of the integration are outlined and discussed in detail.

- The empirical verification of the expected benefits from the combination of different IR toolkits and libraries is a central element of this thesis. The first prototype of the Xtrieval framework was implemented in 2007. Since then, it was used for many types of IR evaluation experiments. These include ad-hoc IR experiments on domain-specific and general library records, image retrieval on photographs, question answering on speech transcripts, video subject classification, and prior art search in patent document collections. This variety of retrieval tasks demonstrates the flexibility of Xtrieval. The strong retrieval performance that was achieved on these different evaluation tasks proves the quality of the strategy to combine state-of-the-art IR toolkits into a common framework.

- The Xtrieval framework provides solutions for the conceptual problems of automated evaluation at component-level. Its meta-level design for accessing and combining different state-of-the-art retrieval toolkits allows fine-grained empirical studies at the component-level. By incorporating the findings of previous evaluation initiatives, Xtrieval is deployed to run a large series of grid experiments that provide a better understanding of the orchestration of components in modern IR systems. Xtrieval allows to focus both on overcoming practical limitations and addressing methodological issues without gathering research groups or implementing additional data exchange protocols. The valuable test collections produced by evaluation campaigns like TREC, and open IR toolkits like Lemur, Lucene, and Terrier, representing the state-of-the-art in IR theory and practice, are the additional resources that are required to understand IR phenomena better.

- A large-scale empirical experiment is designed and discussed to investigate the effect of three key IR system components and their corresponding state-of-the-art implementations. The general experimental set-up is evaluated on a selection of ad-hoc test collections. In contrast to typical empirical IR experiments which study the effect of one particular system component implementation with respect to some baseline model, this thesis covers a total amount of 13,176 unique IR system configurations. Each of these configurations is tested on four different test collections which contain 100 test queries each. As a result, this empirical study generated about 5 million observations.

- To ensure the usefulness of the designed component-level experiment, it is evaluated against different types of document collections. For each of these collections several standard test sets, consisting of topics and corresponding relevance assessments, were aggregated into test collections of identical size. The analyses of the results on these test sets will help to understand, whether, or not, it is possible to predict the behaviour of system configurations on standard test collections. The proposed test set-up also allows the determination of appropriate baselines for particular instances of system components which can be used for testing and verifying future components on the deployed test collections.

- The exploratory data analysis of the experimental results allows the comparison of the effect of different component implementations on retrieval effectiveness. Each of the three components is studied separately by examining the probability distribution for selected instances on the experiment set. This method allows the identification of systematic failures of particular system components. It will be demonstrated that such failures exist and that it is likely that they can be corrected with a few steps.

- The large amount of experimental results is a problem that is addressed by providing a tool to visually compare subsets of the complete experiment set. The visualisation is based on parallel coordinates that allow the interactive comparison of individual component implementations and their impact on retrieval effectiveness. Since the analyses supplied in this thesis only investigated particular aspects of the generated data, the experimental results will be provided for in-depth analyses of further aspects that were out of the scope of this work.

1.3 Organisation

The content of this document is structured in three major parts excluding the Chapters 1 and 8. The first part is covered in Chapters 2 and 3 and deals with the foundations of modern IR systems, their components, and the scientific approaches to the evaluation of both. Chapters 4 and 5 constitute the second part of this work which is concerned with software frameworks that provide implementations of the most successful theories and models in IR. In the final part, covered by Chapters 6 and 7, the challenges of component-level evaluation are addressed by employing the introduced generic IR framework Xtrieval. The results of a large-scale retrieval evaluation experiment demonstrate that the automated evaluation of different components of IR systems enhances the understanding of their impact on retrieval effectiveness. The detailed outline of the individual chapters of this thesis is as follows:

- *Chapter 1* defines the term evaluation in the context of IR, its history, as well as its current and future applications. The relation between existing research in IR evaluation and the present work is established in order to introduce open questions in the field of experimental IR research. Key challenges in the field are presented in order to define the aims and the scope of this thesis.

- *Chapter 2* reviews the history of traditional approaches to IR evaluation. Starting with a detailed description of the key elements of empirical evaluation, test collections and metrics for the assessment of the quality of search results, recent issues of the empirical methodology are discussed. The concept of independent IR evaluation is introduced as an important step beyond the Cranfield evaluation paradigm. A comparison of recent evaluation experiments demonstrates key challenges for the assessment of retrieval systems at the component-level.

- *Chapter 3* introduces the key components of modern IR systems. Starting from an abstract point of view, the three fundamental processes: text pre-processing, core retrieval models, and feedback methods, are discussed in detail. The presented selection of theories and models summarises the state-of-the-art for each of the essential system components. These models form the basis for the empirical comparison of respective software implementations on ad-hoc test collections.

- *Chapter 4* discusses open-source software frameworks which contain different implementations of the system components described in Chapter 3. The emphasis is placed on widely-used toolkits and libraries in order to ensure the best possible coverage of component instances. A critical analysis of selected retrieval frameworks demonstrates their advantages and limitations.

- *Chapter 5* explains the motivation for the development of the extensible retrieval and evaluation framework Xtrieval which represents one of the central contributions of this thesis. First, the general architecture of the framework is described with the focus on compensating individual limitations of the frameworks discussed in Chapter 4 with the corresponding advantages of other retrieval toolkits. This concept is illustrated in detail by the description of the technical integration of two major retrieval frameworks. The strength of this approach of integrating the state-of-the-art in retrieval systems into a common framework is verified by the presentation of selected results on international IR evaluation tasks. At the same time the diversity of these search tasks demonstrates that Xtrieval is a multifunctional framework.

- *Chapter 6* summarises the experimental set-up and results of two recent evaluation tasks that focused on providing new insights into the impact of particular components of IR systems on retrieval effectiveness. Based on the conclusions from these evaluation efforts, five fundamental challenges that affect the usefulness of component-level evaluation are identified. It is discussed how these challenges can be addressed with Xtrieval.

- *Chapter 7* provides the empirical experiments for the validation of the presented approach to automated component-level evaluation. The presented experimental set-up relies on Xtrieval and generates a large set of retrieval system configurations. This set is studied on four test collections which were aggregated from different ad-hoc tasks of the two evaluation campaigns: TREC and CLEF. An exploratory data analysis based on beanplots allows the study of the interactions between the selected system component configurations. The results are reported in terms of optimal retrieval effectiveness. They demonstrate that the optimal system configuration is dependent on both the test topics and the document collections. It is also shown that some particular system components favour specific configurations of remaining components in order to achieve good retrieval effectiveness, while others are more robust against changes to other elements of a system configuration. The optimal results on the generated set of experiments demonstrate that the selected components of IR systems can be configured to improve retrieval effectiveness over the best experiments submitted to corresponding evaluation tasks.

- *Chapter 8* recaps the major findings and contributions of this thesis. It also provides an outlook for future developments of the Xtrieval framework and denotes the potential impact of the created component-level experiment set on IR evaluation in general.

2 Information Retrieval Evaluation

Empirical evaluation of technical implementations of theoretical models is a generic approach to study how theories affect particular use cases. In the field of IR research, a central question is how to provide the best answers to an information need of a user. The field of empirical evaluation of IR systems is primarily concerned with finding evidence which systems or system configurations are optimal for specific search tasks. Effectiveness and efficiency are two impartial aspects that contribute to the overall utility or success of an IR system.

Efficiency can be measured in terms of time and space requirements for the system. It can be easily quantified once a certain metric is defined, e.g. index size, indexing time, query throughput, or query latency [45]. Query throughput describes the number of queries processed per time unit (usually seconds) and query latency is the response time (in milliseconds) a user has to wait before receiving an answer. Search engine applications are typically designed to stay below a certain threshold of query latency, say 150 milliseconds, for conventional queries issued. However, optimising a system to reduce latency leads to worse throughput and vice versa: a conflictive duality between desirable properties that can be also found in effectiveness evaluation.

Effectiveness determines how well the ranked output of a system corresponds to an optimal ranking of documents, given a specific request and a definition of relevance. Generally speaking, it describes the quality of the search result presented to the user. Assessing the quality of new methods has always been a particular interest for IR researchers. However, evaluation of natural language search is challenging, because it is not deterministic like querying a database system. Typically, when users formulate their information need, it is underspecified or incomplete. Potential answers in a targeted collection are mostly unstructured, heterogeneous, and fuzzy on the level of their basic elements. Individual words can have multiple meanings, at the same time different words can express a single concept. These challenges contribute to the complexity of IR systems and the empirical assessment of their quality. This complexity is one of the reasons for the cyclic character of IR research, consisting of theoretical development, practical implementation, and experimental evaluation.

One of the central concepts in the evaluation of IR system effectiveness is relevance, which was introduced to obtain a measure for the utility of search results. There is a major difference between topical and user relevance. The latter type of relevance describes a subjective perspective and is hard to assess and compare. In contrast to that, if a document is assessed as being topically relevant to a query, it has to be concerned with the topic of the query. This broad and objective definition of relevance can be incorporated into the underlying model of an IR system. Topical relevance is the standard concept for system-based IR evaluation. Consequently, a set of user requests is usually called topic set. Another important aspect is to which level of granularity relevance judgements are collected. The degree of relevance is typically defined by the effectiveness measure that is used for evaluation. Binary relevance is the most traditional form, meaning a document can be either relevant or not relevant to a corresponding query. Multivalued relevance is usually applied in user-based evaluation, to account for diversity and the interactive nature of search. However, in traditional system-oriented evaluation three levels of relevance are sometimes used, where a document can be definitely relevant, possibly relevant, or definitely not relevant. Ternary relevance can be converted to traditional relevance by assigning possibly relevant documents to one of the two other classes.

Almost every experimental retrieval evaluation follows a prior definition of a retrieval task. Such a task is usually designed as an abstraction from a real-world search problem a user might be confronted with. All components of the evaluation, as well as their main properties, such as size, type, and structure depend on the objectives of the task. Additionally, the evaluation methodology itself can be motivated from two distinguishable perspectives: system-driven or user-driven [183]. The user-centric approach focuses on the interaction between a user and the system and is therefore the most realistic form of evaluation. It can be applied for systems under development as well as for operational systems. Typical metrics are: (1) the time a user spends on solving a specific task, (2) the amount of relevant items found, or (3) the subjective level of satisfaction with the system. A major criticism of the approach is that user satisfaction does not correlate well with their ability to find relevant items, or even worse, with the efficiency with which a search task was solved. User-based experiments are the most expensive form of empirical IR evaluation.

In a typical system-driven evaluation experiment, a collection of documents and a collection of user requests are carefully selected. Systems under evaluation run the set of requests against the document collection and return a ranked list of documents as response to each query. Human assessors examine ranked documents and provide a decision on their relevance to the corresponding request. The performance of the systems is then summarised by utilising an effectiveness metric that compares the ranked results to the relevance assessments. The system-centric approach is designed as a controlled laboratory environment with the key objective to provide data sets that can be reused for subsequent evaluation. It is criticised for its lack of realism, mainly because of its artificial formulation of information needs and the static nature of relevance.

The cognitive IR evaluation approach [22] aims to address realism and experimental control, the fundamental issues in IR effectiveness evaluation. The key element of this method is to incorporate a simulated work task into the evaluation. As a result, the experimental set-up is less static than in traditional laboratory experiments. It allows experimental control through the definition of the task and its underlying context, but leaves the user freedom to complete the task and therefore retains the realistic character of the formulation of an information need. The model is based on the concept of an anomalous state of knowledge, which is a user's recognition of an insufficient knowledge given an external motivation that leads to an information need [17]. In contrast to system-driven evaluation an information need and the perception of relevance are dynamic and depend on (1) the work task, (2) the user, (3) the context, (4) and the situation. The interactive IR evaluation model manages to balance the trade-off between experimental control and realistic search situations. Unfortunately, the model inherits a major drawback from the user-centred evaluation: the cost.

The main focus of this work lies in the effectiveness evaluation of IR in the laboratory setting. It is based on using test collections and methodologies that were specifically designed for the system-driven approach to IR evaluation. This method allows rapid testing of new ideas and models in a controlled environment which is needed in order to assess the effect of a new technological development. Thus, the system-driven IR evaluation will remain one of the most important tools for the advancement of IR. Throughout this work we use the term evaluation to refer to the system-driven approach for assessing the effectiveness of IR systems. In the following sections we will review key concepts that constitute the foundations for the contributions of our research.

2.1 Evaluation Campaigns

Empirical evaluation of IR systems is a task that is both complex and expensive. In this section, we will briefly discuss the set-up and impact of the most popular IR evaluation campaigns. A comprehensive review on the efforts and relationships of these initiatives is given in [124]. A key milestone for IR evaluation was the formation of the Text Retrieval Conference (TREC) in the early 1990's. It served two major goals. First, the assembly of full-text document collections that have sizes comparable to collections used in commercial search engines. Second, the organisers of TREC formulated the evaluation tasks, the test queries, and the relevance assessments, in order to ensure an independent comparison of the different implementations of information retrieval theories and models.

Now in 2012, TREC celebrates its 21th anniversary. Nevertheless, it is still the largest and most popular IR evaluation campaign. TREC has a considerable influence on implementing empirical evaluation as one of the most important aspects in IR research. Considerable improvements in IR systems in the early 1990's can be attributed to the independent TREC evaluation methodology. In 2007, an overview on past activities at TREC listed 27 tracks [185], which investigated different aspects of the retrieval process, numerous types of document collections, and various search tasks. Due to new fields of research and evolving commercial search applications in the web, the total number of tracks rose to 34 over the past four years. TREC test collections are particularly relevant for this work. Since ad-hoc search is the exemplary evaluation scenario here, TREC disks 4 and 5 were obtained. The data collection was used for the Ad-Hoc tracks at TREC-6 through TREC-8 and for the Robust track at TREC 2004. The Congressional Records sub collection was excluded in evaluations that used TREC disks 4 and 5 from TREC-7 onward. Section 7.1.3 describes the components of the test collections used for the present work in more detail.

Over time, TREC inspired several spin-offs with specific goals like cross-language evaluation in European languages (CLEF) and evaluation of search in Asian languages (NTCIR). At the time of writing, both NTCIR and CLEF have existed for more than a decade. In 2000, the Cross-Language Evaluation Forum (CLEF) emerged from the Cross-Language track at TREC. The variety of languages and their different rules for obtaining morphemes for indexing was a central motivation to specialise and expand that particular topic of IR. As a consequence, CLEF followed the TREC model and developed document collections for more than 13 European languages in total [124]. In order to stimulate research on bi-lingual search applications, CLEF developed topics in several languages for each track. Thus, even more languages could be investigated. Relevance assessments were created by native speakers of the document languages. Over time, CLEF has offered different cross-language related search tracks. Most of them were also studied at TREC, but the corresponding tracks at CLEF focused on the cross-lingual aspect. Examples are the Question Answering track, the GeoCLEF track, or the ImageCLEF track to name but a few. The latter attracted a large research community and hence became an independent satellite workshop of CLEF from 2010. Since 2006, the Chemnitz retrieval group participated in different tracks at CLEF. Most of them were closely related to ad-hoc search scenarios. The lessons that have been learned from these experiments with varying test collections, led to the formulation of the central questions addressed in this work. Given the observation that search effectiveness is highly variant, the question is to what extent system configurations affect the quality of the results for different types of data collections.

NTCIR is an evaluation initiative that is very similar to CLEF. The first campaign was held in 1997 and in contrast to TREC and CLEF, NTCIR runs in cycles of 18 months. The focus of the NTCIR evaluation campaign is on cross-language retrieval tasks that use the major East-Asian languages, Chinese, Korean, and Japanese. Similar to its counterparts in the western hemisphere, the first tracks used news article document collections. Later, the tracks diversified to other topics, like Named Entity Recognition, Question Answering, and Word Segmentation. The latter is a problem specific to Asian sign languages. It is one of the reasons why cross-lingual retrieval on Asian document collections is more complex than similar tasks with documents in European languages.

Newswire document collections were widely used in the early years of the major evaluation campaigns. The reason was that they were easy to obtain and collections could be assembled from different sources. Another aspect was that no experts were needed for creating relevance assessments. But soon this concentration on a single type of mostly homogeneous documents drew criticism. In response to those comments, new document collections were created to design realistic and more specific evaluation tasks. This led to considerable diversification of document types, including web documents, patents, or multimedia annotations. Hence, more complex document structures could be exploited for IR evaluation. INEX (an acronym for initiative for the evaluation of XML retrieval) is a campaign that is dedicated to the retrieval of structured documents. Since 2002, it runs in an annual cycle. One of the central goals of INEX was bridging the database and the information retrieval communities by designing tasks, which demanded extracting only the smallest elements of relevant documents.

Another aspect of the diversification of evaluation task is the ever-growing amount of multimedia contents and descriptions. TREC and CLEF also organised tracks that investigated problems regarding multimedia retrieval. In 2001, a video track at TREC dealt with segmentation and low-level feature extraction. Since 2003, the video track is organised as an independent evaluation campaign named TRECVid. Recent tasks at TRECVid include semantic indexing, content-based copy detection, surveillance event detection, multimedia event detection. The variety of tasks shows that TRECVid aims to cover many real-world problems in different domains, which deal with (semi-) automatic processing of video content.

Since 2003, the ImageCLEF track explored techniques to combine visual and textual features of images in cross-language search scenarios. In general, images can be thought to be language independent, but in fact image annotations may appear in various languages. ImageCLEF also offered tasks using different types of collections, like tourist photographs, for ad-hoc search or medical images for automatic labelling. Another aspect of multimedia content that should not go without notice is audio. CLEF organised a track named Cross-Language Spoken Retrieval (CL-SR) from 2003 until 2007. The intention was to retrieve documents based on partially inaccurate transcripts that were generated automatically. The Music Information Retrieval Evaluation Exchange (MIREX) assessed approaches for content-based processing of music. The organisers offered tasks like querying a music database by humming, melody extraction, or a music similarity challenge. The variety of available retrieval evaluation campaigns shows that many real-world search problems are already being addressed. The recent growth of evaluation initiatives can be perceived as a reaction to the ever-growing amounts of digital information. At the same time it demonstrates that efficient and effective tools are needed to be able to access a desired piece of information.

2.2 Test Collections

Three main components compose a test collection in system-based IR evaluation: (1) a document collection in which every document has a unique identifier, (2) a set of topics with a unique identifier for each query, and (3) a query relevance set, (abbreviated: qrels), that consists of pairs of document and query identifiers, defining the relevance of documents to topics [158]. The fundamental set-up traces back to the early 1960's when Cyril Cleverdon and his team compared different indexing schemes for library records in the Cranfield I & II experiments [39, 40].

The starting point of the Cranfield experiments was Cleverdon's idea that an independent evaluation was needed to be able to fairly compare systems of different groups. The Cranfield I document collection consisted of 18,000 articles that were manually indexed using four different library classification schemes. 1,200 search questions were formulated with the aim of retrieving a single document from the collection. Based on these experiments Cleverdon and his colleagues decided to run a follow-up study that became known as Cranfield II. However, there were some differences between the two subsequent experiments. The Cranfield II collection consisted of about 1,400 hand-crafted document records from 200 articles recently published on the subject of aerodynamics. To obtain a set of topics the authors of the papers were asked to summarise the main idea that inspired the paper. In addition, the experimenters requested a multivalued rating for each reference the authors used. Based on the responses all documents were checked against all the formulated topics. Subsequently, the authors were contacted again to judge the documents that were found to be relevant. This resulted in an exhaustive test collection consisting of approximately 1,400 documents, 225 topics, and a set of complete multivalued relevance judgements [158].

Although this empirical set-up was adopted as standard for controlled laboratory experiments, it featured some properties that are worth pointing out from a current perspective. All of the following elements were created and controlled manually: the topic sets, the relevance judgements, and the documents that represented the collection. Relevance was perceived and modelled as topical relevance, although it was obtained from expert users. And, most importantly, all documents in the collection were judged with respect to each topic. The topics represented information needs as they would be formulated by a library user and issued to a librarian, whose task was to locate documents containing the desired information. As a result, topics were created as natural language requests as long as an ordinary sentence to imitate the service of a librarian.

2.2.1 Document Collections: From Cranfield to TREC and Beyond

A major problem from the 1960's to the early 1980's was the assembling and the distribution of the document collections for a meaningful comparison of retrieval models. Consequently, IR researchers were obliged either to re-use available collections or to build a new collection on their own. Since the latter was extremely time-consuming, it led to relatively small collections. However, various collections consisting of different types of documents were created during this period. Frequently used collections from this time are presented along with basic statistics in Table 2.1. It provides a basic view on characteristic features in order to point out the key issues of these early test collections. Almost all of these collections were assembled by IR research groups to test and assess their models, in contrast to the motivation of the Cranfield experiments which was to conduct independent evaluation [39]. Another important aspect is collection size.

As can be seen from Table 2.1 the number of documents and topics were in some kind of a stalemate. In the 1960's commercial IR services had several tens of thousands of records in their database. Only a few years later, in the early 1970's, retrieving items from hundreds of thousands of documents was considered routine in commer-

Name	Year	# Docs.	# Topics	Terms/Doc.	Terms/Topic	Rel./Topic
Cranfield II [74, 166]	1962	1,400	225	28.7	8.0	7.2
ADI [74]	1968	82	35	27.1	14.6	9.5
NPL [74, 166]	1970	11,429	93	20.0	7.2	22.4
MEDLARS [74]	1973	1,033	30	51.6	10.1	23.2
TIME [74]	1973	425	83	570	16.0	3.9
UKCIS [66, 172]	1973	27,361	182	6.7	7.4	58.9
INSPEC [166, 172]	1981	12,684	77	36.0	17.9	33.0
LISA [166, 172]	1982	6,004	35	39.7	16.5	10.8
CACM [74, 166]	1983	3,204	64	24.5	10.8	15.3
CISI [74, 172]	1983	1,460	112	46.5	28.3	49.8

Table 2.1: Statistics for commonly used collections from 1962-1983.

cial search applications [18]. Table 2.1 illustrates that the test collections used for research were considerably smaller. Test collections were designed for different purposes of evaluation and they consisted of document surrogates, i.e. titles, abstracts, or other bibliographic information [73]. The TIME collection, built and used by Salton's research group, was the only exception to this. It was assembled by creating manual transcripts of articles from Time magazine. The largest collection in terms of number of documents was UKCIS, but the documents were assembled by the extraction of titles from a database containing articles on the subject of chemistry.

Assessing the relevance for each topic in the creation of larger test collections was another major obstacle during this period. The larger amounts of documents rendered the exhaustive approach used for the Cranfield experiments unfeasible (see Section 2.2.3). Retrospectively, the most influential work was the formulation of the need for an ideal test collection [171]. The report addressed the major problems of test collections which existed at the time, but it had almost no impact on the evaluation methodology of that time [158]. An important contribution was the introduction of the concept of pooling documents to be judged for relevance in larger test collections. Spärck-Jones and van Rijsbergen [171] also proposed the creation of larger test collections that would be maintained and distributed by a common organisation. However, it took almost 20 years until the idea was put into reality.

Name	Year	# Docs.	Terms/Doc.	Short Description
CJACS [117]	1992	96,900	2786	Chemical Journals of the American Chemical Society as full text
CA [117]	1992	9,528,000	129	Titles, abstracts, keywords and phrases from citations on chemistry
TIPSTER [73]	1992	741,856	444.4	Mostly news articles, used in TREC-1 to 3, collected 1987-1992
TREC [187]	1994	556,077	541.9	Mostly news articles, used in TREC-6 to 8, collected 1989-1994
OHSUMED [82]	1994	348,566	250	Subset of references from MEDLINE, collected 1987-1991
AQUAINT [182]	2000	1,033,00	363	News articles, used from TREC-11, collected 1996-2000
AQUAINT-2 [47]	2006	907,000	-	News articles, used from TREC-16, collected 2004-2006

Table 2.2: Summary of commonly used collections from 1992-2004.

It has been pointed out that the National Institute of Standards and Technology (NIST) was funded to build a large test collection in the early 1990's. The objective was to use the collection for a research project named TIPSTER. Shortly thereafter, NIST decided to launch a program called Text Retrieval Conference (TREC). Making the TIPSTER collections available enabled the research community to work with document collections of a more reasonable size (see Table 2.2). A comparison of the commonly used INSPEC collection and two collections on chemistry, namely CJACS and CA, which were available in a commercial IR system, demonstrated the large difference between the size of the collections used in practice and in science [117]. As a result, building larger test collections became an urgent problem for the IR evaluation research community.

Already before the first TREC conference was held in 1992, it was obvious that the efforts to build large collections could not be managed by single research groups. Reviewing IR literature of the early 1990's shows that outside TREC, there was almost no intention to build new types of text collections. One exception was the publication on the OHSUMED collection by Hersh et. al. [82]. In Table 2.2 a selection of text collections from the TREC era (which is still continuing at the time of preparing this work) from research and commercial search applications is presented. This illustrates

Name	Year	# Docs.	Size	Short Description
VLC [78]	1997	7.5 M	20 GB	used at TREC-6; included TREC CDs 1-5, newspaper and government data as well as USENET news; a 10% sample was offered
VLC2 [77, 80]	1998	18.5 M	100 GB	used from TREC-7 to 9, web data indexed by the Internet Archive; two sub samples were also offered: BASE1 and BASE10
WT10g [13]	2000	1.7 M	10 GB	used at TREC-9,10; composed as a subset of VLC2 by server selection and by removing duplicates as well as binary content
.GOV [44]	2002	1.25 M	18 GB	used from TREC-11 to 13 for the web tracks a partial crawl of the .gov web domain
.GOV2 [38]	2004	25.2 M	426 GB	used from TREC-13 to 17 for the terabyte and million query tracks; a larger crawl of the .gov web domain
Blogs06 [136], [121]	2006	0.1 M	25 GB	used from TREC-15 to 17 for the blog tracks; crawl of blog data covering top, spam, and general interest blogs
Blogs08 [122]	2008	1.3 M	453 GB	used at TREC-18,19 for the blog track; a markedly larger crawl of blogs collected over a longer period of time
ClueWeb09[1]	2009	1.0 B	5 TB	used at TREC-19 for several web tracks; web pages crawled in 2009 covering ten languages; 50 M pages sub sample available

Table 2.3: Overview on web collections created from 1997-2009.

that the assembly of a document collection of appropriate size became unmanageable. TIPSTER, TREC and OHSUMED are considerably smaller than CA. Apart from the commercial collections, CA and OHSUMED, all documents were full text articles.

One major benefit of a central authority like TREC is its ability to direct the focus of the research community. Its success of motivated the formation of several other evaluation campaigns that were described in Section 2.1. All of these forums have become main authorities for IR research, producing new tasks and test collections in an (bi-)annual cycle, (for a more detailed description see Section 2.1). This diversification of tasks for IR evaluation reflects both the growth of available information and the need to access information anywhere and at any time. Hence, IR theory and research has become ubiquitous in practice. One turning point for this development was the creation of the worldwide web.

---

[1] `http://lemurproject.org/clueweb09`, retrieved on March 1, 2012

The web started in late 1993 and grew rapidly, especially around the turn of the millennium. It soon became a rich source of information that asked for fast and reliable search. Although the indexed size of the web[2] was already between 26 and 140 million pages [79] in 1998, it was not used as a source for IR research until then. A key reason was the unclear legal aspect of the web, i.e. it was not clear whether collecting and distributing web pages was a copyright violation or not [158]. Before the creation of the first web collection, IR research had advanced from using mainly bibliographical surrogates in the 1960's to the early 1990's when using full text collections of various sources became the de-facto standard for evaluation.

Table 2.3 lists the most commonly used collections that abstract the problem of searching the Web. Again, sizes are considerably smaller than collections used in commercial applications, except for the earliest collections VLC and VLC2 as well as the latest collection ClueWeb09. In the period covered in Table 2.3 the size of the web grew from hundreds of millions to tens of billions of web pages. It can be seen that collection sizes actually decreased from 1998 until 2004. In fact, WT10g was a sub sample of the earlier VLC2 collection that was cleaned-up to be able to focus on textual information from web pages. The second crawl from the .gov web domain, which is known as the .GOV2 collection was an attempt to create a test collection of one terabyte in storage size.

2.2.2 Topic Sets

Apart from assembling large document collections, creating a set of queries is another crucial step in IR evaluation. Again, the basic methodology was developed by the TREC organisers using the influential research outcomes from the previous decades. For the first eight TREC exercises 50 topics were created each year. The topics were designed to reflect the information needs of real users. Hence, they were created by

---

[2] `http://googleblog.blogspot.com/2008/07/we-knew-web-was-big.html`, retrieved on March 1, 2012

assessors who also created the relevance judgements, and each of the topics represented a genuine information need. The diversity of the topic set was also considered. Each topic was tested by running an initial search on a sample of the document set. Only topics that matched approximately 25 to 100 documents from the sample set were used for the final topic set [74]. This procedure generated a range of narrower and broader topics.

Another important contribution of the early TREC exercises was the definition of a topic structure to allow easy automatic processing. It became the norm for almost every text retrieval task and consists of four elements: (1) a topic identifier, (2) a short title, which reflects the type of requests users submit to web search engines today, (3) a one-sentence description of the information need, and (4) a narrative part that aims to provide a complete description of document relevance for assessors [74]. Figure 2.1 contains an example to illustrate the structure. The intention of the detailed formulation was to ensure the topic was understood completely. However, it enabled the research community to evaluate specialised tasks by restricting the parts of topics to be used for automatic query construction.

The number of queries used to compare a set of systems is one of the central problems in IR evaluation. A typical topic set used in TREC-like IR system evaluation contains 50 topics. The underlying assumption is that the sample of topics should reflect real user information needs. In order to achieve a high coverage of all kinds of queries, a reasonable amount of different types of topics is needed. The exact amount of topics required for meaningful comparison of IR systems also depends on the underlying search task and the data collection. For homogeneous collections like library catalogues or news wire, 50 topics are considered to be an adequate amount.

In heterogeneous web search scenarios the number of topics needed to achieve high coverage of possible formulated information needs might be considerably higher. This particular problem was addressed by the Million Query track at TREC from 2007

```
<top>

<num> Number: 192

<title> Topic: Oil Spill Cleanup

<desc> Description: Document will identify a method, procedure, or
chemical used in cleaning up oil spills such as the Exxon Valdez.
Only oil spills on water (ocean, bay, lake, etc.) will be used.

<narr>  Narrative:
To be relevant a document will identify a method, procedure, or
chemical process used in cleaning up the water and beaches after a
major oil spill, such as the Exxon Valdez incident. The mere mention
of cleanup efforts without identifying the method or chemical used
is not relevant. A document that refers to a procedure such as
scrubbing, spraying, etc. is relevant. References to the cost of
cleanup and number of people and equipment involved without
mentioning the method are not relevant.

</top>
```

Figure 2.1: Example topic from TREC-3 ad hoc task.

to 2009 [2]. 10,000 queries (in 2007 and 2008) and respectively 40,000 (in 2009) were made available to participating institutions. Different amounts of topics from the total pool were evaluated each year: 1,755 (2007), 874 (2008) and 684 (2009). Using complex but efficient methodologies for evaluation, the organisers were able to show that human assessor effort could be spread on up to 20 times more queries than in traditional ad-hoc evaluation tasks [37].

2.2.3 Relevance Judgements

Creating new test collections for evaluation experiments involves a considerable amount of manual work to obtain relevance judgements. For small collections comparable to Cranfield I and II, assessing most, or even all, of the documents for relevance seems feasible. For large-scale collections with several millions of documents, exhaustive evaluation for relevance, even using just a few topics, is impossible.

To reduce the manual effort, a method called pooling was introduced [171]. Rather than judging each document with respect to every topic, all top k documents returned by each competing system are merged into a common pool and all duplicates are removed. The depth k of the pool corresponds to the common cut-off level of the contributing result lists. Only documents in the pool are assessed for relevance to a corresponding query by presenting them in some random order to the human relevance assessors. All documents that are not part of the pool are considered irrelevant based on the coherent assumption that documents not listed within the top k ranks of any of the contributing systems are unlikely to be relevant. But the application of this pooling strategy results in incomplete relevance judgements.

2.2.3.1 Incompleteness

Incompleteness is a potential problem regarding the re-usability of test collections. Therefore, an ideal test collection should include enough relevance judgements to allow comparison with systems that did not contribute to the original pool. If a new system retrieves relevant documents within the top k documents that are not part of the initial pool, incompleteness results in a bias that penalises new systems. The main reason for this effect is the assumption that documents excluded from the pool are not relevant. Some experimental studies showed that pooling does omit relevant documents [72, 74]. Although different amounts of new relevant documents can be found per topic in a second round of assessing relevance, comparative evaluation of systems remains reliable [199].

Another approach when handling documents without relevance information, is to exclude them from the ranked system output and use the resulting condensed lists for evaluation [154]. However, excluding a document that was rejected because it had been considered to be irrelevant by many systems previously, and promoting a document that is more likely to be relevant (because it was in fact returned by other systems) instead, tends to result in evaluation bias in favour of new systems [155].

Other effectiveness metrics are specifically designed to handle incomplete relevance information, namely BPref [30] and RBP [133] (see Section 2.3). A recent study [189] on TREC ad-hoc experiment data proposed to account for the pooling bias against unpooled systems by bias estimation based on existing systems. It also reported that assessing pooled and unpooled systems on a small common set of topics, and using the observed bias to adjust the bias on existing topics, reduced error rates in ranking systems. Another form of bias is introduced when top k pooling is applied to larger collections (see Table 2.3), and the pools tend to be too small in relation to the total document set size. A recent study [27] showed that relevance assessments obtained by traditional pooling can be biased in that they favour relevant documents containing topic title words. Instead of being dependent on the number of relevant documents for a topic [72] this effect was found to be wholly dependent on the size of the collection [27].

2.2.3.2 Reducing the Judgement Effort

In TREC-7 and TREC-8 the large number of submitted experiments for the traditional ad-hoc search task, (103 and 129), demonstrated that even pooling top $k = 100$ may result in an immense human effort to obtain a total of 80,345 (TREC-7) and 86,830 (TREC-8) assessments. A single assessor would need to work 24 hours a day for approximately four weeks, assuming he could decide about relevance at a rate of two documents per minute. As a result, some researchers proposed methods and studies on empirical data to reduce the assessment effort.

The most obvious approach is to reduce the depth of the assessment pool, i.e. minimising the number of judgements per topic. An empirical study that restricted the depth of the pool to a rank cut-off level of 20, 10, and 5 by algorithmically selecting documents, showed that even small pools are good approximations to evaluations using larger pools (50 or 100) [35]. However, such an approach might harm the reusability of a test collection. Another option is to formulate topics that either have a small number of relevant documents by definition, such as known item search [16] or constrain the document set in a specific way. This restriction dramatically reduces variance across topics and restricts the information about the difference of systems accordingly. A strategy named “Move-to-Front pooling” [43] introduced a policy to change the priority ordering of the pool by assuming that documents at higher ranks, and documents from systems whose recently discovered documents were relevant, are more likely to be relevant. A similar approach started with judging a shallow pool first, and extends the pool using extrapolation from systems and topics [199].

More recent works have focused on sampling the documents for the assessment of relevance. Both uniform random sampling [197] and stratified sampling, a combination of pooling and random sampling [11], have been studied. Both approaches treat incomplete relevance judgements as a sample drawn from a complete set of judgements, and use statistical methods to estimate the actual values of effectiveness metrics. Another idea is to select a subset of documents for relevance judgement and assign weights to these documents. The higher weights indicate higher value in determining a difference between systems with respect to a given evaluation metric [34, 36]. However, this “Minimal Test Collection” method produces values of metrics that are difficult to interpret and might be problematic when comparing pooled and unpooled systems.

Another line of research focused the selection of a subset of topics that could be sufficient to estimate the ranking of systems under evaluation. By means of a network analysis on TREC-8 experiments, the initial hypothesis that some topics are better suited to distinguish between systems than others, was tested and verified [132]. However, serious concerns regarding generalisation were expressed in a recent follow-up study [146]. It was shown that a subset of topics that works well for distinguishing one set of systems may not be significantly better than a random subset of topics for another set of systems.

Using automatic relevance judgements for evaluation represents another group of ideas to drastically reduce human effort in system evaluation. A first approach was to form a top 100 pool from systems and randomly sample documents from the pool, assume those to be pseudo-relevant, and evaluate the systems using pseudo-relevance judgements [169]. Manually defining a number of query aspects that represent different articulations of a user information need was proposed as an alternative method [58]. In the latter study, a single IR system was used for evaluation, assuming the union of top k documents of the generated query aspects to be relevant, where each aspect represented a system. Although evaluation without human judgements seems to be a promising idea, the methodology and corresponding approaches have been criticised for producing a ranking according to the popularity of systems, rather than the actual performance in terms of some effectiveness metric [12]. A recent study across several TREC test collections improved automatic system ranking by applying the idea of topic subset selection [76].

In the context of the Web 2.0 with peer collaboration and user-generated content, new applications have recently emerged. A promising phenomenon called crowdsourcing has attracted researchers from the IR evaluation community, based on its potential to reduce the costs for relevance assessments. Crowdsourcing describes a process where a large task is sliced into relatively small items of work that are then outsourced and completed by human workers [3]. Usually, the workers are offered a small financial compensation for their work. Popular platforms at the time of writing are Amazon Mechanical Turk[3] or CrowdFlower[4]. Inspired by the potential applications of crowdsourcing for IR evaluation, a framework named TERC (technique for evaluating relevance by crowdsourcing) was developed [5]. In this article the authors highlight advantages like fast and cost-effective evaluation and also describe potential pitfalls, like the artificiality of the assessment task. One of the first contributions on the topic of crowdsourcing aimed at answering the question of whether TREC assessors can be replaced or not [4]. By running an experiment on a single topic and asking the workers

---

[3] `http://www.mturk.com`, retrieved on March 1, 2012

[4] `http://crowdflower.com`, retrieved on March 1, 2012

to decide on the relevance of 29 selected documents (15 were known to be relevant and 14 were known to be not relevant) they found that agreement across the workers was high for relevant documents and considerably lower for those not relevant. Regarding their initial question, they concluded crowdsourcing could be a useful alternative in obtaining relevance judgements.

When crowdsourcing is deployed to collect relevance information, assuring quality becomes an important issue. Some researchers have already addressed the problem and thoroughly investigated possible variables and mechanisms to control and assure quality. A standard approach is to incorporate a training phase (or a qualification test in a restrictive manner), and to apply subsequent sporadic tests using training data [116]. Such qualification tests were shown to increase the quality of the outcomes [95].

In the same study the researchers were also interested in how the amount of financial compensation and the effort of the task affected the quality of the results. They observed that higher payment increases quality. Task effort introduced more spam, but when it was removed, quality was higher than for lower task efforts. Another work suggested to apply quality control to parts of the experimental design, i.e. clear formulation of instructions and good presentation of documents (e.g. highlighting query words) [3].

All of the presented strategies aimed at reducing the effort needed to develop new test collection. Although most of them demonstrated improvements repeatedly over traditional TREC-like designs, they share the downside of introducing additional errors to the evaluation process.

2.2.3.3 Inconsistency

One of the major criticisms of system-based IR evaluation is the use of inherently subjective relevance judgements. Consistency of judgements is closely related to this problem and represents another important issue of relevance assessments. A study on early TREC experiments showed that system rankings correlate well, even for different sets of relevance judgements, (obtained by asking three different human assessors to decide on the relevance of documents) [181]. It is also commonly accepted that relevance judgements from a single human assessor may change over time [22, 164]. Another potential issue that might affect the system ranking is how much knowledge the assessors have about the topic to be judged. Similar to previous results, a study that grouped judges according to their knowledge about topics found that obtained relevance sets are different, but similarly the resulting rankings of systems correlated well [14].

2.3 Effectiveness Measures

It has been pointed out that evaluation is a crucial part of the development process in IR. Consequently, numerous metrics have been proposed to assess the quality of search results. The Cranfield experiments compiled the foundations of modern IR evaluation and hence the first measures to summarise system quality were introduced at that time [39, 40]. Ever since, precision and recall are the most commonly used metrics to determine system effectiveness. Loosely speaking, recall measures how well a system did at finding all relevant documents and precision determines how well it did at rejecting irrelevant documents [45]. Although both measures were introduced in line with the Cranfield experiments, they have been used in some experiments with Boolean search systems before [158].

Many early IR systems produced such Boolean outputs: returning a set of documents matching a user query without any order or preference between documents. Based on these result sets, a contingency table (see Table 2.4) capturing several fractions of the total document collection was created [40]. It divides documents into different subsets based on the fact that they were retrieved in response to a query or not. It also

	Relevant	Non-Relevant	
Retrieved	a	b	a + b
Not Retrieved	c	d	c + d
	a + c	b + d	a + b + c + d

Table 2.4: Contingency table representing fractions of collections.

separates the actual ground truth information on relevance for each document being either relevant or not.

Three initial measures for retrieval effectiveness were derived from the contingency table by combining cells that represented subsets of search outputs. These are precision, recall and fallout. Precision is defined as the proportion of retrieved documents that are relevant, recall measures the proportion of relevant documents that are retrieved and fallout determines the fraction of non-relevant documents that are retrieved. The corresponding equations are reproduced below.

$$Precision = \frac{a}{a+b} \quad (2.1) \qquad Recall = \frac{a}{a+c} \quad (2.2) \qquad Fallout = \frac{b}{b+d} \quad (2.3)$$

A possible interpretation of Table 2.4 is to treat retrieval as a classification problem. A binary classifier would predict a document to be relevant or not and would return a corresponding un-ordered set of documents. Given this observation two kinds of errors can occur. First, a document might be retrieved (or predicted relevant) when it is actually not relevant, which is referred to as a false positive. Second, a false negative is a document that is relevant, but not retrieved (or predicted to be relevant). Besides the measures in Equations 2.1, 2.2, and 2.3 some other measures that are used for classification problems are also based on contingency tables. Two of these metrics are specificity and accuracy which take into account the number of true negatives, i.e. the number of non-relevant documents that are not retrieved in retrieval parlance.

Since current search tasks are based on collections containing millions of documents and only a small fraction of those is relevant to an arbitrary query, treating retrieval as classification may return counter-intuitive results. A retrieval engine that is trained to minimise classification errors could simply focus on true negatives and would consequently tend to return no documents for any query issued. However, the distribution of relevant documents is not always skewed like this, for instance for smaller collections which are more likely to feature uniform distributions for relevant and irrelevant documents. Therefore, some researchers questioned the dominant application of precision and recall for retrieval evaluation [141]. In spite of this criticism most current effectiveness metrics are still based on precision and recall.

Similar to the phenomenon between query throughput and query latency of search systems there is a trade-off between precision and recall. Let us consider two different retrieval systems theoretically: system A aiming at optimal precision and system B trying to maximise recall. System A could simply return just the first relevant document for each query, because returning more documents would increase the probability of degrading precision. In contrast to that system B would simply return all documents to any query since it guarantees all relevant documents are contained in the results. Consequently, a system that tries to optimally answer a user query has to achieve both a high precision and a high recall, i.e. returning as many relevant documents and rejecting as many non-relevant documents as possible. Already in the 1950's researchers became aware of the interconnection between recall and precision when they were studying the effectiveness of library indexing systems [98]. The inverse relationship between the two measures was first identified with the Cranfield experiments [39].

An important research question was how precision and recall could be merged into a single measure. A number of possible summary measures were surveyed by van Rijsbergen [179]. Based on this research, a measure called *e* was later proposed [180]. It is the precursor of the widely-used *F-measure*, which is defined as presented in Equation 2.4.

$$F = \frac{1}{\alpha \cdot \frac{1}{precision} + (1-\alpha) \cdot \frac{1}{recall}} \tag{2.4}$$

The constant α is used to emphasise how much effect recall or precision have on the resulting *F-score*. In a common configuration α is set to a value of 0.5, which is defined as the harmonic mean of recall and precision. The resulting definition for the F-measure is:

$$F = \frac{1}{\frac{1}{2} \cdot (\frac{1}{precision} + \frac{1}{recall})} = \frac{2 \cdot precision \cdot recall}{(precision + recall)} \tag{2.5}$$

The advantage of the harmonic mean over the arithmetic mean, is that it is sensitive to small values and that it is not affected by outliers that are unusually larger. Given the two theoretic systems A and B that were "optimised" for precision and recall, summarising by using the arithmetic mean would return F-scores greater than 0.5, although either recall (system A) or precision (system B) are close to 0, (assuming that the issued query has multiple relevant documents and the collection contains a large number of documents that are not relevant). In contrast to that, the harmonic mean will be close to 0, accounting for small values of recall or precision.

2.3.1 Evaluating Document Rankings

Early IR systems implemented and used in the 1950's and 1960's were based on Boolean matching and so they returned unordered lists of documents in response to every query. In contrast to that, modern retrieval systems rely on other retrieval models and return ranked lists of documents. The IR community focused on adapting recall and precision as evaluation metrics to assess the quality of these ranked lists. However, further definitions were needed to be able to use the two measures with document rankings. Because of the limited utility of the original definitions of recall and precision, other formal definitions were proposed [173] and verified based on empirical data [125, 174].

A straightforward approach to adapt the definition of recall and precision would be to calculate corresponding values for every position in the ranking. But queries may feature a large set of relevant documents or the document ranking might be weak, i.e. relevant documents are widely distributed in the ranked list. Therefore, it was proposed to cut off the ranking at specific positions in the ranking and to calculate precision at those positions. This measure is called *precision at rank cut-off r* (or $P@r$ in short). Its definition is given below (see Equation 2.6), where *rel(i)* is a function that returns either 0 or 1 as a notion of binary relevance for a given document position *d* in the ranking.

$$P@r = \frac{1}{r} \sum_{i=1}^{r} rel(i) \tag{2.6}$$

It can be seen that, if $P@r$ is higher for one ranking than for a second ranking, the latter must consequently contain less relevant documents. The measure is typically used to evaluate web search, where the results are typically presented on a single page that contains a restricted number of documents (10 or 20). The example in Table 2.5 illustrates rankings obtained for two topics A and B, where topic A has five and topic B has three relevant documents in a corresponding collection. Recall and precision are presented for the top ten positions of both rankings. However, using smaller cut-off levels changes the goal a retrieval system should aim for. If only documents ranked above position ten are part of the evaluation, the goal of the search task changes from finding *as many relevant documents as possible* to finding *at least ten relevant documents and rank them at the top of the list* for every query.

	Rank	1	2	3	4	5	6	7	8	9	10
Topic A	Rel.	1	1	0	0	1	0	1	0	0	1
	Prec.	1.0	1.0	0.67	0.5	0.6	0.5	0.57	0.5	0.45	0.5
	Recall	0.2	0.4	0.4	0.4	0.6	0.6	0.8	0.8	0.8	1.0
Topic B	Rel.	0	1	0	0	0	1	1	0	0	0
	Prec.	0.0	0.5	0.33	0.25	0.2	0.33	0.43	0.38	0.33	0.3
	Recall	0.0	0.33	0.33	0.33	0.33	0.67	1.0	1.0	1.0	1.0

Table 2.5: Recall and precision for two arbitrary topics.

2.3.2 Visual Representation

To avoid this problem, one option is to summarise document rankings by calculating precision at defined levels of recall. Typically, these points of recall are 0.1 increments from 0 to 1 resulting in eleven standard recall points. Using the full range of recall gives the desired result of capturing the complete document ranking, even if not all relevant documents were retrieved. The values are usually presented in tables or by using visual representations named *recall-precision graphs*. However, computing precision at standard levels of recall can be difficult if there are only a few relevant documents (e.g. the values in Table 2.5). Additionally, individual queries may produce different shapes in visual presentation, which makes them difficult to compare. Figure 2.2 shows the recall-precision graphs for the example topics A and B from Table 2.5. To obtain precision values at all standard recall levels, interpolation is used to estimate non-existent values. A standard definition of interpolated precision P_r is given in Equation 2.7, where a missing recall level r equals the highest actual precision value for any level of recall $r' \geq r$:

$$P_r = \max_{r' \geq r} P_{r'} \tag{2.7}$$

The assumption lying behind this is that an ordinary user would be willing to look into some more documents given the chance that it would increase the proportion of relevant documents in the viewed set. Additionally, it provides a definition of inter-

Recall	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Topic A	1.0	1.0	1.0	1.0	1.0	0.6	0.6	0.57	0.57	0.5	0.5
Topic B	0.5	0.5	0.5	0.5	0.43	0.43	0.43	0.43	0.43	0.43	0.43
Average	0.75	0.75	0.75	0.75	0.71	0.51	0.51	0.5	0.5	0.46	0.46

Table 2.6: Precision at standard recall levels computed using interpolation.

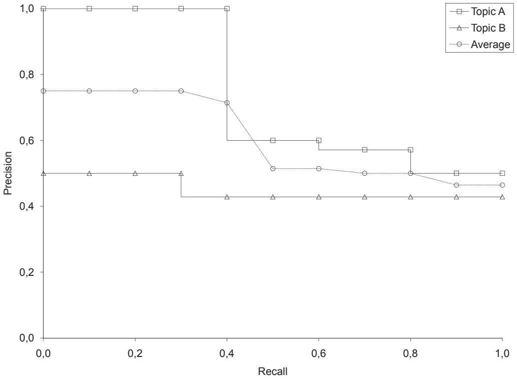


Figure 2.2: Recall-precision graphs based on standard recall levels.

polated precision at the recall level 0. A standard way to obtain a single graph for a set of topics is to convert recall-precision values for each query to precision values at standard recall levels by means of interpolation. Precision values for all queries at each standard recall level are averaged arithmetically. Table 2.6 presents interpolated precision values for the example topics A and B from Table 2.5 as well as their average at standard recall levels. Figure 2.2 illustrates corresponding recall-precision graphs.

Although being inconsistent with the interpolation method, averaged recall-precision curves are commonly presented by simply connecting the precision values at standard recall levels. Averaged recall-precision graphs are used throughout current evaluation campaigns to visually compare top performing systems.

2.3.3 Summarising Document Rankings

Even though recall-precision curves characterise the system performance achieved on a single query, comparisons of different queries can be difficult. For this reason another method to summarise document rankings in a single number was proposed at the second TREC conference [71]. The measure was called *non-interpolated average precision*, referring to other figures such as interpolated precision, which was commonly used until then. By now it is the most frequently used metric for the evaluation of ad-hoc search, and referred to as average precision AP. Its formal definition given in Equation 2.8 is based on the definitions from Equation 2.6, where N is the total number of documents in a ranking and R denotes the total number of relevant documents. Note that P@r is used within the sum, which is the reason for passing index values of *i* to P@r denoting $r = i$.

$$AP = \frac{1}{R}\sum_{i=1}^{N} rel(i) \cdot P@r_i = \frac{1}{R}\sum_{i=1}^{N} rel(i) \cdot \frac{\#\, relevant\, up\, to\, i}{i} \tag{2.8}$$

It can be seen from that the average precision computes the mean of precision values for rank positions with relevant documents. Precision values from non-relevant documents are discarded. Consequently, average precision is just the sum of precisions at retrieved and relevant documents, divided by the total number of relevant documents for a query. A similar single figure measure is *R-precision*. It is defined as $P@r$, where r is the total number of documents relevant to a query. Although R-precision is usually reported in evaluation exercises, it is used rarely for rankings of systems. Average precision and R-precision share the feature of approximating the area under the recall-precision curve [141], which is why both are reported to be highly correlated.

2.3.4 Averaging over Query Sets

Retrieval systems are usually compared based on a number (typically 50) of different queries asking for a further aggregation measure. Similarly to average precision, a desirable property of such a metric would be to summarise document rankings for a set of queries into a single number. A straightforward way to achieve this is to arithmetically average the non-interpolated average precision values for all queries. This measure is called *mean average precision* or MAP in IR literature. It is by far the most widely used figure in research articles on IR evaluation. Given a query $q_i \epsilon Q$, MAP can be easily defined using the average precision as follows:

$$MAP = \frac{1}{\|Q\|} \sum_{i=1}^{\|Q\|} AP_i \tag{2.9}$$

The resulting MAP value has a linear scale, i.e. a certain interval for two queries at the lower end of the scale has the same influence on the result as the same interval for two other queries on the upper end of the scale. Recently, a lot of research has focused on how difficult queries are accounted for by such a linear scale. In the course of that development, system *robustness* was identified as an important problem to focus on. The idea of robustness is that an IR system should return at least passable results for every query submitted by a user. Consequently, to assess the robustness of systems more attention needs to be drawn to the lower part of the scale of an evaluation metric. However, there is no obvious solution to adjust MAP to emphasise on difficult topics. Instead, a different way of averaging was introduced. *Geometric mean average precision* or GMAP solves this issue, because it multiplies AP values. A formal definition is presented in Equation 2.10.

$$GMAP = \exp \frac{1}{\|Q\|} \sum_{i=1}^{\|Q\|} \log AP_i \tag{2.10}$$

Using the logarithm (or the product) over AP introduces a mathematical problem when a system is unable to identify any relevant document for a specific query. In that case AP equals zero and causes GMAP to be undefined or zero respectively. In IR evaluations some queries are always hard to solve. Given that two systems would fail on different queries they can not be ranked using GMAP. A pragmatic solution to this problem is to introduce an arbitrary small quantity ϵ that is added to all AP values before taking the logarithm, and subtracted afterwards (see Equation 2.11) [143]. The choice of an appropriate value for ϵ has caused some discussion in the IR community, mainly because identical values have to be used to assure comparability across evaluations. Another option is not to adjust the values of AP unless they are zero, assuming that system failure occurs only in rare cases, and choosing ϵ to be smaller than the lowest possible AP values. In a typical evaluation each ranked list contains 1,000 documents resulting in a minimum AP of 10^{-3}. Thus, the maximum for ϵ should be 10^{-4}. Several evaluation tracks that addressed robustness of IR systems used $\epsilon = 10^{-5}$ [49, 184].

$$GMAP_\epsilon = \exp(\frac{1}{\|Q\|} \sum_{i=1}^{\|Q\|} \log (AP_i + \epsilon)) - \epsilon \tag{2.11}$$

2.3.5 Multivalued Relevance

All measures introduced so far are based on relevance judgements indicating a document to be either relevant or irrelevant. These metrics fail to consider multiple degrees of relevance. However, in real search scenarios some documents may appear to be more relevant to an information need than others. The family of *cumulative gain* measures have been proposed based on a model of user search behaviour [90, 91]. Fundamental for this user model are the following two assumptions: (1) highly relevant documents are more beneficial than marginally relevant documents, and (2) the lower a relevant document appears in a ranking the less valuable it is for a user, since it is less likely to be examined. Within the framework of cumulated gain-based measures, graded relevance is assumed to be a numerical measure of usefulness or gain obtained by examining a document at a specific rank. To account for the second assumption, relevant documents that appear at lower positions of the ranking are discounted by using a function of the rank.

The *discounted cumulative gain* (or DCG) at any particular rank r can then be defined as presented in Equation 2.12, where rel_i is the graded relevance level of a document at rank position i. The discount or reduction factor is represented in the denominator of the sum. A different version of a discount function [31] that was proposed for evaluation of web search results is reproduced in Equation 2.13. Given a set of graded relevance assessments, the latter alternative significantly increases the impact of retrieving highly relevant documents first.

$$DCG_r = rel_1 + \sum_{i=2}^{r} \frac{rel_i}{log_2 i} \quad (2.12) \qquad DCG_r = rel_1 + \sum_{i=2}^{r} \frac{2^{rel_i} - 1}{log(1+i)} \quad (2.13)$$

Note that there is a close relation between the present DCG measures and P@r. Both measures accumulate precision values up to a specific position in a document ranking. While for P@r using P@10, P@20 or P@50 are commonly used figures, DCG is mostly used at higher cut-off levels such as five or ten. Similarly to the effectiveness measures presented before, DCG can be normalised resulting in a measure named *normalised discounted cumulative gain* (or nDCG). The standardisation is achieved by relating DCG for a ranking against the *IDCG* obtained for the perfect ordering of relevant documents for a query, i.e. sorting relevant documents in decreasing order of relevance grades (see Equation 2.14). The resulting interval for nDCG is the same as for other commonly used measures like AP or MAP and lies between zero and one (both being part of the scale).

$$NDCG_r = \frac{DCG_r}{IDCG_r} \tag{2.14}$$

Besides the popular NDCG metric, a number of other measures based on graded relevance have been proposed. Some researchers argued that using a discount function based on variants of logarithms may not be an appropriate model for user search behaviour on a ranked list of documents. Instead of using a general reduction for lower ranks, they proposed to model the patience of a user in a measure called *rank-biased precision* [133]. This is achieved by incorporating the probability of a user's persistence when moving to documents at lower positions in a ranking.

2.3.6 Other measures

Many of the evaluation campaigns that aim at specific aspects of information access developed test collections with graded relevance judgements. Measures like *average weighted precision* (or AWP) or the *Q-measure* were introduced in the course of the Asian language exercise NTCIR [152]. Another example is *extended cumulated gain* (or XCG) that was developed at the INEX campaign to assess retrieval of document passages, when only fragments of large documents are relevant for a query [96]. Some applications of information access require focusing on specific aspects of effectiveness, but do not necessarily rely on graded relevance. In the domain of legal search, for example, the main objective of a search task is to find every relevant document, since a single missed document may invalidate a patent application, or could help a lawyer in a current case. As a result, the evaluation metric for this scenario has to focus on recall, of course without undervaluing precision. Such a metric was recently proposed and tested in a prior art candidate search scenario [123].

Other applications, like Question Answering, or known item search, require measures that capture the position of a few or just a single relevant document. *Mean reciprocal* *rank* (MRR) was the first metric proposed for evaluation of such scenarios [94]. Like the name suggests, the measure simply calculates the reciprocal of the rank at which the first relevant document is retrieved and the resulting values are then averaged over a set of queries. Another group of metrics is used for evaluating search without the presence of explicit relevance statements. Those figures are typically based on preferences of documents, where a preference ranking is usually obtained from query logs. Evaluation based on preference is a comparison of two ordered sets of documents with one being the ideal ranking. This interpretation reveals that evaluating preferences and multivalued relevance are closely related (see Equation 2.14). The earliest measure for comparison of rankings or ordered sets is *Kendall's tau coefficient* [97]. Despite its age, this metric is very popular in the IR evaluation community. The *binary preference* (or BPref) measure was especially designed for handling preferences based on incomplete binary relevance information [30].

The present section discussed a small selection of popular effectiveness measures used for IR evaluation. It showed, that the growth of search applications and corresponding evaluation scenarios led to new methods to assess search quality. This development can be expected to continue in the future. All these efforts serve the ultimate goal of alleviating user access to information. This thesis aims at bringing more transparency to system-evaluation in laboratory environments. Nevertheless, using various measures is important, especially in laboratory experiments both as a sanity check for any conclusion that may be drawn, and to detect specific features of rankings that may only become visible because of this comparison of metrics. In the present experimental investigation popular measures for ad-hoc search, such as AP, MAP, GMAP, but also nDCG, will be used to evaluate the effectiveness of system configurations.

2.4 Statistical Significance

The purpose of experimental IR evaluation is to assess which of the systems or configurations under investigation perform better with respect to a given evaluation measure. To be able to answer such a question, differences between metrics have to be interpreted. A straightforward approach might be to calculate relative or absolute differences from the effectiveness measures obtained for varied systems. However, the mere presence of an observed difference does not allow to draw the conclusion whether this difference is meaningful or not. The reasoning behind this is straightforward. Evaluation measures are designed to summarise the data obtained from retrieval experimentation, whose output is usually a list of ranked documents for a set of queries. Thus, calculating an effectiveness metric could be considered as lossy compression of the actual data (ranked document lists). Consequently, such an abstraction may hide important details about the differences between search algorithms or retrieval systems.

Significance tests are statistical tools designed to draw conclusions about large sets of experimental data. In their application for IR evaluation, significance tests are typically used to compare the ranking of two search systems or varied configurations of a single system. Such tests are based on a *null hypothesis* (H_0) and estimate the probability p of whether the observed differences are likely to be caused by random chance or not. In IR parlance, the null hypothesis states that there are no differences between two retrieval systems under examination, and that observed differences only occurred by random chance.

An experimenter may conclude that H_0 can be rejected if the probability p, also termed *p-value*, obtained by applying a test statistic, is below a predefined threshold α. Typical values for the *significance level* α are 0.05, or 0.01 in a more rigid setting. If the null hypothesis is rejected, it is a common approach to assume that the *alternative hypothesis* (H_1) is valid, i.e. that there is a significant difference between two systems.

In fact, instead of observing just meaningful differences, an IR researcher may rather be interested in the question of which of the two systems is better. These tests are called *one-sided* or *one-tailed* significance tests, because only one side of an underlying probability distribution is of interest. In the scenario of determining the difference of two retrieval engines, both systems are usually tested on a common set of topics and the scores for each topic are compared on a per-topic basis. Thus, statistical tests in IR evaluation are usually *matched pair* experiments.

Since testing significance requires a Boolean decision about the validity of a hypothesis two kinds of errors may occur. A *Type I* error appears when H_0 is rejected, although in fact it is true, which is called a false positive event. In contrast to that a false negative results from accepting H_0 although it is in fact false. These errors are called *Type II* errors.

Similarly to the number of available IR evaluation measures, various types of significance tests emerged in research on statistical methods. They are based on specific assumptions and therefore lead to different error rates and consequently to a different balance between the two error types. A common measure for the quality of a significance test is *power*, which determines the probability that a test rejects H_0 correctly. The power of a statistical test can be defined based on its test statistic. More powerful tests make more assumptions about the data to reduce the chances of Type II errors. A possibility to improve the power of a test is to increase the sample size, i.e. the number of queries used to compare two systems.

Common significance tests used in IR evaluation are presented in the following. An example using real data is used to illustrate the differences between these tests, all of which having their own test statistic and null hypothesis. Figure 2.3 shows differences between two system configurations from our experimental evaluation for individual topics. Two distinct configurations are compared on a standard news collection using a set of 84 topics that have been used in the Grid@CLEF track in 2009 [62] and at several other CLEF tasks before. The example system configurations achieved a MAP of 0.5657 (system A) and 0.4859 (system B). Based on this effectiveness metric, a relative difference of 0.1641 and an absolute difference of 0.0798 are found. Given these observations, the question of whether those measured differences are statistically

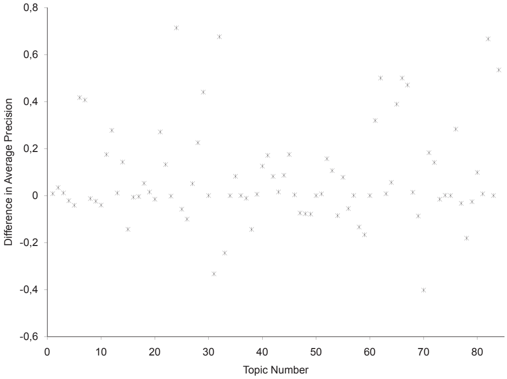


Figure 2.3: Differences of AP for two system configurations on per-topic basis.

significant or not still remains open. However, the criterion, by which the difference between system A and B is judged, is well defined. Since MAP is used as effectiveness metric, the differences in MAP forms the basis of the test statistic. The next step is to formulate a null hypothesis H_0 and to chose a significance test in order to determine whether the corresponding H_0 can be rejected or not.

Statistical tests can be roughly divided into *non-parametric* and *parametric* tests. Non-parametric tests make fewer assumptions about the underlying data and tend to generate more false negatives (accepting H_0 and concluding that there is no difference between two systems, although there is in fact a difference between them) [158]. In contrast, parametric tests are more powerful, especially in reducing the chance of such Type II errors, but they make more assumptions about the data. If an underlying assumption of a statistical test is violated, the rate of Type I errors is very likely to increase. A selection from these two groups of tests which covers those commonly used in IR evaluation research, are considered in the following subsections [167].

2.4.1 Student's t-test

The most commonly used significance test in IR evaluation is the *t-test*, also referred to as Student's t-test. The null hypothesis for the t-test is that the values from two systems are random samples from the same normal distribution, and that the mean of the distribution of differences is zero [45, 64]. The test statistic is given in Equation 2.15, where N is the total number of paired values (i.e. the number of topics), μ_D denotes the mean of the differences of all matched pairs, and σ_D gives the standard deviation of the differences. The constant μ_0 is usually set to zero in IR evaluation. It can be used to test whether the mean of differences is significantly different from μ_0. Given the data from the example systems A and B in Figure 2.3 the following values result for the test statistic: $\mu_D = 0.0797$, $\sigma_D = 0.2122$, and $t = 3.4434$. Using the sample size of the example experiment ($N = 84$) the resulting p-values are $\alpha = 0.0005$ (one-tailed) and $\alpha = 0.0009$ (two-tailed). Based on the assumption that the pre-defined significance level is set to 0.05, the null hypothesis H_0 can be rejected. Thus, using Student's t-test, and the given experimental data, an IR researcher would conclude that the example systems A and B are significantly different.

$$t = \frac{\mu_D - \mu_0}{\sigma_D} \cdot \sqrt{N} \tag{2.15}$$

$$\mu_D = \frac{\sum_{i=1}^{N} B_i - A_i}{N} \tag{2.16}$$

$$\sigma_D = \sqrt{\frac{\sum_{i=1}^{N} (B_i - A_i - \mu_D)^2}{N}} \tag{2.17}$$

Some IR researchers argued that the underlying assumption of normally distributed samples is typically not met for experimental evaluation, based on IR effectiveness measures. This is true especially if the size of the sample (the total number of queries) is small. This assumption was also verified by empirical studies [159]. Another source of criticism is concerned with the assumption of the t-test that the sample data is measured on an interval scale. Measuring data on an interval scale means that equal differences on different parts of the scale have the same meaning. The motivation for the GMAP metric (see Section 2.3) shows that this assumption is questionable. Nevertheless, recent experimental studies compared several tests that are commonly used in IR evaluation and found that the t-test produces similar p-values when compared to other tests, which are distribution-free and do not rely on random sampling [159, 167].

2.4.2 Randomisation

When using a randomisation test, the null hypothesis is that the two systems under examination are identical [64]. Hence, on a given test collection, consisting of a document collection, a fixed number of topics, and a set of relevance judgements, there is no effect of system A, compared to system B, on the target effectiveness metric MAP. Given this assumption, one could imagine a single system C, that produced the rankings for both systems A and B. Consequently, the present results could be just labels for different outputs of system C, obtained by issuing every topic twice to system C and distinguishing the outputs using the labels A and B. Following this idea, the resulting labels can be thought to be arbitrary. The two systems are compared on a sample of 84 topics and therefore, 2^{84} ($\sim 1.93 \cdot 10^{25}$) permutations of labelling the scores for systems A and B exist under the given hypothesis H_0. In addition, all of these outputs for system C are equally likely to occur. One of these labellings is precisely the one that generated the differences reported in Figure 2.3.

When all permutations are generated the difference between the system A and B can be calculated for each of them. Given the differences for the 2^{84} permutations, the number of times such a difference is greater or equal to the observed difference of the example ($0.5657 - 0.4859 = 0.0798$) can be accumulated and divided by the total number of permutations (2^{84}). The resulting number would be the precise one-tailed significance level for H_0. The two-tailed p-value could be obtained by counting the number of times, when the *absolute value* of differences is equal or greater than the observed difference.

Even today computing 2^{84} permutations for a dataset with only 168 discrete values takes an unacceptable amount of time, especially when contrasted with the amount of information that is gained as a result. Thus, it seems reasonable to take a random sample from the total number of permutations in reasonable size. Evidently, the larger the size of the sample the more accurate the resulting estimate for the significance level will be. In the time when the randomisation process was formulated, such a test was in fact infeasible due to the missing availability of cheap automated computing. Therefore, further assumptions and constraints were inevitable in order to reduce computational effort. One of the approaches to achieve such simplification was to replace the actual differences with ranks of the scores.

2.4.3 Wilcoxon Signed Rank Test

Since the *Wilcoxon signed rank test* is a simplification of the randomisation test, it is based on exactly the same null hypothesis. It was especially designed to obtain an approximation of the significance of differences in a fast way [191]. In order to limit the computational cost, some additional assumptions were needed. The test assumes that paired differences of effectiveness values can be ranked and that their magnitudes can be discarded. To obtain the test statistic, the absolute values of paired differences are ranked in ascending order first. In a second step, depending on whether the paired difference was greater or smaller than zero, each rank is given the sign of the original difference. Zero differences are discarded. A list of increasing integers mixing positive and negative values is obtained as a result. It is used to accumulate the test statistic w as given in Equation 2.18, where R_i denotes a signed rank integer and N equals the total number of non-zero differences in paired scores.

$$w = \sum_{i=1}^{N} R_i \tag{2.18}$$

For the example given in the beginning of this section six zero value pairs were found resulting in 78 signed ranks and a value of $w = 1093$ as test statistic. The corresponding p-value of 0.0033 for a one-tailed test and 0.0065 for a two-tailed test allows the conclusion that H_0 can be rejected. As a result, the improvement of system A over system B can be regarded as significant. Due to the availability of fast personal computers the original purpose of Wilcoxon's test - to give a fast approximation of statistical significance - is not a valid condition any more. Several studies compared the test to others and suggested that it be omitted, because of high Type II error rates [87, 167, 190].

2.4.4 Sign Test

Similar to the significance tests introduced so far the null hypothesis of the *sign test* is that two systems have the same distribution of test samples. The precise formulation of H_0 is, that given a sufficiently large sample, the number of trials where system B is better than system A, is expected to be equal to the number of trials where system A is better than system B. This assumption allows further simplification. In contrast to the Wilcoxon signed rank test, where magnitudes of differences are roughly approximated using ranks, all information about the magnitudes of differences is discarded for the test statistic of the sign test. The test statistic is based on the binomial distribution and relies on the total number of sample pairs that are not tied. Since most IR effectiveness metrics use an interval scale from zero to one they could be calculated up to infinite precision. Hence, some definition for the minimum absolute difference d of a matched pair is needed. A value of $d = 0.01$ was proposed for empirical significance testing using the sign test [167, 180]. This variant of the test is termed *sign minimum difference test*.

For the example given in Figure 2.3, system A has the better performance in terms of AP on 48 topics. Therefore, the sign test returns a two-tailed p-value of 0.0535 based on 48 successes out of 78 trials, (note that the six ties are discarded). The result would allow the conclusion that H_0 can not be rejected, a conflictive observation with respect to the Wilcoxon signed rank test. The sign minimum difference test using $d = 0.01$, has 40 successes out of 66 trials and returns a p-value of 0.109 that is even larger. But it would also lead to the conclusion that system A and B are identical. The sign minimum difference test is naturally dependent on the value of d. When d is set to 0.05 the p-value for the given example decreases to 0.0153 allowing the rejection of the null hypothesis. Obviously, this sensitivity to the value of d is problematic as it can result in such contradictory results.

Both the Wilcoxon signed rank test and the sign test are variants of the randomisation test. All of these tests count the number of successes, using different statistics from the total number of permutations for the paired scores. Although the sign test seems to be inconclusive on the provided example data, it could be used to obtain an approximation for the significance of differences based on the number of successes a new IR method achieved over a baseline approach on a common set of queries. From a current point of view, the Wilcoxon signed rank test and the sign test are inappropriate as detailed statistical significance tests for IR evaluation, due to the simplicity of their underlying statistics. But they are still good enough to provide a rough idea about the significance of difference.

2.4.5 Bootstrapping

Another significance test that was proposed to be used for IR evaluation is the *bootstrap* test [57, 160]. The bootstrap test is not based on assumptions about normal or continuous distribution of the underlying data. Its null hypothesis is that the sample scores for two systems A and B are random samples from the same distribution. The goal of the bootstrap method is to estimate the population distribution of the data set. This is achieved by sampling with replacement from the observed samples. In the case of IR evaluation, data pairs of scores for system A and B are drawn until the total number of samples equals the number of queries used in the experiment. In the next step the test statistic is computed over the obtained set of matched pairs. Any test statistic can be used for the bootstrap method. Some researchers prefer the difference in the mean average precision [167] others use the test statistic from the t-test [153]. The procedure is repeated B (for bootstrap) times to create the bootstrap distribution of the test statistic. Values for the number of iterations B differ for empirical studies, ($B = 1,000$ in [153], $B = 100,000$ in [167]). In general, the more iterations are used, the more precise the estimation of the distribution of the sample population will be. The sampling process might produce biased samples and therefore the bootstrap distribution may be biased. The H_0 distribution is generally unknown except for the fact that the means of the scores are zero. Thus, a convenient way to adjust bias in the bootstrap distribution is to shift it so that its mean is zero [167].

Given the fact that modern computers are capable of generating large samples of small sets of matched pairs in a short time, it is possible to obtain a good approximation of the significance of differences between IR systems. Many recent studies use the bootstrap method, because it does not rely on normal or continuous distributions of effectiveness scores, i.e. it can be used with effectiveness metrics like GMAP or nDCG.

2.4.6 Concluding Remarks and Implications

As with the use of effectiveness metrics, no statistical significance test is suitable for every IR evaluation task. However, it is important to keep in mind what conclusions are to be drawn from the sample data and to ensure that the underlying assumptions of the test's null hypothesis, and the assumptions about the data under examination are met. An IR researcher who is interested in determining whether a new method works better on most queries from a given set of topics than a reference baseline, or not, could use the simple sign test. Given a specific effectiveness metric, if the question is whether the scores are significantly better for system A over system B, other tests such as student's t-test, randomisation or bootstrapping should be considered.

For the two example systems A and B with MAP of 0.5657 and 0.4859, the following test statistics were used to calculate statistical significance levels. The sign test returned a p-value of 0.0535 and the sign minimum difference test (with $d = 0.01$) gave a p-value of 0.109. The student t-test resulted in a p-value of 0.0009 and the Wilcoxon signed rank test's p-value was 0.0065. Given the assumption that a researcher declares p-values less than 0.05 to indicate significant difference for the examples above, all except the two sign tests are able to detect this significance. If a researcher uses one of the sign tests and fails to detect significance he might be spending more time developing methods that improve retrieval performance than actually necessary. In contrast to the suggestions in Section 2.4.4 the Wilcoxon test detected the present significant difference between systems A and B which indicates that it still gives a good approximation.

Another idea in early IR research was to use rules of thumb for determining significant improvements in effectiveness evaluation. A widely-used rule is based on absolute differences in MAP and states that any performance differences of less than 5% percent must be discarded [170]. Another vague formulation for the degree of significance without having a statistical test at hand was also proposed in [170]. Differences in the order of five to ten percent could be termed *noticable*, while differences larger than ten percent could be regarded as *material* improvement. Due to the lack of statistical foundation these rules of thumb are only rarely used in research publications. An empirical study on the reliability of test collections and effectiveness measures showed that, given a standard topic set with 50 queries, relative difference in effectiveness scores between two systems of ten percent or larger can be regarded as reliable [159]. Based on the data for the example used throughout this section, with an absolute difference of about eight percent, the given rules seem to agree with most of the actual statistical significance tests.

Some of the presented statistical significance tests were found to return poor estimates of the significance of differences [167]. However, it is worth noting that a curious researcher could use a selection of the present tests and take resulting significance levels as a sanity check for the obtained results and their implications. In contrast to that, a faulty scientific approach would be failing to detect significant difference on a sample of effectiveness scores using a test T_1 and, because of being in the hope of finding significance, to test the same data set on a second test T_2. Furthermore, every IR researcher needs to be aware of the fact that statistical significance tests are generic tools to summarise observations and return a Boolean decision given a prior hypothesis. Therefore, they return estimates that are dependent on representative samples. In many cases of IR evaluation this is the main source of error resulting in unreliable conclusions from experimental evaluation. In addition to these objections, even when a researcher is able to detect statistical significance another question remains open. Are these results in fact practically significant? This question cannot be addressed by laboratory research and needs to involve models of users and user search behaviour. Another undesirable effect of the summaries resulting from statistical tests is that researches are tempted to omit detailed analysis of their experimental data, once they have found statistical significant difference. A recent alternative to overcome this problem, is the calculation of *confidence intervals*. In other areas of scientific research, they have replaced significance tests as the standard method for analysing experimental data [158].

2.5 Limitations of Traditional IR Evaluation

The presented key elements of IR effectiveness evaluation demonstrate that laboratory research only covers a small part of underlying real world problems. A test collection and its components constitute just a small sample from actual data collections. As a result, the design of the experimental test with the selection of documents and queries, as well as the process of obtaining relevance judgements and the choice of appropriate effectiveness metrics, affect the quality of the evaluation. More importantly, all these factors influence the reliability and the possibility of generalisation from any observation made in an experiment. Based on the success of TREC many researchers investigated aspects of the reliability and re-usability of test collections components to enhance the quality of subsequent test collections. A re-usable test collection needs to be designed in a way that ensures fair comparability with systems that did not contribute documents to the pool of relevance assessments. Thus, some level of variance from the systems under examination is needed to avoid a bias in the judgement pool.

2.5.1 Comparability of Empirical Experiments

The effects of topics, as well as inconsistency and incompleteness of relevance assessments on the ranking of participating systems, are subject to many studies in IR research literature [93]. A few recent studies focused on the development of effectiveness metrics that account for possible sources of variance in evaluation. Other approaches propose to adjust effectiveness measures according to possible bias from documents that were not judged for relevance. Some researchers suggested score standardisation approaches to allow meaningful comparison of IR systems across test collections. The main stream of research on the topic of reliability and re-usability concentrated on reducing efforts for the creation of new test collections. A careful selection of methods targeting at enhancing comparability and reliability of IR evaluation based on effectiveness is presented in the following.

A central question in IR evaluation is: how many topics are needed to obtain a reliable preference ranking of systems? Typically, researchers use large sets of empirical data to gain insights into the more complex underlying problem. One of the most influential works introduced the concept of *swap rates* for pairwise comparisons of systems [29, 183, 186]. In the first of these studies [29], given a pair of systems, the binary decision concerning which of them had the better performance was investigated by using different sizes of topic sets and various widely-used effectiveness measures. It was a first attempt to find statistical evidence on how the number of queries, the effectiveness measure used, and a definition of difference, affects the confidence of conclusions that can be drawn from IR experiments. Based on the results of that empirical study it was suggested that researches should be sceptical before drawing conclusions, even when

50 topics were used. The authors also recommended the use of multiple test collections to check the stability of observations. In [183], the pairwise swap rate approach was supplemented with empirical calculation of error rates based on the number of topics used in experiments. Error rates for large topic sets (up to 50) have been extrapolated by fitting an exponential curve to the given empirical error rates for topic sets up to size 25. A foundation based on statistical principles for the rather untidy extrapolation method was later given in [118]. The swap rate method was refined in [159] to consider only statistically significant differences between pairs of systems. Several significance tests were applied in this study and similar conclusions have been drawn. In a recent follow-up study [186] that did not rely on extrapolation, an approach for score standardisation across test collections was adopted [188]. The results showed that even for topic sets with 50 queries, statistically significant differences may result in high error rates for some effectiveness measures like P@10 or R-precision. As a consequence it is recommended that researchers should remain sceptical when they used only a single test collection.

All of the aforementioned research focused on just determining a fixed number of topics that are needed to be able to draw reliable conclusions. It is commonly accepted that more topics allow more confidence about the observations. Thus, the cost and reliability of a test collection are proportional to the number of topics included [186]. This reasoning motivated another stream of research that deals with reducing the cost for the creation of test collections. A number of approaches that focused on selective strategies to keep the judgement effort as low as possible was presented in Section 2.2.3.2.

2.5.2 Sources of Variance

Several studies investigated the effects of test collection components on system performance. The question had already been addressed after the first few TREC experiments. In [15] six different methods for data analysis were applied to TREC-3 results in order to gain insights into interaction effects between common sources of variance in IR evaluation. The researchers tried to answer the question of whether the observed differences in system performance are real and how these differences relate to the variance of topics and document collections. They fitted an analysis of variance model to the data and applied a special test for interaction. Three sources of variance were analysed: (1) systems, (2) topics, and (3) interaction. All three sources of variance were significant and the effect of the topics was two and a half times larger than the system effect. The observed interaction effect was about a fifth of the system effect. This result suggests that differences in effectiveness scores for systems given a topic are mainly caused by the nature of the topic and less due to the difference in retrieval systems. Similar observations have been found in a recent study that applied *classic test theory* and *generalisation theory* to IR evaluation [19]. The researchers tried to find an answer to the question of how good a test collection is. An alternative formulation of the problem is: how reliable is a present performance comparison? They argued that empirical analysis of test collections are data-driven assessments that focus on test results rather than test design. In generalisation theory all sources of variance are considered simultaneously. The authors studied topic, system and assessor effects on retrieval performance and found that using more queries instead of obtaining more assessments for topics should be preferred for future test collections. A limitation of the generalisation theory method is that it investigates the reliability of a defined test design given the number of items in each of the facets, i.e. if one is interested in which occurrence is more reliable, each of the facets needs to be analysed separately.

The question as to what extent reliable and reusable test collections contribute to the advancement of IR systems and theory was recently addressed in [9]. The authors studied 106 publications from 1998 to 2008 and found that a total of 83 variants of original TREC test collections were used. The result of the review and data analysis was that an improvement in terms of system effectiveness was not measurable over time, although the majority of published research claimed significant improvements. This finding is quite discouraging and is considered to be due to the rare use of competitive baselines. As a consequence statistical significance was often found.

However, the authors reported that many of the reported results were actually worse than the median at the original TREC tracks. This longitudinal meta-study impressively demonstrated the side effects of averaging measures, be it effectiveness metrics or statistical significance. Summary measures hide important details about empirical data obtained from experiments and should only be used after the detailed analysis of empirical results.

2.5.3 Longitudinal Analysis

At the time of writing, TREC is close to its 21st anniversary and other major initiatives like CLEF and NTCIR have celebrated ten years of successful IR evaluation recently. Over the period of time various test collections have been created. Document and test collections have been re-used in various retrieval tasks both at different evaluation campaigns and in private or academic evaluations. Due to the annual character of evaluation campaigns, many document collections had been used for only a few years. Respective test collections consisting of a fixed set of topics, system outputs, and corresponding relevance assessments are completely unique, i.e. the test collections were not re-used twice without changes to at least one of their elements. The *trec_eval* evaluation tool and the experimental results of experiments contributed to any task at TREC were made available in order to allow other research groups to compare their systems to the original experiments. Unfortunately, due to the practice of archiving the results of the experiments, and the description of the experimental set-up and its parameters separately, careful comparisons require a lot of work on the part of researchers who are interested in benchmarking a new model and its software implementation. Thus, it is a difficult task to track the process of the overall improvement of IR approaches.

The organisers of the CLEF workshops also developed a platform named DIRECT[5] which provides access to the system outputs and evaluation results for most of the

---

[5] `http://direct.dei.unipd.it/`, retrieved on March 1, 2012

retrieval tasks from 2000 to 2009. In contrast to the TREC website, DIRECT provides short descriptions for each of the experiments. These brief descriptions were, however, collected as free-text information, which means each participant included the information that he considered to be valuable. In addition to that brief information on the experiment itself, a general categorisation of the query generation process, which could either be manual or automatic, and the description of which part of the topics had been used, were also collected. Together with the appendices of major retrieval tasks which contain experiment meta-data and values for selected evaluation figures for each submitted experiment, the CLEF resources provide a fair amount of descriptive meta-data.

The availability of frequently used test collections, previous evaluation results, and publicly available IR systems allows a thorough investigation of the progress the IR community has made over time. However, only a few systems are available to the public, e.g. *EvaluatIR* [9]. The motivation for the development of this on-line repository is based on a critical review of IR publications from 1998-2008. The authors of that study claim that their system could be used by IR researchers and publication referees. Without belittling the efforts to create, operate and maintain a platform for longitudinal evaluation there are some smaller limitations that could be addressed in the future. The link between any research publication and corresponding empirical results is not tracked within the system. Since IR systems and their configurations are complex, this missing connection could still impede comparisons between systems. Especially because it may not be clear what the actual differences are between a system implementing a new method and a system that created the baseline reference. The possibility of comparing systems on component-level is another desirable feature for a resource like *EvaluatIR*. However, given current IR evaluation practice this seems to be an ambitious goal. A meta-data standard for IR experiment description could help to address the problem.

2.5.4 General Concerns

The preceding sections covered major aspects of IR evaluation using test collections. It may become obvious that many great ideas have emerged in the course of the development, from experiments based on the Cranfield paradigm, to the organisation of retrieval experiments like TREC. The achievements of the TREC organisers in institutionalising IR evaluation especially deserve a great deal of acknowledgement. However, there are also a few reservations that developed along with the success of TREC and its successors. Three of these concerns were presented in [144] and serve as a point of origin for the motivation of *component-level evaluation*.

A first objection to TREC-like experiments is their competitive nature. On the one hand it advocates the development of better methods, but on the other hand it is merely focused on results that are usually obtained from summary measures like MAP and others. Statistical significance tests create a further level of abstraction when comparing experiments. This is an undesirable trend of considering the evaluation of IR only as an investigation of the effectiveness of complete systems. As a result, detailed analysis of experiments and their resulting outcomes is omitted, although it is likely to provide insights into underlying issues.

Another problem with the manner in which TREC carries out IR evaluation is that it is laboratory based experimentation. As a consequence many of the resulting collections, formats, and methods serve as tools for laboratory experiments. However, any kind of laboratory experiment involves abstraction to allow the researcher to focus on certain aspects of a specific problem, and as a side effect other aspects of the problem under examination are ignored. But some aspects of the real world and their features are easier to abstract than others. Thus, inevitable biases are introduced into experimental studies. In the current IR evaluation paradigm one of the variables that resists abstraction is user variance. Formulation of requests and obtaining relevance judgements are a powerful abstraction to deal with user variance, but they limit the space of research questions that can be addressed. It simply avoids questions coming from actual user information needs that may express actual anomalous states of knowledge [17].

A final limitation given in [144] arises from the objection regarding abstraction. Evaluation of IR based on test collections is an artificial task that constrains the impact of experimental systems. Finding pieces of information is in fact just a sub-task of any other task, be it locating an email to follow-up on some pending communication process, or gaining knowledge from pieces of information that are needed to solve a piece of work at hand. Taking this point of view would involve looking at the goals of underlying tasks and how achieving a sub-goal, like finding a desired piece of information, could contribute efficiently to solving a wider problem.

2.6 Component-Level Evaluation

Systems (or experiments) are ranked according to a carefully selected effectiveness metric, like MAP or nDCG. Despite the laboratory nature of this experiment design, ranking complex systems which consist of many configurable components, could be considered superficial. Component-level IR evaluation is a concept that aims to overcome this drawback.

A component of a modern IR system is an element that is specifically designed to solve a particular task in the overall retrieval process. Prime examples for a traditional adhoc text retrieval experiment are stop-word filters or stemmers in the indexing stage, as well as ranking and relevance feedback models in the retrieval phase. Depending on the retrieval task at hand, other components might be necessary. The importance of each component and their orchestration, i.e. how all of the system components interact, may contribute to the overall system effectiveness for a search task under investigation. Details on how major components work and why different models or parameters are needed are given in Chapter 3.

In IR evaluation it is not possible to evaluate the effectiveness of a single component without plugging it into a complete IR system, because search effectiveness is traditionally measured on ranked document lists. Conversely, that means if one is interested in evaluating a single component (or its configuration), it is necessary keep all other components fixed. However, this will not reveal anything about the effect that changes to this particular component will have on the resulting search effectiveness when other components of the systems are altered.

Another important aspect of the evaluation methodology is concerning the question of which of the instances of any system component will be treated as suitable baseline for comparison. It might be clear in the case of stop-word removal or stemming, where omitting the components could be a straightforward baseline reference. But in the case of selecting or altering parameters of a ranking model this strategy will not work due to missing results. In order to properly analyse the effect of IR system components on search effectiveness, the instances of each of the components under investigation should represent the state-of-the-art.

In IR research, like in almost any field of research, most of the scientists specialised on a single, or only a few aspects of the retrieval process, which results in a distributed knowledge pool. A major challenge for component-level evaluation approaches is to overcome this distribution, i.e. to design an evaluation architecture that allows the combination of specialised components into a generic system. Some of the existing approaches and theoretical frameworks are discussed in the following subsections.

2.6.1 Models and Approaches

Attempts to overcome the limitations of traditional IR evaluation cycles have been already made in several evaluation campaigns. An overview on selected approaches is presented in [70]. The authors of this study compared existing methods with respect to the design of a generic framework for automated component-level evaluation.

Although automated evaluation of IR system components is appealing to promote IR evaluation research, the goal itself seems to be ambitious. Especially, with respect to the organisation overheads which are already obstructing traditional evaluation campaigns. It was pointed out before that the general problem with evaluation of IR system components is the global distribution of expertise in the field. Creating a platform to discuss and promote the expertise was one of the major goals of the TREC initiative. The growing complexity of IR systems, which is driven by the variety of applications for search technology, may have limited the utility of TREC-like campaigns in promoting the exchange of expertise in the past decade. In order to thoroughly investigate current obstacles for evaluation at component-level, the stakeholders and their fields of work need to be identified first.

IR researchers and evaluation campaign organisers are the most important stakeholders in the process. The former group contributes expertise in at least one of the IR systems components, typically by means of evaluating proprietary or open programming code. Organisers of IR evaluation tasks provide data sets and a specific problem description for the search task under investigation, and take care of an independent evaluation process. In traditional IR campaigns the data is simply distributed to the researchers, who run proprietary experiments and create standardised results. These results are then collected and assessed by the task organisers in order to create a ranking of experiments (or research groups!).

If the goal of the evaluation is to create a ranking of components, which of course has to be done for each type of component separately, an open question is how to realise the comparison. From a very abstract standpoint the issue could be broken down to decide either to distribute the input and output of each of the components, or to exchange or collect the programming code of each component. The latter approach could be realised by defining web interfaces or by organising actual exchange of programs or program code. In the following subsections, we present existing evaluation initiatives and models in areas related to IR, which tackle the problem of comparing technological approaches at component-level.

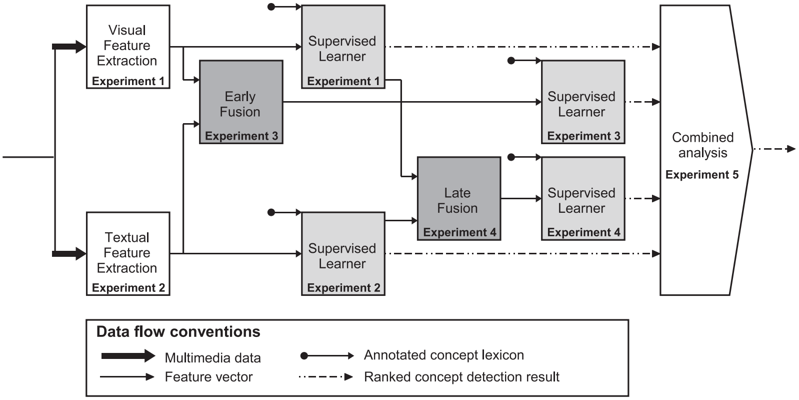


Figure 2.4: Component-level data flow in the MediaMill evaluation framework, redrawn from [168, p. 425].

2.6.1.1 Exchanging Programming Code

Publicly available open-source frameworks that cover complete systems or self-contained components are powerful methods in order to conduct empirical experiments at system and component-level. Such an experimental framework for semantic concept detection in video was made available to task participants for the *MediaMill Challenge* [168]. A component-based architecture enabled research groups to replace parts of the provided reference implementation with their own components that best fitted their own research interest and expertise. Figure 2.4 represents the basic data flow between components of the concept detection system implemented in [168]. In that framework, a system for video concept detection was broken down into three major parts, namely feature extraction, supervised machine learning and data fusion. At least one reference implementation was supplied for each of the components. Each research group which used the framework, could potentially exchange every component with an implementation of its own concepts.

An appealing feature of the approach is the distribution of a basic workflow, which is implemented as a software framework. For this reason no additional protocols are needed to exchange intermediary system or component outputs. But it also has the limitation that the general workflow, which was designed to solve the problem of detecting concepts in audiovisual content, is fixed [70]. As a consequence, radical ideas to approach the underlying problem might not fit into the defined workflow.

A further drawback regarding the utility for component-level comparisons is the fact that the provided components themselves are fixed, i.e. they are not further deployed. Assuming that one research group developed a quantum leap in the feature extraction component and another group made substantial progress in the machine learning component, the combination of both can only be tested if both groups contribute their results to the framework. For that reason it seems plausible that such a framework should either be used within an independent evaluation campaign or it should at least be publicly maintained, in order to ensure that the underlying technology reflects the state-of-the-art.

In the case of the *MediaMill Challenge* no information could be found as to whether the provided framework was accepted by the respective research community and thus it is hardly possible to draw a reliable conclusion regarding the success of the approach with respect to component-level comparison. There is, however, a recent publication that reports substantial improvement in two major components of video concept detection systems when comparing two state-of-the-art systems from 2006 and 2009 [7].

2.6.1.2 Uploading Component Code to a Central Server

Another approach to component-level IR evaluation is a centralised *algorithm-to-data model* [56]. In this scenario the test data and the general framework, including contributions from task participants, are managed on a central server. This approach was adopted for several instances of the *Music Information Retrieval Evaluation eXchange*

(MIREX). The decision to implement the *algorithm-to-data* model was made due to copyright restrictions on the test data, which did not allow to distribute the content for research.

In the framework of MIREX the data collections have been stored on a central sever and participating institutions were asked to upload the programming code of their algorithms for evaluation. MIREX already organised the tasks according to different sub-problems of the music retrieval domain. This evaluation design allowed the collection and comparison of models and respective programming code at a low-level, similar to the components of modern IR systems. However, the central server architecture introduced a number of additional challenges, which were discussed in detail in [56]:

- *Limited infrastructure capacity*
  In 2008, the data sets maintained by the MIREX organisers contained about 30,000 audio tracks, which required more than two terabytes of storage. Typical state-of-the-art music IR algorithms create intermediate feature sets that are much larger in size than the actual data sets. Due to space limitations, these intermediate feature sets had to be discarded in order to be able to store the raw experiment outputs. Regarding experiment re-usability this is a serious limitation.

- *Managing submitted algorithms*
  MIREX did not introduce any formal restrictions on the programming code and language of algorithms submitted for evaluation. But this introduced a heavy burden in the management of the central server. The organisers used guidelines to restrict file input and output formats and introduced coding conventions as well as error handling schemes to lower this burden. As stated by the MIREX organisers, managing and monitoring submitted algorithms consumed an average of 1,000 person hours per year [56, p. 251].

- *Re-running experiments*
  Due to the *algorithm-to-data model*, which resulted in limited storage, computation, and monitoring resources, there was no chance to correct bugs in submitted programming code during the evaluation cycle. Moreover, even re-running bug-fixed experiments on-demand was rendered impossible by the constraints of the centralised architecture.

- *Data integrity testing*
  Initiatives like TREC act as an independent authority in IR evaluation managing the creation of test collections and organising various search tasks. In the focused music retrieval community, test collections have been created by research groups that work in the field. Thus, another challenge for MIREX was to obtain such collections in order to incorporate them. Due to missing standards for the creation of music IR test collection, all of these test collections had to be checked for data integrity and correctness.

These issues show that implementing a central architecture for component-level evaluation also has a number of pitfalls. Most of them are related to constrained resources, which suggests that cost distribution is an important aspect for the design and implementation of a revised or new evaluation architecture. An ideal solution would be a framework, which requires almost no additional work for research groups, but also keeps low the expenses of maintaining the distributed or central evaluation. It is obvious that the two aspects conflict and every evaluation architecture should assess the stakes of this trade-off.

MIREX relieved the researchers from these burdens, but at the cost of the organisation authority. However, this solution was sacrificed for some other limitations. First, there might be objections from potential participants who do not want or are not allowed to submit programming code due to intellectual property restrictions. In fact, this could also limit the usefulness of the comparison, because submitted components might not necessarily reflect the actual state-of-the-art in the field. Secondly, due to the additional costs for the organisation and the resulting constraints (due to limited computational and human resources), the evaluation cycle takes at least one year. In comparison to traditional evaluation campaigns, which are conducted by central and mostly independent authorities, this appears to be normal. From the perspective of a researcher, however, waiting almost a year to be able to analyse the results of an empirical experiment will be discouraging.

The MIREX experiments showed, that evaluation using a central server architecture is an approach that deserves to be considered for the evaluation of IR systems at component-level. What needs to be taken into account is the additional burden, which is introduced at the expense of the task organiser in order to guarantee reliability of the experimental evaluation.

2.6.1.3 Exchanging Intermediate Output

A further method to deal with the shortcomings of traditional IR evaluation is to exchange intermediary results between participants. This approach was adopted for the *Grid@CLEF* pilot track in 2009 [62]. A framework for the generation of intermediate component output, named *CIRCO (Coordinated Information Retrieval Components Orchestration)* was created by the organisers. Each of the participating institutions was asked to integrate CIRCO into their system. In order to co-ordinate the components, a basic linear framework was proposed for the pilot task. It covered the indexing process of IR systems consisting of tokenizer, stop-word filter, stemmer, and lastly, ranking components. Figure 2.5 illustrates the planned data flow for *Grid@CLEF* and shows obligatory and optional components.

On the one hand, this experimental design allowed participants to evaluate system components on-site without integrating the program code of other groups. But on the other hand, it turned out that the accumulated size of the intermediate output was

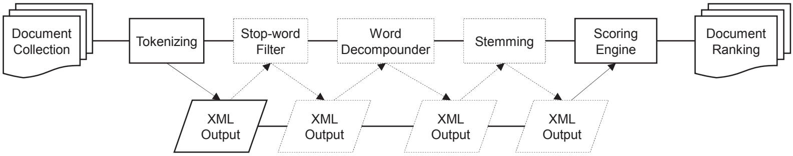


Figure 2.5: Data flow of the Grid@CLEF pilot task for component-level evaluation, redrawn from [62, p. 6].

up to 30 times the size of the original collection. Given the fact that common document collections require several hundreds of gigabytes of storage this may become a challenging aspect for participating institutions that have limited resources at their disposal. Moreover, the task used a linear framework that focused on the indexing process and it contained only up to three intermediate component outputs, i.e. if more components had been considered for evaluation, the data explosion issue (similar to MIREX) would have been much worse.

In fact, out of nine groups which subscribed for participation in the task, only two were able to submit their results [62]. Although the outcome of the task was rather disappointing from the perspective of the organisers, some aspects of the task deserve both attention and acknowledgement. The great potential of exchanging intermediary results from components is that it creates a high degree of transparency. Having different component outputs at hand, it is possible to run a retrospective evaluation of components in new combinations. More importantly, the component outputs could be stored and made available for comparison in future evaluation tasks. However, to be able to do so, the amount of data produced as intermediate output needs to be reduced. A straightforward approach could be to apply compression technologies that are used for incremental backup of digital data or for revision control of documents. Since many components of modern IR systems, which are part of the indexing process, transform a stream of text or other pieces of information, it might be promising to integrate methods for compression of incremental changes into the framework for the generation of intermediary results. This could resolve one of the issues with the *Grid@CLEF* pilot task. The idea of the grid experiments for component-level evalua-

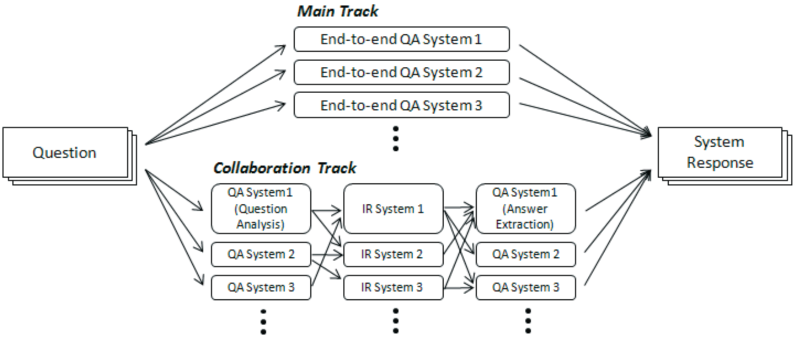


Figure 2.6: Modular architecture for the ALCIA evaluation at NTCIR-7 and NTCIR-8, redrawn from [131, p. 11].

tion for ad-hoc text retrieval will be addressed in Chapter 6. It will be combined with the analysis of the experimental results from the *Grid@CLEF* pilot task. The corresponding findings will be adapted in order to design a large-scale grid-like experiment using the Xtrieval framework in Chapter 7.

The organisers of the *Advanced Cross-lingual Information Access (ACLIA)* task at NTCIR-7 and NTCIR-8 also adapted the method of specifying an XML exchange format to connect common IR modules and Question Answering components. Figure 2.6 illustrates the formal architecture for experiments submitted to the ALCIA tasks. In contrast to *Grid@CLEF* generating and exchanging system output was less complex due to the architecture of the QA task, which operated at a higher level of abstraction.

The ALCIA evaluation experiments demonstrated a high potential of component-level evaluation. It showed that exchanging intermediate results is both feasible and beneficial. In particular, experiments and intermediate results that were submitted to the collaboration track outperformed the configurations that were submitted from a single group. This result suggests that the challenges of component-level evaluation can be mastered, once a focused task architecture and corresponding data structures to exchange intermediate system output have been designed. For the present example of Question Answering systems the task formulation and the definition of system components is straightforward. Moreover, the intermediate output is rather small, because it consists of processed queries and result lists only. The situation might be more complex for other search tasks.

2.6.1.4 Implementing Components as Web Services

Many of the models and frameworks used for IR evaluation that have been discussed so far, mainly focused on a few particular tasks of the underlying problem related to information retrieval, such as text transformation in the *Grid@CLEF* task, or Question Answering in the *ALCIA* experiments. New tasks may appear over time and some of them might be related to a task that ran a number of years ago. Thus, it is desirable to store exact descriptions of tasks, systems and their components and configuration, and evaluation results. The protocols to exchange queries and system outputs for evaluation are defined well. But there is no standard to deliver less formal descriptions of system configurations. An attempt to solve this issue was made by developing the *Service-Oriented Information Retrieval Evaluation (SOIRE)* system [55]. Basically, this is a post-evaluation framework for analysis and report delivery, but it can also be used to manage the archival storage of experiment data. Interfaces to web-based tools were implemented so that every type of document is available as a resource. The framework was successfully used for the evaluation of the CLEF-IP track in 2009 [151]. In the view of the present author, using service-oriented architectures for the organisation and management of evaluation tasks is a key aspect to move beyond traditional TREC-like evaluations.

2.6.1.5 Designing a Framework for Automated Evaluation

Automated component-level evaluation has been proposed to solve key issues of traditional IR evaluation [70]. The basic framework follows the architecture of modern

IR systems and resembles the illustration given in Figure 2.5. But instead of submitting ranked result lists or any type of intermediate output obtained from proprietary systems, participants will be asked to make available any system component via web service protocols. All components should be registered at a central authority (e.g. TREC). Based on this straightforward architecture a large number of experiments can be conducted by alternating the components and configurations used.

Advantages over traditional IR evaluation are: (1) reduced amount of work for participants, (2) less emphasis on the final ranking of systems, (3) identification of best system configurations based on components contributed from different groups. The author of the present work favours the design of the automated component-level evaluation model. This design and the implementation of the protocols for data exchange over the web are crucial for the success of the methodology. But, a number of requirements have to be considered for the design of the protocols. Three essential properties, namely stability, simplicity, and wide applicability, were also presented in [70]. These properties are contradictory and a major limitation in practice. Since motivating participation is one of the key problems of new evaluation models, it was suggested to initiate the evaluation architecture by means of a publicity campaign that emphasises potential benefits. Whether this idea will stimulate enough interest within the IR evaluation community remains an open question.

2.6.2 Challenges for Component-Level Evaluation Architectures

Although the approaches to component-level evaluation presented in the preceding sections have been used for different evaluation scenarios, interesting similarities can be found among them. The most obvious problem of any architecture that relies on distributing data sets is the burden of additional communication cost in exchanging inputs and outputs of intermediate system components. But what are the lessons to be learned from the different approaches to the problem, when we are interested in composing a component-based evaluation task for ad-hoc text retrieval? In fact, all of the presented architectures provide useful ideas for this scenario. But few of them can be considered as successful evaluation exercises in the sense that the component-level comparison yielded additional insights. In the opinion of the present author, the ALCIA tasks at NTCIR are a good example, since they showed that a subsequent re-combination of experiments using components from different institution improved the effectiveness of the results. Such a result is very important in order to motivate participation in similar tasks. What we can learn from MIREX is that a central server architecture needs a lot of computational resources. The less restrictions are imposed on the contributions of programming code, the more human resources are needed to maintain and manage the evaluation cycle. The grid experiments at CLEF taught us, that exchanging intermediate component output may suffer from the data explosion problem. Last but not least, MediaMill demonstrated that an open component-based framework could be used to investigate the overall progress of state-of-the-art system components.

Based on these observations, the most important aspects of an ad-hoc text retrieval evaluation architectures are discussed next. In order to balance the cost trade-off for the communication overhead depicted in Figure 2.5, a web-based architecture may be suited best. The efforts of the community to develop open frameworks like Lemur, Indri, Terrier, and others, allow the assumption that these toolkits reflect the state-of-the-art in ad-hoc IR. In case of a component-based experiment like Gird@CLEF, considering such toolkits would allow the use of their architecture and data structures to cover the respective parts of the IR process. For instance, the IR library Apache Lucene provides a good framework and numerous instances for the text transformation during indexing. Thus, Lucene's index format could serve as protocol for exchanging raw or transformed inverted indices. The resulting amount of data for the indexing process would be limited to the order of magnitude of the number of component configurations and the size of the data collection, i.e. the storage overhead would be predictable and also adhere to the set-up of the component-level evaluation task. It seems obvious that the initial component-level evaluation tasks should be limited to certain parts of an IR system in order to keep the resulting experiments manageable. A similar approach can be formulated for other parts of an IR system, like ranking, feedback, or data fusion.

An ideal architecture for component-level evaluation of IR systems incorporates the most important frameworks according to their strengths in the respective part of research (index compression, text transformation, ranking, data fusion, etc.). In order to automate the evaluation or to dissolve the boundaries of time-consuming annual evaluation cycles, light-weight protocols could be implemented as wrappers for carefully selected existing toolkits. Note that the described steps are only small enhancements of existing technologies, i.e. component-level evaluation is possible as soon as the protocols are generally accepted. One approach to achieve this acceptance is to persuade the community with an initial campaign that demonstrates the added values of the approach.

Another question is how to incorporate the description of system components, which are likely to be contributed from different sources, into the data sets that are archived, to track the progress in the field. Traditional evaluation campaigns already capture most information about submitted experiments. Using the ad-hoc task from CLEF 2009 as an example, such information is the task an experiment was submitted to, and the institution, which designed and conducted the run. Digital object identifiers are assigned to experiments for archival documentation. Further meta-data like object identifiers for the topic set, the language of the topics, and the fields of the topics, i.e. title, description, and narrative, which are used to simulate short, average or long formulations of information needs, are also stored. Finally, Boolean fields are used to indicate whether the queries were created from the topics manually or automatically and whether the experiment was used for the creation of the relevance assessment pool. TREC established a format for storing the final output of evaluation experiments, i.e. a list of 1,000 ranked document candidates for each topic, which is used to collect the results and provide for further statistical analysis of the results or certain aspects of a retrieval task.

Keeping records of each system or experiment configuration would allow to draw more meaningful conclusions from the results of any IR system evaluation. It would also allow researchers to study and analyse relations between experiment configurations and their resulting outcome instead of observing statistical effects on an *unknown* set of experimental results. The lack of a standard that defines which components were considered for an empirical evaluation, does not allow to draw conclusions from the experimental set-up on the results of the evaluation. It is the authors strong belief that capturing only basic information on the experimental set-up with a standard meta-data format would be of great value for the IR community in the future. History has demonstrated that simplistic meta-data standards like Dublin-Core or TV-Anytime are widely accepted. Thus, it might be useful to use an abstract model of an IR system as a starting point to develop a simple but well-defined standard to document IR experiment configurations.

2.7 Summary and Implications

The present chapter discussed the state-of-the-art in information retrieval evaluation. Since one of the main contributions of this work is the Xtrieval framework, which is designed to analyse the state-of-the-art in IR research, the focus was on the most important elements of traditional evaluation. In order to reflect the enhancements of the methodology, the development of IR test collections was presented in chronological order, starting with the pioneer experiments at the Cranfield College of Aeronautics in the late 1950's.

With the design of empirical search experiments in a laboratory setting came the need for metrics to compare the quality of methods for indexing and retrieval. For that reason, the most popular effectiveness measures were also described. As test collections grew in size and more applications included search technology, estimating the significance of empirical evaluation became a challenging issue for IR research. Thus, this chapter included a review of statistical methods that are used in IR research. After the introduction of the most important tools and methods of IR evaluation, a critical view on the results that were obtained in half a century of IR research and evaluation was provided. Most of these critical comments serve as motivation for the present work.

TREC and other evaluation campaigns are organised as annual workshops to inspire collaboration, but participants usually submit results produced by their own proprietary systems. The limitations of these system evaluations were discussed in Section 2.5, especially with respect to the analysis of resulting data and possible conclusions. Modern IR systems are very complex and likely to reflect the specific expertise of an institution or research group. But each IR system consists of various components, like a tokenizer, a stemming algorithm, or a ranking model. All of them contribute to the overall search effectiveness of the system. Given the traditional TREC-like test collection consisting of a set of topics and the ranked output of a system, it is not possible to investigate how any particular component contributed to the overall performance of the system. Nevertheless, it is worth investigating, which instance from a set of alternative components (or component configurations) is optimal for a given IR task. An obvious approach to answer this question is to evaluate the alternatives separately while all other components remain fixed as it was suggested in [144]. However, interactions between alternatives of the component under examination and other parts of the system may also affect retrieval performance. A study on the quality of test collections has shown that interaction effects exist between topics and systems as well as between topics and assessors [19]. Another problem with system evaluation is that it impedes retrospective analysis and long-term reviews of IR tasks. Although evaluation results are made available at the system level, it is hard to conduct retrospective analysis and draw conclusions from experiments, because it requires the re-assembling of the experimental results and corresponding descriptions in the form of technical reports.

Most of the limitations of traditional IR evaluation are obstacles that impede the clarity of the results. Component-level evaluation architectures aim to overcome these limitations. This particular field of IR research is still in its infancy. But it has the potential to create more value for the wide field of applications for search technology. In the following Chapter, the most relevant IR system components are introduced in order to provide insights into the complex architecture of modern IR system implementations.

3 Key Components of IR Systems

The composition and structure of an IR system can be defined in several ways depending on the level of abstraction. In Figure 3.1 [180, p. 4], a high level of abstraction is used to illustrate the core architecture of an IR system as being composed of input, processor, and output. The input to a search engine black box (processor) can either be documents, or queries, and the output contains a number of document references. Although such a definition appears to be trivial, it indicates the most important elements and corresponding interfaces.

Modern IR systems contain numerous components that process documents and answer queries that are formulated in natural language. In order to ensure two major goals of a search application, namely effectiveness (quality) and efficiency (speed), the system

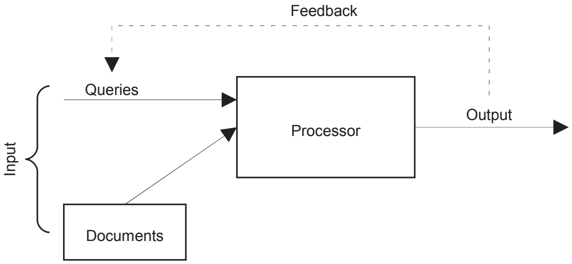


Figure 3.1: Abstract representation of the IR process, redrawn from [180, p. 4].

architecture needs to be designed carefully [45, p. 13]. *Indexing* and *retrieval* are two central processes that any IR system provides. In the definition above, they represent the interfaces between the input and the processor, and between the processor and the output. While the indexing process creates data structures that allow searching, the retrieval process generates a ranked list of documents based on a user request by utilising the index data structures. Both processes are described in more detail in the following subsections.

The main purpose of the present chapter is to describe central processes in current IR systems by means of different state-of-the-art models and corresponding software implementations. The concept of a system component is used to describe the process in an IR system that can be based on different theoretical models. Since the emphasis of this work lies in retrieval effectiveness, the presentation and discussion of IR system components is limited to three major elements: *text pre-processing*, *core retrieval models*, and *relevance feedback*.

Text pre-processing is an essential element in every IR system, because it prepares the vocabulary of tokens on which all of the other components operate. Both the documents and the queries have to be processed using the same chain of text processing algorithms in order to ensure that the implemented core retrieval model will work appropriately. The core retrieval model defines how the documents are matched and ordered in response to a query. This is typically achieved by the application of a ranking function, which is based on term frequency statistics obtained from the documents, the query, and the collection. Relevance feedback is an optional component in IR systems, which is applied to improve the quality of the search result. In the following sections, these key components will be discussed in detail. The focus is on widely used state-of-the-art models and algorithms and the selection is restricted to components that will be used for the empirical analysis in Chapter 7.

3.1 Text Pre-Processing

According to the definition given in [45, p. 14] the indexing process of search engines can be divided into the major components *text acquisition*, *text transformation*, and *index creation*. The purpose of a text acquisition component is to provide access to the documents that will be searched. Documents are provided as static collections in traditional IR evaluation scenarios superseding the acquisition step. In practice almost any resource of information is dynamically changing over time and therefore requires a component that collects and stores document data accordingly. These sources of information are typically organised in specific ways to facilitate user access, e.g. documents from the worldwide web or corporate networks are specified using uniform resource identifiers. In the scenario of web search a crawler component identifies and collects documents to be made accessible by the search engine. Since not only web documents are heterogeneous in content, structure, and format, a major task of text acquisition is to preserve as much of this additional information as possible. Because of large amounts of documents such meta-data needs to be both carefully selected and stored efficiently.

A major challenge when collecting documents and passing their contents to text transformation components is the variety of digital text formats like HTML, XML, PDF, and several other proprietary formats used in common office software packages. The latter typically includes control sequences and content compression. Thus, components that allow the conversion of these formats into plain text are required.

Another problem in this particular stage of the indexing process originates from the diversity of character encoding schemes. The Latin alphabet, which most European languages are based on, can be encoded using eight bits. Common encoding schemes are ASCII, ISO-8859-1, or the UTF-8 format. All documents have to be encoded using a single common encoding scheme to ensure consistent preparation of text streams for following transformation procedures.

A text processing component in IR transforms streams of text coming from documents. In order to keep the size of a resulting index as small as possible the documents are reduced to a number of *index terms*. Index terms are selected to reflect the content of the corresponding document and therefore ensure the quality of search results. Obviously, a more restrictive selection strategy leads to poorer quality of the document representation in an index and is likely to degrade retrieval performance. How the streams of text coming from acquisition components are processed in order to prepare the data for index creation will be discussed next. *Tokenising*, *stopping*, and *stemming* are the most common concepts for this purpose, although they are by no means all. The presentation of these steps is based on the typical sequence of the three steps in an IR system.

3.1.1 Tokenising

For text transformation, documents are prepared as streams of alphanumerical characters. Tokenising describes the process of splitting the character stream into tokens, i.e. meaningful groups of characters like words. A simple form of a Tokenizer for languages using the Latin alphabet could split a stream at occurring white space characters. Given this example, tokenising may appear to be a very straightforward procedure. This is in fact not true due to the variety of word forms or special characters. Variations like capital letters, hyphens, or apostrophes, potentially affect the performance of a search engine. In English, for example, the words *apple* and *Apple* have different meanings and using a Tokenizer that transforms any capital letter into a lower-case letter would prevent the system being able to differentiate between the two. In a similar scenario, a Tokenizer that removes apostrophes without any distinction between cases like *O'Hare International Airport* and *in a user's point of view*, will likely degrade the system performance. Since large proportions of languages are static, typical Tokenizers are rule-based algorithms that handle most of these special cases. In order to match query and index terms for retrieval purposes, submitted user requests and documents have to be transformed into tokens in identical manner. In contrast to tokenising all subsequent text transformations operate on word level.

3.1.2 Stop-word Removal

Most words describe or contribute to the description of particular concepts in documents. There are, however, other types of words that do not contribute to the description of concepts. Thus, it is important to identify the latter type in order to exclude them from the index vocabulary. Stopping, or stop-word removal, is the task of eliminating those words from an incoming stream of tokens or text. Typical stop-words are function words that contribute to form the syntactical structure of sentences. Particle words like “the”, “of”, “or”, or “to” belong to this category. Stop-word removal has usually no negative impact on the effectiveness of the IR system, in fact, there is a lot of empirical evidence from TREC experiments showing that stop-word filtering improves retrieval effectiveness.

Since stop-words are most likely function words, they depend on the morphology of the language. Invariant stop-words like particles in English can be easily identified, but for example in German there are cases of inflective stop-words like “ein”, “eine”, “einer”, “einen”, and “einem”. Two straightforward ways exist that deal with these inflections. First, all inflections can simply be included in the stop-word list. Or second, they can be grouped by a stemming algorithm in a first step and subsequently filtered by removing the group’s representative.

In spite of the potential advantages of stopping, a major limitation of the technique is processing phrases, especially those that wholly consist of stop-words. Shakespeare’s quote “to be or not to be” is a prominent example. Although a list of such phrases will be small in comparison to all possible requests, a search engine should be designed to avoid such potential disadvantages. An approach to the problem could be to use a smaller stop-word list during indexing (e.g. excluding “not” and “be”) and an adaptive stopping strategy (in terms of reducing the number of words in the request) when processing queries. Another idea is to introduce a rule that does not allow two or more subsequent words to be filtered by stopping. A remaining question is how many words should be included in a stop-word list. Although typical stop-word lists contain hundreds of words, there is no general limit for the number of stop-words to be removed.

3.1.3 Stemming

The motivation behind using a stemming component in IR systems is to increase the likelihood that words from queries and documents will match. Consequently, a stemmer processes streams of text on the level of words. As mentioned before, documents are considered to be about concepts, which are described by words. But a single word can cover a number of different concepts and a single concept can be expressed using different words. A *homonym* like “bench” may refer to a bench to sit on, or a raised portion of ground in a river, or a desktop. And the concept “money” may be expressed using *synonyms* like cash, currency, or capital.

A *stem* is a common derivation, which is shared by a group of words. Sometimes the common stem of a word is also termed *root*, although a stem may contain prefixes and the root form of a word does not. However, such definitions are usually not needed for the purpose of stemming in IR, because common algorithms only identify and remove suffixes. Three basic groups of suffixes can be distinguished for stemming, namely *attached suffixes*, *inflective suffixes*, and *derivational suffixes*.

An attached suffix is a particle word that is appended to another word. They are common in some Romance languages like Italian, Spanish, or Portuguese, where personal pronouns are attached to specific verbs. Inflectional suffixes represent forms of the grammar of languages. Thus, they can be applied to all words of a particular grammatical type. Some irregularities may nonetheless exist. Common examples are an attached “-s” for the plural of nouns or the regular past tense extension “-ed” for verbs in English.

Modifications of the stem, like an extra “p” in “mapped” or a dropped “e” in “solved”, can be problematic for suffix stripping algorithms. A single inflective suffix can actually represent different types of transformations. In English, for example, the suffix “s” could be a noun ending that indicates plural, a noun ending that designates possession, or even the ending of a verb that is used in third person singular. The last class of derivational suffixes creates new words that often fall into a different grammatical category. They may even change the meaning of the word. Derivational suffixes cannot be attached by general rules of grammar. In English the suffix “-ness” transforms an adjective to a noun with corresponding meaning, e.g. happy can be converted to happiness or lazy can be reformed to laziness. Another suffix “-ly” can either alter an adjective to build a corresponding adverb (e.g. glad - gladly) or transform a noun into an adjective (e.g. master - masterly). The usual order of the presented classes of suffixes in a word is derivational, inflective, and attached. Thus, they could be removed algorithmically from the right to the left (assuming words are written from left to right). A typical rule-based stemmer will try to remove all attached suffixes first, all inflective suffixes subsequently, and lastly those derivational suffixes that are readily identifiable. The process of reducing at least two words to a common stem is termed *conflation*.

3.1.3.1 Stemming Errors

Since a stemming component conflates synonyms or variations of a common root form it may introduce homonyms that are in fact unrelated. Depending on how aggressive the stemming algorithm is, it may either create too many new (or wrong) homonyms or it may not identify all synonym forms. The former is termed *over-stemming* and the latter is called *under-stemming*.

Under-stemming leaves a number of synonyms, wasting the potential to reduce the index size. In contrast to that, over-stemming may degrade the performance of retrieval systems, because different concepts are likely to be mapped as being one. A further subtle distinction can be made between over-stemming and mis-stemming. The latter happens in cases where some part of a word is removed because it looks like an ending although in fact it is not. The former results from conflating two words with different meanings by removing true suffixes.

To overcome mis-stemming and over-stemming errors, a dictionary can be employed to identify different meanings of words that would be conflated otherwise. However, even an exhaustive dictionary is not an optimal solution to stemming errors, because it needs to be maintained over time in order to reflect the changes in the contemporary use of the language.

3.1.3.2 Common Stemmers for English

One of the first published stemming algorithms for English was the Lovins stemmer [119]. It was developed for use in a library information system. The algorithm combines basic reduction rules with a context-sensitive recoding procedure to account for modifications of the stem, e.g. doubled consonants in past tense transformation of regular verbs.

The most widely-used stemmer for the English language is the Porter stemmer [140]. It treats complex suffixes as being compounds of simple suffixes and subsequently removes those simple suffixes depending on the form of the remaining stem. Martin Porter was the first to note that only a number of carefully designed rules are needed to develop a suffix stripping algorithm for use in an IR system. And, more importantly, he stated that from a certain point during the development of a stemming program, adding more rules to improve performance in one part of the vocabulary will inevitably degrade performance elsewhere.

Although it is important to include rules to overcome irregularities, these rules could be regarded as practically irrelevant based on their frequency in language. The particular focus on practical relevance of the applied rules was, and still is, the key to the success of the algorithm. Porter evaluated his algorithm on a dictionary of 10,000 words and achieved a reduction of roughly one third in terms of index storage size. Compared to a more elaborate stemmer [48], implemented by Dawson, using a standard IR test collection, the Porter stemmer performed slightly better in terms of retrieval effectiveness.

The purpose of another stemming program [101], named Krovetz stemmer, was to improve the performance of the Porter stemmer by adding a dictionary lookup to avoid stemming errors. Checking the output of every step of the Porter stemmer against a dictionary turned out to be a dead end, because words that were previously correctly conflated, produced different stems. These errors degraded retrieval performance and a completely new algorithm was designed to make use of a dictionary. In addition to that, the implementation aimed to detect word senses in order to disambiguate words that should not be conflated. It was shown empirically in [101] that this approach to stemming improves retrieval performance, especially for short documents.

Another implementation of a stemming program [137], called Paice stemmer, was motivated by the observation that stemmers were usually evaluated in terms of IR system performance, but not in terms of the rules they apply. To optimise the rules underlying the stemmers by Lovins, Porter, and Dawson, it was suggested that a single, but ordered table of rules, and an iterative process that removes single letters, be used. The rules were organised in sections that affect words with identical ending letters to allow fast lookup and processing. In each step of the algorithm the rules that match the final letter of a word were applied and a corresponding action (retaining or removing) was taken. The algorithm iteratively continues, unless there is no rule found for a current letter. Based on an empirical evaluation [137] the stemmer was found to produce a higher number of over-stemming errors than the Porter stemmer.

Another group of stemmers uses statistical modelling to overcome the need for linguistic knowledge and derived rules. Thus, statistical stemmers are applicable for all alphabetic languages. The most common implementation is character *n-gram* stemming. In fact forming n-grams from words is rather a tokenising process than a stemming procedure. Since its purpose is to overcome some of the limitations of stemmers, it is usually compared to stemming algorithms and also perceived as being one of them. The basic idea behind the approach is to shift a character window of size n over the words resulting in a number of pseudo-stems with fixed length n. There are, however, different approaches to process the stream. The window technique could be applied by either including or omitting preceding and adjacent blank spaces. Another alternative is to completely ignore the word-level. An advantage of this technique is that it captures some information about adjacent words. Since it is a very simple form of word co-occurrence it might help to improve retrieval performance, especially for phrasal queries.

The major drawback of the family of character n-gram approaches is its negative impact on the efficiency of the IR system. N-gram stemming significantly increases the size of the index, because of the redundancy introduced by the windowing technique. Since there is a positive correlation between index size and query response time it also degrades system performance at query time.

An even larger impact on the response time is that queries have to be transformed in the same way as the documents, which results in long queries. A comprehensive empirical study on different n-gram stemming techniques across 18 languages showed that 4-gram and 5-gram stemming and ignoring the word level can significantly improve retrieval performance on average [128]. But the authors also provided a statistic based on the English test collection they used that demonstrated the impact on index size and average query response time. Compared to indexing without any stemming algorithm the index size was three to five times larger. They reported average query processing times that were seven times larger than the baseline.

Motivated by the observation that too many, or too strict, rules can result in overstemming and consequently degrade IR performance, a number of (less aggressive) *light stemmers* were proposed. Light stemmers focus on removing inflective suffixes only, because it makes them much less error-prone. A very simple variant for English is the straightforward implementation of an *s-stemmer*. Its only purpose is to conflate plural to singular nouns, third person singular of verbs to their basic form, and to remove possessive apostrophes.

Another simple implementation could remove the common ending of regular verbs in the past tense, although there are a number of exceptions, like doubled or deleted consonants, that need to be handled by a few extra rules. However, for languages other than English, especially those that have a more complicated morphology, considerably more effort is needed even for light stemming methods.

A collection of light stemmers for various European languages other than English is described in [162]. The intention of the design of these light stemmers was to focus on fewer but more frequent morphological rules. Thus, they only handled gender and plural for adjectives, and for nouns, they added some rules to conflate grammatical cases. This study supplied empirical evidence that light stemming can produce statistically significant improvements over conventional stemming algorithms. A similar conservative stemmer was also developed for English [88]. In the empirical evaluation discussed in Chapter 7, it is used as the prototype of a light stemmer.

3.1.4 Information Extraction and Analysis

Despite the previous text transformation operations, extracting syntactical information about words can be more complex. The purpose of information extraction is to collect additional input about a single word or a passage of text that might be relevant for search. In text mark-up languages like HTML such additional information can be easily extracted from highlighted text in various forms, like italics or headings.

More complex information extraction components typically require a prior tagging of *part-of-speech* (POS tagging). The models which are used to recognise the syntactical structure of sentences or paragraphs are computationally expensive. But they can be used to extract different forms of phrases, like noun or verbal phrases, that may help to improve retrieval effectiveness.

POS tagging can serve as a basis for the extraction of semantic text features. These components are called *named-entity recognisers*. Current state-of-the-art approaches can reliably detect entities like proper names of persons, places, and companies as well as variants of dates. Being able to collect such contextual information could help to disambiguate search results on very large collections such as the web.

In the scenario of web search, various types of contextual information are readily available. Document parsers can extract links and corresponding anchor text to be processed during indexing or search. Both features are extensively used in web search engines, because there is substantial empirical evidence that they can significantly improve retrieval effectiveness for specific types of search queries. Specific analysis algorithms are needed to be able to exploit the linked structure of resources like the web. The PageRank algorithm [25] is the most famous representative for link analysis and it provides a relative rating for the importance of a web page. Link anchor text is a source of context information, which is used to enrich the content of a web page it points to. In a dynamic environment like the web such algorithms are exposed to potential abuse. Hence, search engine companies try to keep the core of these algorithms a secret.

Employing content classification components is another common approach to gather additional information. The purpose of classification algorithms is to assign class labels to documents. There are various applications for the classification for indexing, retrieval and result presentation. An obvious example is to use topical labels like “sports”, “politics”, or “economy” to categorise documents. But more importantly, classification can be used as a selective strategy to decide which documents, or which parts of documents, should be indexed. Classification algorithms are important tools to filter spam and non-content portions in documents for many search applications.

3.1.5 Creating Index Structures

One central concept to create index data structures in any search engine is *inversion*. The purpose of inversion is to convert a document-term stream coming from text transformation components into term-document information in order to create inverted index files. The actual structure of an index partially depends on the ranking models in the IR system. Fast processing of queries is the main purpose of index inversion. Both documents and terms are usually represented by identifiers to make the index construction more efficient. For each term in the collection all document identifiers are stored in a list. The term-document pairs are sorted by term identifiers and for each term all document identifiers are organised into a postings list.

A common method of efficiently indexing large document collections is the *blocked sort-based* indexing algorithm [126, p. 71]. It is designed to overcome the problem that posting lists for large collections can be multiple times larger than the physical main memory available for indexing. The basic idea is to split the collection into equal parts, create and sort the postings list for each part in the main memory, store each sorted posting list to a hard disk, and merge all intermediate results into the final inverted index. The time complexity of the algorithm is $\Omega(n\, log\, n)$ for sorting n postings lists. In addition to that, a data structure is needed to store the mapping of terms to term identifiers. Due to memory limitation this could be a potential efficacy problem for very large collections.

A more scalable alternative implementation is termed *single-pass in-memory indexing* [126, p 73f.]. This algorithm relies on terms and stores a dictionary of terms for each block. As a consequence, the only limitation for this indexing approach is the available storage capacity. The single-pass in-memory implementation processes the incoming text stream per token. When a new term occurs, it is added to the block dictionary and a new posting list is created. Instead of collecting and sorting a complete list of pairs of term-document identifiers the algorithm directly inserts every posting (document) in its corresponding postings list. This results in two major advantages over the blocked sort-based approach. Firstly, no sorting of posting lists is required. And secondly, since the terms are represented as the key of each postings list, the data structure to map terms and their identifiers can be omitted to save memory consumption. When no more memory is available to process further tokens the index file for the current block is written to disk. This index file simply consists of an alphabetically sorted term dictionary and corresponding postings lists. Sorting the blocks before writing them to disk ensures that the final merging of the blocks can be realised with linear scans. Since expensive sorting of term-document identifiers is neglected and all other operations will run in linear time at most, the complexity of the single-pass in-memory indexing algorithm is $\Omega(n)$, where n is the size of the collection.

3.1.5.1 Term Weighting

Term weights are usually derived from empirical observation and hence investigations on statistics of text have a long history in research. The two algorithms presented above order postings lists with respect to document identifiers. Although this method has advantages in terms of space requirements when compression is applied, it is not optimal for building retrieval systems that produce ranked output. In typical rank-based retrieval systems, postings are sorted by their impact or weight, which is usually a score computed from statistical information. This allows the early termination of the scanning of long postings. When the weight of a posting in a list is considered to be too small to be predicted as being similar to a request, any adjacent postings can be skipped. Using an index that is sorted according to posting scores, requires considerably more effort for updating the index when new documents are added. This limitation is due to the fact that the postings for that new document have to be inserted at particular positions in affected posting lists. Weights are usually collected for terms to reflect their relative importance. The exact form for the calculation of term weights depends on the retrieval model and its ranking function. A desirable feature of an efficient search engine is that most weights that are required for ranking are computed during indexing. However, this efficiency is traded for flexibility, because once the index is created with particular weights, a restriction to the corresponding ranking model is made.

Already in the late 1950s it was proposed that the significance of a word could be considered as a function of its frequency in a document [120]. Statistical observations on text distribution reveal that the distribution of term frequencies in document collections are not uniform, but skewed. That means that a few words occur very often and many other words appear only a very few times. In English the two most frequent words are "the" and "of" in regular document collections. Together they account for roughly ten percent of all word occurrences. Given a sufficiently large collection another general observation is that about half of the unique terms appear only once. This distribution is known as Zipf´s law and it states that the frequency of a term is inversely proportional to its rank in the ordered set of term frequencies.

An alternate formulation is motivated by the probability P_t for the occurrence of a term. P_t can be obtained by dividing its frequency by the total number of terms in a collection. Given the rank r_t for a term, Zipf´s law can be reformulated in the sense that P_t multiplied by r_t is a constant number c. For English text collection c is approximately 0.1. In general it can be observed that Zipf´s law is inaccurate for terms on both tails of the frequency ranking.

Another interesting statistical property of text collections is the size of the vocabulary. In environments where the document collection grows over time and new documents have to be indexed, it is useful to be able to predict the growth of the vocabulary. By means of an empirical analysis a simple model was established to describe the relationship between the size of a document collection and the size of the vocabulary [81, p. 206f.]. The model is sometimes called Heaps' law and predicts that new terms will be introduced very frequently when the size of a corpus is small. For a larger, or dynamically growing collections, the number of new words will increase indefinitely, but at a considerably slower rate. For this reason it is a common approach to use term statistics to estimate the size of potential result sets and of (unknown) collections.

3.1.5.2 Collection Statistics

In order to ensure the effectiveness of the IR system several statistics are computed and stored during the indexing process. Statistical information about terms, documents, and other features is used by the ranking functions to compute the document scores during the retrieval process (see Section 3.2). A few figures are generally required for ranking. Most common numbers include the counts of index term occurrences in individual documents, which is usually called *document frequency*, the positions of tokens or terms within a document, the counts of documents over particular groups of documents, which might be the entire document collection or just a topical category, and summary numbers like the total amount of terms, the size of the vocabulary, or the length of a document in terms of the number of tokens. In fact, the actual figures that are used depend on the algorithm used for ranking and its underlying retrieval model.

3.2 Core Retrieval Models

This section describes theoretical models for the ranking of documents used in IR systems. The following presentation is based on standard books [45, 84, 126] and articles [147, 148, 157] on the subject and organised chronologically in order to illustrate the theoretical relationships between the proposed models. Note that these standard works are being referenced throughout this section (see the citation marks at the end of each paragraph).

The point of origin for this description are classic ranking models, because they serve as baselines for the empirical comparison (see Section 7) with more complex models, which were developed recently. In order to provide meaningful insights into the effectiveness of different ranking theories, the language modelling technique, the divergence from randomness framework, and a few recent models that enhance these models further will be considered here.

Since the late 1960s one of the primary goals in information retrieval research has been to promote the formalisation of the processes that are involved when a person makes a decision that a certain piece of text is relevant to his information need. During that time a number of theoretical postulates were proposed in the form of mathematical models in order to simulate the complex concept of relevance. A selection of the most influential retrieval models is presented in the following. Some of the older models are described first. They were exceeded in terms of retrieval effectiveness by more recent models that will be discussed subsequently.

3.2.1 Exact Matching

One of the earliest models for information retrieval was the *Boolean retrieval model*. It belongs to the group of exact matching models, where documents are only retrieved if they precisely match a given query specification. There are two basic assumptions for the Boolean model. Relevance is binary and all documents in a retrieved set are equal with respect to relevance. The latter represents the main limitation of the model, because a request may return a large number of documents and the resulting presentation is likely to be arbitrary. Without any notion of ranking, such results could be frustrating for users.

The model inherited its name from the mathematical logic by Boole, which is used for query formulation. In an iterative usage scenario Boolean operators enable the user to directly manipulate the size of the retrieved set. The size of a result set can be reduced by introducing new query terms and concatenating them using the AND operator. In contrast to that, using the OR operator to add further query terms will produce a larger result set, which represents the aggregate of the result sets for each of the query terms. Such a system behaviour has the great advantage that it is absolutely clear why a document was retrieved. As a result, an expert user will obtain reasonable control over the system and its output. Another advantage is that the queries are not restricted to words. In fact it is possible to formulate query specifications that mix terms with other types of information, like dates or document types. Since Boolean logic is used for query formulation, an efficient software implementation of the model can be considerably faster than other models. Despite these positive aspects, the main limitation of the model is that it shifts the problem of creating a good result set to the user. In response to that limitation, actual Boolean search systems were typically operated by expert users, which were called search intermediaries. They translated user information needs into complex Boolean query specifications and returned their final result set to the user. Based on the statistics given in Section 2.2 it becomes obvious that the Boolean model would require much more effort to formulate an appropriate query on present document collections. They are several orders of magnitude larger than collections in the era of the Boolean retrieval model.

Nevertheless, a few extensions to the Boolean model were developed during the 1990s with the group of *region models* being one of them. Region models treat a document collection as a string of tokens. A region can be any sequence of consecutive words that is identified by markers of a start and an end position. The main advantage of the approach is that it is not restricted to the retrieval of complete documents. Instead paragraphs or other parts of structured documents can be retrieved. Region models operate on sets of regions and define at most two more operators in their query language: CONTAINED_BY and CONTAINING. The model proposed in [32] explicitly separates mark-up from content to allow a straightforward formulation of more complex queries. Other models of this category do not distinguish between content and non-content. As a result, the query formulation process can be more verbose. Region models share the main limitation of Boolean retrieval models, which is unranked output.

3.2.2 Vector Space Model

In the 1960s the *vector space approach* to information retrieval became one of the most dominant retrieval models for more than two decades. It is motivated by empirical observation and the analysis of statistical properties of text documents. The most appealing advantage of the model is that it provides an intuitive framework for the implementation of term weighting, ranking, and relevance feedback. A major drawback is that there is only little theoretical explanation on how term weighting and ranking relate to the concept of relevance. In [120], a first approach based on term statistics was proposed as follows. In order to find documents that are relevant to a certain information need, a query should be formulated as a document that is similar to the desired documents. The degree of similarity between the "query document" representation and the documents in the collection could then be used to rank the results. But how to compare the representation with the actual documents? A straightforward idea is to count the amount of shared terms, i.e. the number of terms that appear in both the query representation and the document in the collection. For this purpose the documents can be represented as being a vector $d = (t_1, t_2, ...t_n)$ of an index term vocabulary. A query vector $q = (t_1, t_2, ...t_n)$ could be constructed using the same vocabulary of terms. Using both representations a straightforward ranking function $r(\vec{d}, \vec{q})$ could be formulated as given in Equation 3.1. When each vector consists of binary values, this vector product measures the number of terms that $\vec{d}$ and $\vec{q}$ have in common. Possible advancements of the function use natural or real numbers to account for the distribution of the index terms. A few of the most widely adapted enhancements are presented in the following.

$$r(\vec{d}, \vec{q}) = \sum_{i=1}^{n} d_i \cdot q_i \tag{3.1}$$

A stronger theoretical foundation for the vector space model was proposed in [156]. This enhancement represents the documents and the queries in a high-dimensional Euclidean space, in which each of the index terms is assigned a separate dimension. Given this spatial definition, the similarity of document and queries can be expressed as angles between the vectors $\vec{d}$ and $\vec{q}$. The most common measure is the cosine function, which is zero if two vectors are orthogonal and one if they are equal. Using the cosine angular similarity as ranking function $r_{cosine}(\vec{d}, \vec{q})$ in the vector space model results in the definition given in Equation 3.2.

$$r_{cosine}(\vec{d}, \vec{q}) = \frac{\sum_{i=1}^{n} d_i \cdot q_i}{\sqrt{\sum_{i=1}^{n} (d_i)^2} \cdot \sqrt{\sum_{i=1}^{n} (q_i)^2}} \tag{3.2}$$

An appealing aspect of the model is its simple representation of document similarity as angles of vectors in Euclidean space. Using a simplified three-dimensional figure, the basic features of the approach can be illustrated in a simple way. A geometric interpretation can be used to explain the angular similarity measures. Besides the cosine correlation other similarity measures can be computed in Euclidean space. Common measures are the Dice coefficient and the Jaccard coefficient. In empirical evaluation the cosine correlation was shown to be superior to other measures and has hence become the de facto standard to measure similarity in the vector space model.

A minor limitation of the vector space model lies in its implementation. For the calculation of the ranking all components of the corresponding vectors $\vec{q}$ and $\vec{d}$ are needed. These values are not necessarily available in an inverted index. Thus, the normalised weights have to be stored in the inverted index. Alternatively, they could be collected in a separate data structure. However, storing these values at indexing time is problematic when incremental updates of the index are needed. When a new document is added to the index, all document frequencies of terms that are contained in the new document change and have to be updated. Some implementations have addressed this issue and showed that it is possible to efficiently handle it.

3.2.3 Vector Space and TF-IDF

In the present discussion the problem of term weighting was covered several times. Term weighting schemes define how the values for the vectors are calculated. Like already mentioned above, the general framework of the vector space model does not provide such a definition. Nevertheless, several term weighting schemes have been proposed over time. The most widely-used variant is the family of *tf.idf* schemes, which represent a combination of a term‘s occurrence frequency *tf* and its inverse document frequency *idf*. The latter is inversely related to the document frequency, i.e. the number of documents in a collection that contain the term. As a result, it reflects the importance of a term across an entire collection. In contrast to that, the term frequency indicates the importance of a term within a particular document. The original definition of weighting terms using *tf.idf* was proposed in [157]. Equation 3.3 is a reproduction of this formula, where $tf(t,d)$ represents the number of occurrences of term t in document d, $df(t)$ reflects the number of documents containing term t, n is the total number of documents in a collection, and w_t is the weight for term t.

$$w_t = tf(t,d) \cdot \log \frac{n}{d(t)} \tag{3.3}$$

Variants of this *tf.idf* weighting scheme are also used in modern IR systems. A change of the original weighting was motivated by the fact that it did not perform very well in experimental evaluations. Although normalisation was part of the cosine correlation measure it was found that the term weights appeared to be skewed for long documents. In these documents many terms may appear only once, but a few terms occur more often. As a result, they dominated the resulting document vector. To account for that, it was suggested to substitute raw counts of the term frequency $tf(t, d)$ with a normalisation using the logarithm. The empirical experiments presented in [157] demonstrated that this particular change can improve the effectiveness of the vector space approach.

3.2.4 Probabilistic Models

The task of retrieving information is a complex activity and several approaches attempt to formalise this process. One of these attempts applies probability theory. In information retrieval the most dominant concept is relevance and a formal definition of the probability of relevance $P(R)$ is central for all probabilistic models. Many different approaches have been proposed over the years and as a consequence the probabilistic retrieval model has become the most dominant paradigm in information retrieval today. All probabilistic ranking models are founded on the *probability ranking principle* (PRP). It was formulated as follows [147, p. 281]:

> “If a reference retrieval system‘s response to each request is a ranking of the documents in the collections in order of decreasing probability of usefulness to the user who submitted the request, where the probabilities are estimated as accurately as possible on the basis of whatever data has been made available to the system for this purpose, then the overall effectiveness of the system to its users will be the best that is obtainable on the basis of that data.”

Based on a few assumptions about documents and the concept of relevance, it can be shown that the PRP is valid. A possible assumption is to treat the relevance of a document to a query as being independent of any other document. Clearly, ranking the documents by decreasing probability of relevance will result in optimal precision at any rank position. But the PRP does not provide any definition of how these probabilities can be estimated. All of the numerous probabilistic ranking models that are available today are based on (somewhat different) assumptions about relevance and the relation between documents and terms. As a consequence they define different ways to calculate the probability of relevance $P(R)$. The remainder of this section provides an overview on the most widely used probabilistic ranking models and their mathematical foundations. The first of the presented models is the *binary independence model* [148]. It is rather simple, but serves as a basis for the most successful model of this particular group. The model is still in use in many applications that require binary classification.

3.2.4.1 Binary Independence Model

As the name suggests the binary relevance model is based on the assumptions that relevance is binary, i.e. a document is either relevant or not relevant, and that the decision about relevance for one document is independent of other documents. Another important requirement is that information about the relevant set is available. Then, the purpose of the retrieval function becomes the calculation of a probability of a document being relevant, which can be formulated as conditional probability $P(R|D)$ given a representation of document D, and to estimate the analogue conditional probability of non-relevance $P(NR|D)$. Given both probabilities a document can be classified as being relevant if Inequality 3.4 is fulfilled.

$$P(R|D) > P(NR|D) \tag{3.4}$$

A system that relies on these binary decisions about document relevance is called a *Bayes‘ classifier*. The question is how the two probabilities can be estimated. Given the assumption that information about the relevant set is present, one could start by calculating the conditional probability $P(D|R)$. In the next step it is possible to apply

Bayes‘ rule of conditional probabilities to obtain a value for $P(R|D)$ using $P(D|R)$ as defined in Equation 3.5.

$$P(R|D) = \frac{P(D|R) \cdot P(R)}{P(D)} \tag{3.5}$$

However, estimates for $P(R)$ and $P(D)$ are needed in this definition. $P(R)$ is simply an (equal) a priori probability of relevance that can be obtained from the (known) distribution of relevance. $P(D)$ is a constant, since it describes the probability of a document being drawn from the collection. Thus, Inequality 3.4 can be reformulated as defined in Inequality 3.6. Simple mathematical conversion results in Inequality 3.7.

$$P(D|R) \cdot P(R) > P(D|NR) \cdot P(NR) \tag{3.6}$$

$$\frac{P(D|R)}{P(D|NR)} > \frac{P(NR)}{P(R)} \tag{3.7}$$

A few more transformations are needed to obtain the document scores. Therefore it is assumed that documents are a combination of terms and that the set of relevant, and the set of non-relevant documents, are represented by term probabilities. Thus, a document $D = (d_1, d_2, ...d_n)$ can be represented as a vector of binary features of terms being either present or not. Under the assumption that all terms occur independently, the probabilities of the terms from document D occurring in the relevant set can be multiplied to obtain $P(D|R)$. However, terms do not appear independently in regular text collections. For example, if the word “Eiffel” occurs in a document, there is a high probability that the document will also contain the word “tower”. Nevertheless, the assumption is used to simplify the mathematical representation. Using the assumption of term independence and the vector representation of documents, the probabilities $P(D|R)$ and $P(D|NR)$ can be calculated as the product of individual term probabilities as presented in Equation 3.8. [45, p. 250]

$$\frac{P(D|R)}{P(D|NR)} = \prod_{i=1}^{n} \frac{P(d_i|R)}{P(d_i|NR)} \tag{3.8}$$

In order to calculate these probabilities for documents, the probabilities for the terms are needed. Since terms are represented as binary features, the right term of Equation 3.8 can be separated into two parts for every document. The resulting definition is given in Equation 3.9, where $i : d_i = 1$ denotes terms that exist in the document and $i : d_i = 0$ represents all other terms.

$$\prod_{i:d_i=1} \frac{P(d_i = 1|R)}{P(d_i = 1|NR)} \cdot \prod_{i:d_i=0} \frac{P(d_i = 0|R)}{P(d_i = 0|NR)} \tag{3.9}$$

Since a known relation exists both for the probabilities in the numerators and for the denominators of Equation 3.9, the expression can be reformulated further. Let u_i represent the probability of a term i that occurs in a document from the relevant set, i.e. the probability $P(d_i = 1|R)$ in Equation 3.9. Given this assumption, the corresponding probability for a term i that does *not* occur in a document from the relevant set will be $1 - u_i$. Using this principle for terms from documents that are not part of the relevant set, the probabilities in the denominators of Equation 3.9 can be rewritten as v_i and $1 - v_i$. The resulting formulation is as follows:

$$\prod_{i:d_i=1} \frac{u_i}{v_i} \cdot \prod_{i:d_i=0} \frac{1 - u_i}{1 - v_i} \tag{3.10}$$

The product in the right term of the definition contains only terms that are not present in a document, but in this present form it depends on a document. If it is possible to transform the term mathematically, such that all terms are part of the product, without changing the factors themselves, the term could be omitted, because it is equal for all documents. Consequently, omitting it will not change the total probability of relevance for a document:

$$\prod_{i:d_i=1} \frac{u_i}{v_i} \cdot \Big(\prod_{i:d_i=1} \frac{1-v_i}{1-u_i} \cdot \prod_{i:d_i=1} \frac{1-u_i}{1-v_i} \Big) \cdot \prod_{i:d_i=0} \frac{1-u_i}{1-v_i}$$

$$\prod_{i:d_i=1} \frac{u_i(1-v_i)}{v_i(1-u_i)} \cdot \prod_{i} \frac{1-u_i}{1-v_i} \tag{3.11}$$

A further mathematical transformation is applied to Equation 3.11 (without the right hand product) in order to ensure efficient implementation in practice. Due to large amounts of documents in collections and many unique index terms, the probabilities u_i and v_i for terms occurring in a document will be considerably small. Computation of products with many small factors can cause severe problems in accuracy. Since the logarithm is a monotonic function, and the order of the probability of relevance for the document is preserved for this reason, the product of the probabilities can be replaced by using the logarithm as given in Definition 3.12.

$$\sum_{i:d_i=1} \log\Big(\frac{u_i(1-v_i)}{v_i(1-u_i)}\Big) \tag{3.12}$$

The resulting figure that can be estimated using Definition 3.12 is called *retrieval status value*. In the present definition no notion of the query exists. It is, however, based on the terms of the document of a collection. Since the query is the only source of information in almost every case, it can be assumed that terms, which are not part of the query, will be equally likely to occur both in relevant, and non-relevant documents. Thus, the sum given in Definition 3.12 can be reduced to those terms of a document that match the query terms. [45, p. 251] This fact reveals a limitation of the binary independence model: it only distinguishes between query terms that are present or absent in a specific document. As a consequence long queries are needed, because the number of distinct scores for documents would be limited otherwise. Consider short queries that contain only one, two, or three terms. For example, given a single-word query there will be only two (four, and eight, respectively for two and three terms) different retrieval status values, because that particular term will either be present in the documents or not. For this reason, the binary independence model should not be applied in web search, where queries are usually short. [84, p. 10]

Due to the high cost of acquiring large amounts of relevance judgements, it can be assumed that relevance information is rare. Given this fact, one can try to further simplify the estimation of the probabilities in Definition 3.12. The probability u_i, that a specific term occurs in a document from the relevant set, can be assumed to be a constant. Thus, using the binary model of independence u_i can be set to 0.5. Given the knowledge inferred from empirical analysis of the term frequency distribution, and the assumption that the relevant set of documents is usually much smaller than the set of non-relevant documents, the probability v_i (a term appears in a document of the non-relevant set) could be estimated based on the term occurrence in the entire collection. A typical value for v_i is the number of documents n_i that contain term i divided by the total number of documents N in a collection. Inserting these rough estimates into Definition 3.12 results in the straightforward scoring function given in Equation 3.13. The formulation is closely related to $tf.idf$ term weighting. Since documents are assumed to consist of binary features, there is no information about term frequency within the documents. As a result, the tf component is missing here. [45, p. 252]

$$\log \frac{\frac{1}{2}(1 - \frac{n_i}{N})}{\frac{n_i}{N}(1 - \frac{1}{2})} = \log \frac{N - n_i}{n_i} \tag{3.13}$$

However, relevance information could be easily collected or might already be present in other retrieval scenarios. Given that term occurrences are known for the relevant

	Relevant	Non-Relevant	Total
	r_i	$n_i - r_i$	n_i
	$R - r_i$	$N - n_i - R + r_i$	$N - r_i$
Total	R	$N - R$	N

Table 3.1: Contingency table summarising relevance information at term level.

and non-relevant set of documents, a contingency table (see Table 3.1) could be used to illustrate different subsets of documents.

The first row in Table 3.1 represents terms that are present in a document under investigation and the second row corresponds to terms that are absent. Thus, r_i is the number of relevant documents containing term i, n_i is the number of documents containing term i, N is the total number of documents in the collection, and R is the number of relevant documents for a query. Let the probabilities u_i and v_i in Definition 3.12 be substituted with corresponding estimates from Table 3.1. The probability u_i could be estimated using r_i divided by R and the probability v_i could result from $n_i - r_i$ divided by $N - R$. Due to using the logarithm in Definition 3.12 the resulting term weights may become undefined in some cases. For example, if the number of relevant documents r_i (that contain term i) equals to zero. This can be avoided by adding the fixed amount of 0.5 or 1 (only for totals) to the figures from Table 3.1. The resulting ranking function is given in Definition 3.14, where $i : d_i = q_i = 1$ illustrates that the scores are computed from query terms that match a document. [45, p. 252f.]

$$\sum_{i:d_i=q_i=1} \log \frac{(r_i + \frac{1}{2})/(R - r_i + \frac{1}{2})}{(n_i - r_i + \frac{1}{2})/(N - n_i - R + r_i + \frac{1}{2})} \tag{3.14}$$

Empirical evaluation showed that document rankings obtained with this function do not correlate very well with human relevance judgements. But one of the most widely-used ranking algorithms, which is known as BM25 (or Okapi BM25), extends this ranking scheme by including term frequency within documents and document length.

3.2.4.2 The 2-Poisson Model

Initially, modelling probability distributions was studied to identify rules for automatic indexing in the context of information retrieval. In [21] a mixture of two Poisson distributions was proposed to model the number of occurrences of terms tf in documents. The original definition (see 3.15) assumes that documents are random streams of term occurrences and X serves as a random variable for the number of term occurrences.

$$P(X = tf) = \lambda \frac{e^{-\mu_1} \cdot \mu_1^{tf}}{tf!} + (1 - \lambda) \frac{e^{-\mu_2} \cdot \mu_2^{tf}}{tf!} \tag{3.15}$$

Another basic assumption of the model is that a collection can be divided into two subsets, where the documents in one subset contain more term references for a specific topic than the documents in the other subset. This fact is represented by the factor λ, which reflects the proportion of documents that belong to the first subset, and by the means μ_1 and μ_2 of the two Poisson distributions, where $\mu_1 \geq \mu_2$. These means can be estimated from the number of term occurrences in the two subsets of a collection. In order to estimate these parameters iterative optimisation approaches like the *expectation maximisation* algorithm can be applied. A document that was randomly drawn from subset one is assumed to have at least the probability of relevance of a document from subset two, because its probability of relevance is assumed to be correlated with the term occurrences for a subject. Since useful terms tend to separate relevant and non-relevant documents well, the means μ_1 and μ_2 of the Poisson distributions will be very different. The main advantage of the 2-Poisson model for information retrieval is that no additional term weighting algorithm is needed. But a major problem lies in obtaining estimates for the parameters λ, μ_1, and μ_2. Nevertheless, the model inspired researchers to include term frequency information from within documents into the probabilistic model, which resulted in the well-known BM25 ranking algorithm. [84, p. 10f.]

3.2.4.3 BM25 Ranking Model

The BM25 ranking model is an extension of the binary independence model, where BM stands for *best match* and the numbers represent identifiers for different combinations of term weights that were used during experimental validation of the model [89, 149]. It incorporates the term frequencies from within documents, as well as the document lengths into the scoring to account for the nature of full-text collections which were made available in the TREC experiments. The most common form of the scoring function is given in Definition 3.16. The summation is accumulated for all terms of the query and the variables in the left hand part represent the values from Table 3.1. Here, r_i and R are set to zero if no relevance information is available. A few more variables are introduced to account for term frequencies: f_i is the frequency of term i in the document, and qf_i is the frequency of term i in the query.

$$\sum_{i \in Q} \log \frac{(r_i + \frac{1}{2})/(R - r_i + \frac{1}{2})}{(n_i - r_i + \frac{1}{2})/(N - n_i - R + r_i + \frac{1}{2})} \cdot \frac{(k_1 + 1) f_i}{K + f_i} \cdot \frac{(k_2 + 1) qf_i}{k_2 + qf_i} \tag{3.16}$$

K, k_1, and k_2 are parameters to adjust the influence of the term frequency and document length on the final score for a document. The constant k_1 is used to smooth the impact of the term weighting component (middle part in Definition 3.16) for higher values of f_i. Term frequency information is omitted if k_1 is set to zero. Large values for k_1 correspond to a linear impact of f_i on the document score. Empirical evaluations showed that a value of $k_1 = 1.2$ results in good retrieval performance. In this case the impact of the term weighting component is non-linear, i.e. after a few occurrences of a term, further occurrences will have almost no impact for ranking the document. Similarly, the constant k_2 calibrates the influence of the term frequency qf_i in the query. In practice, retrieval performance seems to be less sensitive to k_2 and a large range of values (0 to 1,000) was found to be useful. The parameter K is used to normalise the term weighting component by the length of the document as shown in Equation 3.17. [45, p. 254f]

$$K = k_1((1 - b) + b \cdot \frac{L_d}{L_{avg}}) \tag{3.17}$$

It uses the parameter b, which determines the impact of the document length normalisation. If b is set to zero no length normalisation is used, like in the query term weighting component. A value of $b = 1.0$ indicates full document length normalisation. The variables L_d and L_{avg} represent the length of the document under examination and the average document length in the collection. $b = 0.75$ is a typical value found by means of empirical evaluation on full-text collections. An advantage of the BM25 ranking model is that it can be tuned to optimise an effectiveness measure used for evaluation. Thus, it can be used to adjust the ranking function based on user studies to improve the performance of an actual search application. A detailed motivation and discussion of numerous empirical results for the extension of the binary independence model to incorporate term weighting and document length normalisation is given in [89].

3.2.4.4 Language Models

The concept of generating *language models* originates from natural language processing research. It was developed to generate probabilistic models of language for automatic recognition of speech. Current systems for automatic speech recognition usually combine probabilities of two models: the acoustic model, and the language model. While the acoustic model generates candidate text from phonetic transcriptions, the language model estimates which of the candidate text snippets are most likely to appear in the corresponding language. In the late 1990’s the first attempts were made to adapt language modelling for ranking in IR. [84, p. 12]

The most straightforward language model is known as the *uni-gram* language model. In terms of IR ranking, this model simply estimates a probability of occurrence for every word in the vocabulary of a collection. Thus, in the representation of language models the topical content of a document is defined as a probability distribution over words. The most common way of modelling these probabilities is to assume a *multinomial distribution* of words. As a consequence, documents are treated as being a sample of text drawn from a model distribution and the sample is then used to estimate the actual document language model. Each document language model can then be used to calculate the probability of observing a particular sequence of words, e.g. a query. [45, p. 257]

Language models are usually created on document level, but it is also possible to generate models from queries. This leads to three major models for retrieval based on language models. The first is based on estimating the probability of a document language model generating the query (as described below) and referred to as *query likelihood model*. A second approach is the *document likelihood model*, where the probability of a query language model generating the (finite sequence of words given by a) document is estimated. There is, however, the problem that queries tend to be short and therefore the estimation of query models will be worse than for document models. The advantage of query language models is that relevance feedback can be easily incorporated into these models. A third possibility for retrieval based on language models is to directly compare the models for a query and a document. [126, p. 239]

Ranking documents based on the query likelihood model is usually done by estimating the conditional probability $P(D|Q)$ for a document D given a particular query Q. Applying Bayes‘ rule for conditional probabilities (see Section 3.2.4.1) results in Equation 3.18, where the probability $P(Q)$ is a constant for all documents D and $P(D)$ is a prior probability for a document and thus often regarded as uniform for a given collection of documents. Since $P(Q)$ and $P(D)$ can be ignored based on this knowledge, the basic language model approach simply ranks documents by the probability that a query would be observed as a random sample from the respective document language model.

$$P(D|Q) = \frac{P(Q|D) \cdot P(D)}{P(Q)} \tag{3.18}$$

Given a document language model M_d, the document ranking could be created using a multinomial language model, where documents are considered as different classes of the language (see Equation 3.19). The multinomial coefficient K_Q is defined in Equation 3.20, where L_d is the number of words in document d, $tf_{i,d}$ is the number of occurrences of a particular word in document d, and n is the number of words in the query Q. Again, the coefficient K_Q is usually ignored for faster computation of the probabilities based on the assumption that a query (represented by a sequence of words) is treated as a *bag of words*. This will result in equal likelihood ratios for any ordering of that particular bag of words. The motivation behind this basic model is the assumption that users submit queries that contain the words that are likely to appear in a relevant prototype document they have in mind. [126, p. 242f]

$$P(Q|M_d) = K_Q \prod_{i=1}^{n} P(q_i|M_d)^{tf_{i,d}} \tag{3.19}$$

$$K_Q = \frac{L_d!}{tf_{1,d}! \cdot tf_{2,d}! \cdot ... \cdot tf_{M,d}} \tag{3.20}$$

The major problem with estimating the probability $P(Q|M_d)$ as given in Equation 3.20 is that if a document does not contain a particular query term, the resulting probability for that document will be zero due to the calculation based on the product of single word probabilities. This might not be problematic in applications, where typical queries are short and the number of distinct documents is large as for example in web search. But in many other cases this model is clearly inappropriate as it may cause the system to return no documents at all, or alternatively, the system might not be able to distinguish between documents that lack different numbers of query words. A technique called *smoothing* is applied to overcome this limitation. It also helps to deal with the problem of data sparsity, i.e. the amounts of text for the estimation of the (query or document) language models are typically small. In general smoothing discounts the probability estimates for observed events and assigns a non-zero probability to unseen events, i.e. words that in effect have zero probabilities in a respective language model. The standard approach to estimate the values for these unseen events is to apply a *background model*. Such a background model could be created from the entire collection of documents, which could be regarded as a *collection language model*. A smoothed language model usually estimates probabilities using a simple linear interpolation of the document language model and the background model. This interpolation approach is also referred to as *Jelinek-Mercer smoothing*. It is illustrated in Equation 3.21, where $P(q_i|M_d)$ is the probability for a query word in the document model, $P(q_i|M_c)$ represents the probability for a query word in the background model, e.g. the collection language model, and λ is an unknown variable that has to be estimated empirically, but it has to adhere to the interval $0 \leq \lambda \leq 1$.

$$P(Q|D) = \prod_{i=1}^{n} \lambda \cdot P(q_i|M_d) + (1 - \lambda) \cdot P(q_i|M_c) \tag{3.21}$$

In practice, actual ranking functions based on this linear combination of language models use mathematical transformations similar to those for the binary independence model (see Section 3.2.4.1) to avoid the accuracy problem caused by multiplying numerous small fractions of probabilities. One possible ranking function can be derived as illustrated in Equation 3.22, where $f_{q_i,D}$ is the number of times the query word q_i appear in document D, $|D|$ is the number of words in document D, and the fraction $f_{q_i,D}/|D|$ estimates $P(q_i|M_d)$ (see Equation 3.21). Values for the probabilities of the collection language model are estimated similarly using $c_{q_i}/|C|$, where c_{q_i} is the number of times a query word q_i occurs in the collection C, and $|C|$ is the total number of word occurrences in the collection.

$$log(P(Q|D)) = \sum_{i=1}^{n} log(\lambda \cdot \frac{f_{q_i,D}}{|D|} + (1 - \lambda) \cdot \frac{c_{q_i}}{|C|}) \tag{3.22}$$

In the given ranking function the variable λ is used to control the influence of the smoothing component. Larger values for λ result in less smoothing and lead to queries being treated similarly to conjunctive combination like a Boolean AND. Empirical evaluation of the ranking function using different values for λ has shown that more smoothing works well for short queries, whereas less smoothing is better for longer queries. Thus, it was suggested to make λ a function of the query size. [84, p. 13], [45, p. 260f]

Many other forms of language model estimation exist, most of which were originally developed for automatic speech recognition applications. One of them, which is more effective for ranking in general, is called *Dirichlet smoothing* [198]. A Dirichlet distribution is a standard way of incorporating prior knowledge, i.e. a background model, for the estimation of the probabilities of a multinomial distribution. If there is no text available to estimate an actual document language model, the probability estimation for that particular document will be solely based on the pseudo-values from the Dirichlet distribution. However, if there is more text available, i.e. the document under examination is long, the less influence the background model (or Dirichlet distribution) will have on the probability estimation. The effect of this influence is controlled by the parameter μ as can be seen in Equation 3.23. Typical values for μ are real numbers greater than zero and empirical evaluations showed that values between 1000 and 2000 are the most effective. [45, p. 262f]

$$log(P(Q|D)) = \sum_{i=1}^{n} log(\frac{f_{q_i,D} + \mu \cdot \frac{c_{q_i}}{|C|}}{|D| + \mu}) \tag{3.23}$$

Another approach to language modelling is to estimate probabilities for documents and queries. In a subsequent step the language models for documents and queries are compared to generate a document ranking. The reasoning behind this technique is that similar models are likely to be about the same topic. A common measure for the difference of model estimates, i.e. probability distributions, from information theory is the *Kullback-Leiber divergence* (or KL-divergence). In general the KL-divergence is defined based on a true probability distribution P and an approximation of P given by the probability distribution Q, defined as follows:

$$KL(P||Q) = \sum_{x} P(x) \cdot log(\frac{P(x)}{Q(x)}) \tag{3.24}$$

Important features of the KL-divergence are its asymmetric characteristic and that resulting values are always positive. For ranking purposes the query language model (or relevance model) R is assumed to be the true distribution P, as it models the relevant words for the query. Since larger values of the KL-divergence indicate that the probability distributions in question are further apart, the negative KL-divergence is used for ranking. This is illustrated in Definition 3.25, where w represents a word from vocabulary V.

$$\sum_{w \in V} P(w|R) \cdot log(P(w|D)) - \sum_{w \in V} P(w|R) \cdot log(P(w|R)) \tag{3.25}$$

The right hand term in Equation 3.25 is independent of the document and can be ignored for ranking. The probability $P(w|R)$ of a word, given the relevance model R

can be estimated using the frequency $f_{w,Q}$ of the word in the query and the total number of words in the query $|Q|$. The resulting ranking function is given in Definition 3.26. It is rank equivalent to the query likelihood ranking (see the document language model estimate in Equation 3.22), because the frequency of query words is a factor in the summation, i.e. words that do not occur in the query do not contribute to the probability estimate of the document. As a result, the query likelihood ranking model can be regarded as a special case of a retrieval model that ranks documents by comparing the probability estimates of the (assumed to be true) relevance model based on a query, and the probability estimates of a document language model. [45, p. 265ff]

$$\sum_{w \in V} \frac{f_{w,Q}}{|Q|} \cdot log(P(w|D)) \tag{3.26}$$

3.2.4.5 Divergence from Randomness Framework

Based on the approaches to language modelling the *divergence of randomness framework* (DRF) was proposed [6]. A main contribution of the framework is that it provides a number of statistically motivated models for ranking which are free of parameters. In contrast to the smoothing approach of language models no training data is needed to estimate the parameters.

In the divergence from randomness framework, two probabilities P_1 and P_2 are used to model the weight w of a term (see Equation 3.27). It shows that the framework originates from the 2-Poisson model (see Section 3.2.4.2). The divergence of randomness framework is based on the following hypothesis: the higher the difference of the frequency of a term within a document and the frequency of the term in the collection, the more information is carried by the term in the respective document. Ranking models in the DRF can be obtained in three steps: (a) selecting a model of randomness and normalising the informative content of a term (b) based on the information gain, and (c) based on document lengths. The hypothesis of using the probabilities P_1 and P_2

to model random term distributions and information gain is closely related to ranking based on document language models and smoothing with a background model. This observation was investigated more formally in [41].

$$w = (1 - P_2) \cdot (-log_2 P_1) = -log_2 P_1^{1-P_2} \tag{3.27}$$

The probability P_1 models the distribution of terms that bear little information and thus are assumed to be randomly distributed across the collection of documents. Several probability distributions were proposed to model this randomness. In the course of the development of the framework, namely binomial distribution, Poisson distribution (see Definition 3.28), Bose-Einstein statistics (see Definition 3.29), the inverse document frequency model (see Definition 3.30), and a mixed model using Poisson and the inverse document frequency (see Definition 3.31) have been studied. The basic weighting formula in the DFR framework depends on four random variables: the term occurrence frequency tf within a document, the size of the document collection N, the size of the *elite set* of documents of the term n, i.e. a small number of documents containing the term, and F, the total number of occurrences of the term in the collection.

$$tf \cdot log_2 \frac{tf}{\lambda_1} + (\lambda_1 + \frac{1}{12 \cdot tf}) \cdot log_2 e + \frac{1}{2} log_2 (2\pi \cdot tf) \tag{3.28}$$

$$- log_2 (\frac{1}{1 + \lambda_2}) - tf \cdot log_2 (\frac{\lambda_2}{1 + \lambda_2}) \tag{3.29}$$

$$tf \cdot log_2 \frac{N + 1}{n + 0.5} \tag{3.30}$$

$$tf \cdot log_2 \frac{N + 1}{n_e + 0.5} \quad with \;\; n_e = N \cdot (1 - (\frac{N - 1}{N})^F) \tag{3.31}$$

The probability P_2 is used to normalise the information gain of a term based on the *after-effect* phenomenon in sampling theory. The underlying assumption is that once a very rare term is observed in a document, the probability of the document being relevant increases rapidly when this term is observed repeatedly. The DFR framework accounts for this effect by modelling a conditional probability that approaches 1 for increasing term frequencies. Two methods were suggested to estimate this probability: (1) using Laplace's law of succession (see Equation 3.32), or (2) based on Bernoulli trials (see Equation 3.33). The term frequency normalisation factor tf_{n_1} in Equation 3.29 depends on the number F of occurrences of the term in its elite set of documents, and the number of documents n in the collections.

$$tf_{n_1} = \frac{1}{n \cdot (tf + 1)} \qquad (3.32) \qquad\qquad tf_{n_1} = \frac{F + 1}{n \cdot (tf + 1)} \qquad (3.33)$$

The final step to obtain a ranking model in the DFR framework is to incorporate document length normalisation. Based on the density function for tf, two basic formulas were suggested to accomplish this. Both formulas (see Equations 3.34 and 3.35) are dependent on the document length l and the average length of the documents in the collection l_{avg}.

$$tf_{n_2} = tf \cdot \frac{l_{avg}}{l} \qquad (3.34) \qquad\qquad tf_{n_2} = tf \cdot log_2(1 + \frac{l_{avg}}{l}) \quad (3.35)$$

The researchers who developed the DFR framework also showed the relation between the BM25 ranking model and a particular model in their framework. In fact, when the inverse document frequency model for P_1 as given in Equation 3.30, is combined with the Laplace normalisation of the information gain (see Equation 3.32), they were able to formally show this relation. They tested various models that were created with the DFR framework, and compared their performance to the commonly used BM25 ranking formula. In their analysis of the results, they state that the model, based on the mixture of inverse document frequency and Poisson distribution (see Equation 3.31) combined with normalisations tf_{n_1} and tf_{n_2} is superior to BM25 at many recall levels. Furthermore, the authors of the DFR framework ran a detailed investigation of their normalisation techniques. Their experiments showed that some normalisations, namely the combinations resulting from Definitions 3.32, 3.33 and 3.34 — L2 and B2 in the original terminology, seem to be universal factors, meaning that they work independently of the models. They also found that L2 (the combination of Definitions 3.32 and 3.34) is less sensitive to variation in document length than B2 (the combination of Definitions 3.33 and 3.34).

3.2.4.6 The Inference Network Model

The information retrieval models discussed so far have in common that they use ranking functions that are derived by combining various term occurrence frequencies. In order to allow multiple document representations, as well as combining the evidence from different queries and types of queries, and to facilitate flexible matching strategies to overcome the problem of vocabulary mismatch, a theoretical framework termed *inference network model* [178, p. 2f] was proposed. The formal foundation of the model is a *Bayesian inference network*, i.e. a directed and acyclic dependency graph. In general, the nodes of the graph represent pieces of evidence and the arcs describe how the evidence should be combined. All pieces of evidence in the network are binary random variables. The basic inference network for IR consists of four layers of events (or evidence): (a) documents d, (b) document representations r, (c) queries q, and information needs I (see Figure 3.2 [84, p. 12]).

The first two layers remain static when documents do not change over time. Thus, they are sometimes combined and referred to as document network. Using this representation of a directed acyclic graph in the context of information retrieval allows the modelling of causal effects between the layers of the network: from a document node on the top layer; over a number of different representation nodes of the document contents; to possible queries that combine the evidence from representation nodes; down

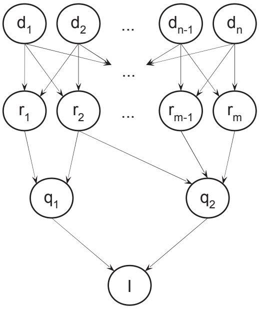


Figure 3.2: Schematic representation of a simple inference network, redrawn from [84, p. 14].

to a specific information need. An apparent question is, why represent an information need by a number of queries instead of just one? One reason is to think of an initial query and a number of expansions or reformulations. It is also possible that one query is represented in various formal reformulations (e.g. Boolean) using this model.

In practice it may become very difficult to compute all possible outcomes for an information need I, if there are a large number of query nodes q. Given a fixed number of k query nodes and assuming the nodes represent binary variables, there are 2^{k+1} possibilities of $P(I|q_1, ..., q_k)$ for the information need I. As with many other models actual values are estimated using heuristics. A straightforward heuristic for the document nodes would be to use $P(D) = 1/n$ as prior probability, where n is the size of the collection. In a practical implementation of the model, a single network is constructed for every document, assuming that only one document is observed at a time [129]. This premise allows the ignoring of the document layer. The representation layer in each of the (document) networks is approximated by a number of language models, which represent significant parts (evidence) of the document structure. Other features, e.g. formal meta-data, like a publication date, that cannot be represented in a language model are also possible. Given the approximations from the representation layer for each document, query nodes and the information need can be estimated using standard probability distributions. They are defined by means of various *belief oper-*

ators [45, p. 276f.]. The most common belief operators can be calculated as given in Definitions 3.36 through 3.40, assuming that node q has m parents with a probability p_i of being true. A weight w_i can be associated with its parent node i to minimise or emphasise the importance of its underlying evidence.

$$bel_{or}(q) = 1 - \prod_{i}^{n} (1 - p_i) \tag{3.36}$$

$$bel_{and}(q) = \prod_{i}^{n} p_i \tag{3.37}$$

$$bel_{wand}(q) = \prod_{i}^{n} p_i^{w_i} \tag{3.38}$$

$$bel_{sum}(q) = \frac{\sum_i^n p_i}{n} \tag{3.39}$$

$$bel_{wsum}(q) = \frac{\sum_i^n w_i \cdot p_i}{\sum_i^n w_i} \tag{3.40}$$

It can be shown that the given operators can be calculated in linear time. The practical relevance of the inference network model is demonstrated by its integration in common open-source search engines from research institutions, namely Inquery and Indri. However, using the full potential of the model requires the implementation of a query language, which allows complex combinations of evidence.

3.3 Feedback Mechanisms

A feedback mechanism in IR describes the process of providing additional information to an information need by means of relevance judgements on a specific number of documents. Depending on the preferred way to obtain these judgements, one can differentiate between three types of feedback methods: implicit feedback, explicit feedback, and pseudo-relevance feedback. Implicit feedback describes the process of collecting relevance information from user data (typically user behaviour in response to a presented result list). Implicit relevance feedback is useful in search applications with many users, such as web search engines. Many users of web search engines submit similar requests and the service operators can collect implicit relevance information by counting the clicks on the web pages presented in the result lists. As a result, this behaviour of the users can be used to rank future result lists for these queries based on the gathered information. Using this kind of feedback results in ranking the results based on popularity rather than relevance, which might be problematic in other use cases. In contrast to that, explicit feedback is typically collected with the knowledge of the user that his feedback is used as relevance judgement. Since the user is involved in this process, explicit feedback is the most expensive method to obtain relevance feedback. One of the most widely used explicit relevance feedback models is presented in the following subsection.

Due to its simplicity and the lowest cost, pseudo-relevance feedback is most widely adopted in ad-hoc search scenarios. Its alternative name, blind relevance feedback, indicates that the automatic method may not work well for every search application. The details of the pseudo-relevance feedback model are presented in Section 3.3.2.

3.3.1 Explicit Relevance Feedback

The Rocchio relevance feedback model introduced in [150] was one of the first adaptations of the vector space approach. It is based on the concept of an optimal query, which uses information about documents that were identified as being either relevant or not. Based on an initial query the Rocchio algorithm aims to optimise the term weights of a query in order to improve retrieval effectiveness. Equation 3.41 reproduces the formula to derive an optimal query, where $\vec{q}_{new}$ and $\vec{q}_{old}$ represent the two query vectors, $|r|$ and $|nr|$ represent the number of documents that were identified to be relevant or non-relevant, and $\vec{d^r}$ and $\vec{d^{nr}}$ are corresponding document vectors. The additional parameters α, β, and γ are used to adjust the weights of each of the components of the formula.

$$\vec{q}_{new} = \alpha \cdot \vec{q}_{old} + \frac{\beta}{|r|} \cdot \sum_{i}^{r} \vec{d_i^r} - \frac{\gamma}{|nr|} \cdot \sum_{i}^{nr} \vec{d_i^{nr}} \tag{3.41}$$

From the three parts in the formula, the two summations deserve some further explanation. The first sum calculates the average of the weights of documents that were identified to be relevant. An analogous sum is computed for the non-relevant documents. By adding the average weight of relevant documents and subtracting the average weight of non-relevant documents the resulting query vector is shifted towards the relevant documents and away from irrelevant documents. As a result, it can be expected that retrieval performance is improved. Experimental studies showed that all unseen documents for an initial query give a good approximation of the set of non-relevant documents. Reasonable parameters for weighting the formula are $\alpha = 8$, $\beta = 16$, and $gamma = 4$. Since queries also consist of terms it is useful to restrict the number of additional query terms for efficacy reasons. [45, p. 247]

3.3.2 Pseudo-Relevance Feedback

In many search applications, it is infeasible to collect relevance feedback neither explicitly nor implicitly. Automatic pseudo-relevance feedback (PRF) is widely adopted in these scenarios. The technique works just as using explicit feedback, except for the source of the feedback information. PRF is based on the assumption, that the documents ranked in top positions in response to an initial query are likely to be relevant. Thus, the top d documents are used to extract a specific number of terms t for the reformulation of the original query.

Typical software implementations of this technique rely on term weighting schemes like *tf.idf* (see Section 3.2.3). The assumption behind the automatic PRF approach and the three parameters of the standard implementation indicate that the technique will not work for every query. Empirical evaluation [28, 114, 175] on the TREC ad-hoc test collections demonstrated, that automatic pseudo-relevance feedback improves retrieval effectiveness on average. Nevertheless, given a specific information need, it is not possible to predict which are the optimal parameters for PRF. Moreover, it is also possible that PRF fails on particular queries regardless of the PRF configuration. In this work the effect of various PRF configurations is investigated in order to study the impact on retrieval effectiveness. In addition to that, there is the interest in analysing possible interactions between selected software implementations of other system components and the standard PRF approach.

3.4 Summary

This chapter gave an overview about state-of-the-art models of the three essential components of every information retrieval system. The selection of methods provides insights into the variety of techniques that are currently available and commonly used. Due to the multitudinous applications of search technology, the present selection cannot be complete. For all of the three components, none of the methods dominates all others in terms of efficiency or effectiveness. As a result, choosing the models that perfectly suit any particular task and data set is not possible without empirical evaluation.

In addition to the problem of selecting the right model for a search problem, the software implementation of corresponding components and their orchestration in a retrieval system may also affect efficacy and effectiveness. Thus, one of the central goals in the remainder of this work is to gather evidence which of the different state-of-the-art text pre-processing algorithms, retrieval models, and automatic feedback methods are more suitable for finding information in specific types of text documents.

A further aim is the investigation of how interactions between these key components of retrieval systems affect the retrieval effectiveness of the complete system. Current standard toolkits and the Xtrieval framework for IR evaluation are discussed in detail in the following Chapters 4 and 5. This course of action provides practical foundations for the large-scale empirical approach to automated evaluation at component-level presented in Chapter 7.

4 IR Frameworks & Software Implementations

Empirical evaluation and analysis are mainsprings for the development of new models and theories in IR. The present chapter links theoretical IR models from the previous Chapter 3 to existing open-source software implementations. The purpose of this course of action is to illustrate the selection of software toolkits to be included in the IR evaluation framework Xtrieval. The architecture of this framework and a selection of scientific evaluation tasks is presented and discussed in the next Chapter 5.

The present chapter is organised as follows. First, an overview of a number of publicly available framework is provided. A number of key features of software implementations are formulated in order to establish a selection of only a few toolkits. Based on this selection, the most important IR toolkits are discussed in detail. This discussion focuses on central advantages of each of the software implementations. Important limitations are also pointed out, if present. As a final step, an overview on further frameworks emphasises our focus on the frameworks, which are most widely used in the IR community. Before the chapter is summarised in the final section, a digression to natural language processing frameworks demonstrates the impact of complex text understanding algorithms on search applications in general.

4.1 Open-Source Toolkits

Table 4.1 gives an overview on publicly available open-source frameworks which can be used to develop a search engine application. All of them are frequently (though differently) used and under further development, i.e. releasing updates with new features and bug fixes on a regular basis. The listed software implementations can be distinguished based on their popularity both in scientific research and the number of adaptations in commercial search applications, although reliable studies on the usage of these frameworks do not exist.

From the provided list of software packages, Lucene can be regarded as a state-of-the-art system for real-world search applications, which makes use of the Solr enterprise search server. Solr extends the Lucene API to fulfil virtually any requirement for a search application. Xapian and Terrier are used in a remarkable number of applications, although the actual number of adaptations is considerably smaller than for Lucene. The Terrier platform and the Lemur project, which also includes Galago, are toolkits which are mainly used to support research in IR. Advancing research in the domain of web search is the purpose of Wumpus and Zettair. Indri (the Lemur search application), Wumpus, and Zettair were compared in terms of efficiency in [26]. The authors of this study report that conducting a fair comparison is incredibly difficult, because operating systems, system architectures, and hardware configurations are variables that are hard to control in a decentralised comparison. The results showed that Wumpus was about four times faster than Zettair and the latter was again about four times faster than Indri. Minion and Sphinx are open-source projects that are under development in private enterprises.

Name	Dev. Cycle	License	Lang.	Ranking	Citations
Galago[1]	2009-*	BSD	Java	Language Model, Inference Network	6
Lemur[2]	2000-*	BSD	C++	Language Model, Inference Network	252
Lucene[3]	2000-*	Apache	Java	Boolean Matching, TF-IDF variant	339
MG4J[4]	2005-2010	LGPL	Java	BM25, TF-IDF, others	15
Minion[5]	2008-*	GPL	Java	BM25, TF-IDF	0
Sphinx[6]	2001-*	GPL	C++	BM25, Boolean	0
Terrier[7]	2004-*	MPL	Java	BM25, DFR, TF-IDF, Language Model, others	172
Wumpus[8]	2006-2009	GPL	C++	BM25, Vector Space	9
Xapian[9]	2001-*	GPL	C++	BM25, Prob. variant, Boolean	10
Zettair[10]	2004-2009	BSD	C	BM25, Prob. variant, Dirichlet LM	29

Table 4.1: Open-source search engines and toolkits.

A basic comparison of various open-source search engines is given in [130]. The authors use quantifiable parameters to create a general overview on system-level that is intended to allow the reader to find the best system for his particular use case. In addition, they evaluated system efficiency and effectiveness of a subset of their initial selection of systems. Due to the number of systems, it is not clear which version they used for each of the systems and how the systems were configured for the experiments in order to ensure fair comparison. This seems to imply that an impartial comparison of retrieval systems is impossible, because almost every system can be tuned for better performance both in terms of efficiency and effectiveness.

In order to substantiate our selection of systems, a brief review of the key features of the systems listed in Table 4.1 is provided. A major aim of the present chapter is to illustrate the reasons for including Lucene, Terrier, and Lemur in our IR evaluation framework Xtrieval. In general, Xtrieval is a framework which is designed to allow

---

[1] `http://www.galagosearch.org/`, retrieved on March 1, 2012
[2] `http://lemurproject.org/lemur.php`, retrieved on March 1, 2012
[3] `http://lucene.apache.org/`, retrieved on March 1, 2012
[4] `http://mg4j.dsi.unimi.it/`, retrieved on March 1, 2012
[5] `http://minion.java.net/`, retrieved on March 1, 2012
[6] `http://sphinxsearch.com/`, retrieved on March 1, 2012
[7] `http://terrier.org/`, retrieved on March 1, 2012
[8] `http://www.wumpus-search.org/`, retrieved on March 1, 2012
[9] `http://xapian.org/`, retrieved on March 1, 2012
[10] `http://www.seg.rmit.edu.au/zettair/`, retrieved on March 1, 2012

easy integration of any other IR toolkit. Table 4.1 points out that the number of retrieval frameworks is large. This diversity is problematic for the decision on which tools should be integrated, because each of the IR toolkits or libraries would have to be examined in detail. For that reason, two abstract criteria were defined to narrow down the list.

The first criteria is a citation analysis. It was conducted to illustrate the impact of the systems on the research community. This is in line with the purpose of the Xtrieval framework: scientific evaluation of IR systems and configurations. The citation analysis is based on publications about, or referring to, the IR toolkits and the results are presented in the rightmost column of Table 4.1. It is based on the following design:

- *General set-up*
  The ACM digital library[11] served as source for the citation analysis, because it aggregates titles, abstracts, and citations, of all major conferences and workshops on IR research. The actual research was conducted on February 11th, 2012.

- *Types of publication*
  In order to restrict the frameworks to their scientific applications, the citation analysis focused on a few major IR conferences, workshops, and tutorials, namely SIGIR, ECIR, CIKM, CLEF, and WWW. The selected list can be considered as being representative for publications in the field of IR research, although further events with considerable impact do exist.

- *Time period*
  The total number of publications referring to each of the IR toolkits was aggregated for the past ten years. This period was selected to be able to distinguish between frameworks that exist for a long time. Since this approach penalises

---

[11] `http://dl.acm.org/`, retrieved on March 1, 2012

newer software implementations, the absolute numbers should be assessed with respect to the corresponding period of existence.

It is clear that the number of citations for an IR toolkit is just a measure of popularity. As a result, it only gives a vague idea which of the frameworks might be suited best for the inclusion in Xtrieval. However, it is presumed that the research community as a peer network of knowledge in IR, can act as a suitable indicator. It can be seen in Table 4.1 that the absolute numbers vary significantly. Lemur, Lucene, and Terrier are the only toolkits that returned triple-digit numbers. For Zettair and MG4J, more than a dozen citations were found. The remaining tools received only little perception of the IR research community.

The second criteria for the assessment of the IR system software implementations focusses on the usefulness in terms of the evaluation of IR theories. Thus, the number different system components as presented and discussed in Chapter 3 serves as central figure here. But the criteria also covers the question of how new instances of IR components like text pre-processing algorithms, ranking models, or relevance feedback, can be integrated. Table 4.1 demonstrates that most of the toolkits cover at least two different ranking models. As a result, we have to elaborate the details for each of the frameworks in order to discuss the key IR system components they cover.

4.1.1 Apache Lucene

Probably the most widely used open-source search engine toolkit is Apache Lucene. The documented number of search applications that are based on Lucene contains almost 200 entries.[11] This list includes almost any kind of web service that requires search technology for its business, large software projects like Eclipse and JIRA, and even commonly known companies like AOL, IBM, LinkedIn, and Twitter. The key to this tremendous success is its large developer community. Although it was originally

---

[11] `http://wiki.apache.org/lucene-java/PoweredBy`, retrieved on March 1, 2012

intended for creating search applications in commercial environments, it also received a number of contributions from the IR research community. Similar to other open-source software toolkits Lucene is more an advanced API than a search application. It is implemented in Java, but several ports to other commonly used programming languages are also available, including C/C++, C#, Perl, PHP, Python, and Ruby. In addition to that, various complementary software frameworks are available to build powerful search applications that are based on Lucene. The most important projects are Solr[12], Nutch[13], Hadoop[14], and Tika[15], all of which are open-source projects that are hosted and funded by the Apache Software Foundation (ASF). They will be considered in the following paragraphs.

Solr is a standalone enterprise search server that features a REST-like XML interface, i.e. it is a server wrapper around the Lucene API. It supports indexing and searching via XML over HTTP, or JSON and allows the easy integration into any client application. The Solr developer community implemented a diverse collection of Solr clients that covers virtually any programming language. For that reason, it is relatively easy to access the Lucene API in cases where software developers are bound to languages other than Java. Some of the key features of Solr include support for faceted browsing, geo-spatial search, incremental index updates and replication, and highly configurable caching for performance optimisation in terms of efficacy.

In 2004, Nutch was the first open-source search engine toolkit for the web. The motivation for creating Nutch was the decreasing number of commercial web search engines and the serious concern for a potential monopoly in that business [127, p. 9]. The software is designed to handle crawling, indexing, and searching of billions of web pages that are frequently updated. Of course, Nutch uses Lucene at its core to handle indexing and retrieval. Scalability is a major issue when dealing with billions of text documents or with large amounts of simultaneous search traffic. These problems are

---

[12] `http://lucene.apache.org/solr/`, retrieved on March 1, 2012

[13] `http://nutch.apache.org/`, retrieved on March 1, 2012

[14] `http://hadoop.apache.org/`, retrieved on March 1, 2012

[15] `http://tika.apache.org/`, retrieved on March 1, 2012

not tackled by Lucene, but both Nutch and Solr support index sharding and replication through Hadoop. Hadoop is another open-source project in the Apache Software Foundation and it provides tools like HDFS, a distributed file system, and it also contains a software framework that implements MapReduce for distributed computation.

The major difference between the two sub-projects is that Nutch contains a web crawler, tools for document parsing, and a web search front end. In contrast to that, Solr was designed as a server wrapper around Lucene that can be easily configured using XML. Over time the relation between Solr and Nutch has shifted. Currently, Nutch integrates Solr (and also Lucene) and adds further web-specific functionality. Document parsing in Nutch is handled by the Tika project, which is another top-level project of the Lucene family in the Apache Software Foundation. Tika supports many formats of documents like web pages (HTML), office documents (Microsoft Office, OpenOffice, etc.), portable document formats (PDF, EDF, RTF), and even archive file formats (TAR, ZIP, BZIP2). The Tika framework provides a standard API for document parser plugins to extract document content and metadata. Tika allows the processing of all supported types of documents with a single common API and therefore purges the programming code necessary to support several types of text documents in a search application.

This list of extensions for Lucene demonstrates that it is only a programming library for core IR functionality and not a complete search engine application, although it is easy to build a simple search application by using the API. The tools provided by Lucene are discussed by referring to the general architecture of a modern search application (see Chapter 3), starting with a closer look at the components that are provided for indexing a document collection. For simplicity, it is assumed that a collection of structured text documents is available and residing in a particular directory on the file system. That allows to skip the additional steps of acquiring the text content using a crawler from a framework like Nutch, and parsing possibly different types of documents with an API like Tika.

4.1.1.1 Text Processing & Indexing

The central concepts for constructing an index using Lucene are documents and fields. A document typically consists of a number of different fields with content values, like title, abstract, and content. Given the structured document collection this transformation is a straightforward process, because it allows the usage of the structure of the documents to separate the fields. But sometimes the structure of the documents that need to be searched is neither well-defined nor known beforehand. Then a more careful design of the index application is needed to ensure good search experience. Given that the documents in a collection are present as a text stream. The next step is to break the stream into a number of individual atomic elements. In Lucene these elements are called *Tokens* and the process of transforming a text stream into token is realised with *Analysers*. A unique feature of Lucene is its powerful and flexible document analysis concept. Other open-source search engines neither provide such an intelligent concept that allows flexible configuration of the text transformation process, nor do they include a comparably rich collection of text analysers. The text transformation concept of Lucene deserves a detailed explanation, which follows next.

A token as an atomic unit of text in Lucene includes a number of attributes that are important to ensure flexibility when searching in an index. Of course the token has a text value, which contains the word it represents. In addition to that, it also retains the start and end character offsets of the word in the original text. And finally, token type and position increment are further mandatory attributes. A stream of tokens is the generic output of the analysis process. A *TokenStream* in Lucene is an abstract class that allows two different ways of emitting tokens. A *Tokenizer* reads characters from a given *Reader* and generates tokens. While a *TokenFilter* takes tokens in and returns new tokens by either adding or removing tokens or altering token attributes. This hierarchy of generating tokens allows the construction of *analyser chains*. Figure 4.1 shows an example with three *TokenFilters*.

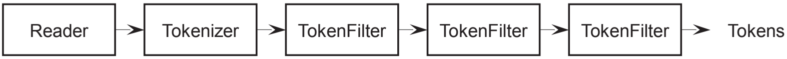


Figure 4.1: An analyzer chain with three TokenFilter instances in Lucene, redrawn from [127, p. 117].

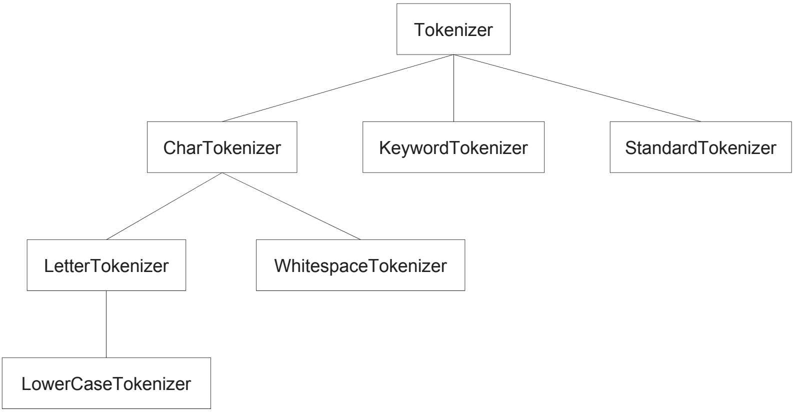


Figure 4.2: The Tokenizer hierarchy in Lucene (version 3.3), redrawn from [127, p. 119].

Lucene also provides a number of *Tokenizers* that can be used depending on the desired application. The hierarchy of these classes is illustrated in Figure 4.2. The *StandardTokenizer* will be suitable for most scenarios that deal with regular European languages. It is a sophisticated *Tokenizer* that implements a grammar to produce tokens for high-level types like email addresses and others. The remaining *Tokenizers* are self-explanatory by means of their names. In addition to this basic set, Lucene also provides more complex *Tokenizers*, e.g. language-specific *Tokenizers* for Chinese, or two variants that implement n-gram tokenising that can be used for character n-gram stemming (see Section 3.1.3).

An even larger collection of implementations extends the *TokenFilter* class. Lucene version 3.3, which is the latest release at the time of writing, contains over 70 *To-*

kenFilter implementations. Figure 4.3 represents an extracted hierarchy that covers general *TokenFilters*. Among these the *TeeSinkTokenFilter* deserves special attention. It reads tokens and then produces its own output token and a clone for any number of defined token consumers called *sinks*. Each of these sinks can then process the tokens in different ways. This architecture adds more flexibility to field-specific transformation by creating virtual analyser chains, which share some of the chain links. For example in some scenarios, a number of fields share the first analysis step, but differ in subsequent processing. [127, p. 120]

Order-dependency of *TokenFilters* is another important aspect that needs special attention. Each step in an analyser chain relies on the output of the preceding step, if present. Thus, the order of *TokenFilters* may affect the final result of the processing. A good example is stop-word filtering. Stop-word lists typically require incoming tokens to be lower-cased in order to avoid redundancy. If an analyser chain consists of these two steps, ignoring this implicit dependence may cause unexpected results, i.e. unfiltered tokens that are still present after processing, because they appeared in upper-case. [127, p. 125f]

The variety of ways to generate and process a stream of tokens serves as further evidence that Lucene is a rich toolbox rather than just a search engine implementation. This variety of implementations covers almost any form of text preprocessing and shows the flexibility that is gained by implementing analyser chains. But given this variety it may become difficult and time consuming to design an appropriate chain for the desired search application. Of course, Lucene provides a solution to this problem. It provides a number of carefully designed built-in *Analyzer* classes that implement specific combinations of *Tokenizers* and *TokenFilters*.

The *StandardAnalyzer* is an implementation that works for most European languages out-of-the-box. It includes the *StandardFilter* to recognise high-level token types, the *LowerCaseFilter*, and the *StopFilter* to remove stop-words. But again, the main focus

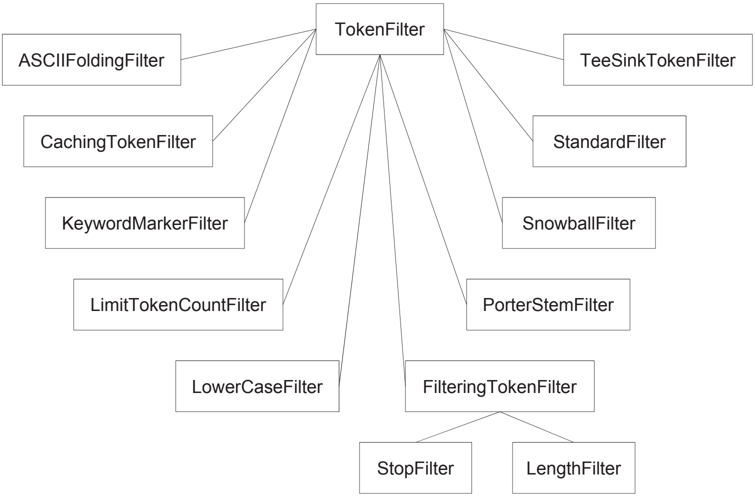


Figure 4.3: TokenFilter hierarchy in Lucene, version 3.3.

is on modularity. There are implementations like a *StopAnalyzer* that splits the text into tokens at non-character letters, lower-cases, and removes stop-words. The *KeywordAnalyzer* is useful for text elements that need to be preserved and thus it creates tokens that are equal to the incoming text. *SimpleAnalyzer* and *WhitespaceAnalyzer* both split tokens in a straightforward fashion either at white spaces or at non-character letters. But Lucene also provides custom tools to tune text transformation for better search effectiveness. [127, p. 127]

A powerful utility in the *Analyzer* hierarchy is the *PerFieldAnalyzerWrapper*. Like the name suggests, it allows the definition of field-specific analyser chains that are handled by mapping field names to analyser implementations. The main advantage of the wrapper is that it provides the potential to retain a customised text transformation set-up for the retrieval process. We will examine the details of the problem, when the query transformation concept of Lucene was introduced (see the next but one paragraph). Another important problem that affects the effectiveness of search are synonyms. The *SynonymAnalyzer* of Lucene tackles this problem by injecting tokens into the token stream. In general, there are two possibilities when approaching this problem. Either the synonyms are injected in the index or into the query. Both methods have advantages and disadvantages that have to be weighted for the desired application.

A *SynonymAnalyzer* uses the position attribute of tokens to inject synonyms at the same position as the original token. This approach ensures that the original token stream remains in its original order, which is needed to guarantee that phrase queries work as expected. Some phrasal queries may, however, become affected when stop-words are removed during indexing. Consider the query "the Wizard of Oz" and a document that contains that exact phrase. When stop-words are removed during indexing, half of the information in the phrase is lost. But that does not affect search quality much, because Lucene retains empty positions of deleted tokens. However, phrase queries are quite costly in terms of efficiency. In case parts of the phrase are not present in the document index, ranking documents in response to our example query is likely to be inefficient. This is mainly because the number of documents that match the four word phrase with wild card tokens at positions one and three is likely to be larger than the number of documents that contain the actual phrase. Lucene provides enough flexibility to solve this issue with a *ShingleFilter*. This *TokenFilter* groups two consecutive words into a single token. The resulting bi-gram tokens (note that bi-grams are at word-level here) have the same position in the token stream as the first word of the bi-gram. Obviously, this results in a larger index, but at the same time exact-phrase queries gain much faster response times. The given examples of advanced analyser capabilities show that many problems of modern search applications can be directly addressed using Lucene.

4.1.1.2 Ranking

In order to search an index created with Lucene, user-generated queries have to be processed and transformed to Lucene's internal Query objects. The *QueryParser* provides a method named *parse* that performs this operation. In order to parse a query, it needs an Analyzer object to specify how to transform the terms for matching indexed documents. Obviously, the *Analyzer* that is passed over to the *QueryParser* should match the one that was used for creating the index. In complex scenarios, the document fields might have been analysed by different *Analyzers*. Lucene does not store this information in the index. But it does provide the *PerFieldAnalyzerWrapper* that can be fed into a *QueryParser* instance using the same configuration that was used for indexing. [127, p. 79f]

Alternatively, *Query* objects can also be constructed programmatic, but it is necessary to ensure that the query terms are parsed appropriately to match the indexed terms. A straightforward Query object in Lucene is a *TermQuery*. It takes a Term object, consisting of a field to text mapping, and constructs a Lucene Query. Lots of other query types exists in Lucene: *PhraseQuery*, *WildcardQuery*, *FuzzyQuery*, *SpanQuery*, *BooleanQuery*, etc. All of them process query text in different ways to translate a natural language query into Lucene's query language. Although the details of these query types are interesting, they are not discussed here in detail – with one exception. A *BooleanQuery* allows the combination of almost all types of Lucene queries and it provides insights into the ranking model deployed in Lucene. In essence, Lucene relies on the Boolean model for document matching and uses an extended version of the TF-IDF vector space model to rank the matching documents. As a result, *BooleanQuery* objects in Lucene are containers of Boolean clauses, which can be optional (OR), required (AND), or prohibited (NOT). This translates into complex nested search structures, which are used to efficiently select documents for ranking. [127, p. 94ff]

$$rsv(d,q) = \sum_{t \in Q} tf(t_d) \cdot idf(t^2) \cdot boost(f(t_d)) \cdot coord(q,d) \cdot lNorm(f(t_d)) \cdot qNorm(q) \tag{4.1}$$

The exact definition of Lucene's scoring function [127, p. 86] is given in Equation 4.1, where the first and the second parts of the sum represent the TF-IDF weight, the third part is a static field-specific boost, the fourth component is a coordination factor, and the two last parts are normalisation factors by field length and query terms. The function $f(t_d)$ returns the field of the document d that contained the query term t. It is obvious that ranking in Lucene is based on the concept that documents consist of fields. As a result, both normalisation and boosting are field-specific and developers can influence the ranking by altering field boosts during indexing. Despite these static boosts, Lucene also allows the boosting of query clauses. This feature only affects the document ranking if queries contain more than a single clause. The coordination factor $coord(d)$ is used to boost documents that contain more than a single query term. It can be compared to the normalisation factor λ in linear language modelling (see Section 3.2.4.4), because it leads to an AND-like boost for respective documents.

4.1.1.3 Summary & Outlook

In order to summarise strengths and limitations of Lucene, we briefly review its key features and relate them to other open-source search engines. The most powerful component of Lucene is the flexible framework for term normalisation during indexing. Even though the advantage of applying certain normalisation techniques might be debatable, none of the other search engine toolkits that were part of the present high-level analysis are comparably flexible and yet powerful in out-of-the-box scenarios. However, Lucene's indexing structure, which is three-dimensional in the sense that it covers documents, fields, and terms, does not exhibit such a level of flexibility.

This is a major limitation from a scientific perspective, because the lack of flexibility in the index structure complicates the implementation of new ranking models. In addition to that, over the past decade Lucene's index format was primarily optimised for efficiency in terms of speed. But over time, new approaches to index compression have been developed. These methods have not yet been taken into account for the index format of Lucene. The reason for that might be the complex structure of the code base for indexing, which grew dramatically over time due to efficiency optimisation. This limitation of Lucene is currently under investigation by the development community. For the upcoming Lucene version 4.0 a complete re-design of the code base for indexing is under development. The main purpose of this development is to provide a new flexible index structure that allows the incorporation of different ranking models by accumulating according statistics at collection, document, and term level. At the same time, the developers focus on including the latest index compression techniques that make use of advanced capabilities of modern computing hardware. A major limitation of Lucene up to version 3.x would then be obsoleted.

Presuming the developer community manages to create a concept of similar flexibility and generality for building index structures like the term transformation toolkit, we could consider Lucene the Swiss Army Knife of information retrieval technology. Given that it would have a selection of different state-of-the-art retrieval models incorporated, until now a striking feature of Terrier only, it is likely to attract the IR research community even more than today. Receiving contributions from the research community to incorporate fresh ideas quickly, in combination with the powerful text transformation framework that Lucene already provides, will certainly be beneficial for both the scientific and the commercial search community. But at present, it is not yet clear if the intended goals for Lucene version 4 can be achieved. Another critical aspect worth noting is that if the developer community succeeds, the solution may still raise new issues, for example degraded efficiency due to the number of possible index formats. A further problem could be trying to keep the core of Lucene as slender as possible. However, the Lucene developer community has mastered all big issues in the past, which indicates the potential of Lucene-based search technology in the future.

4.1.2 Terrier Platform

Another open-source software framework for textual IR is being developed at the Glasgow University since 2000 [135]. The motivation was to provide a framework for rapid development of large-scale IR applications. Consequently, it was named Terabyte Retriever platform (of which Terrier is an acronym). An important aspect for the development of the framework was the lack of a standard software framework as a test bed for IR research and experimentation. In 2004, the first release of the software was published under the MPL license. Until then the most influential institutions in IR research usually developed their own models separately. Comparison of IR methods and architectures was restricted to evaluation campaigns like TREC and still raised concerns regarding good comparability due to the complexity of modern IR systems. Thus, the most important aspect of the motivation for the development of Terrier was the need for a general software test bed that allows the comparison and also the combination of many different IR models to increase effectiveness and efficiency of large-scale textual search applications. Next, the Terrier framework which is implemented in Java, is explained in more detail. Therefore the previous layout of the discussion is followed by considering the main architecture and important concepts of the software implementation.

4.1.2.1 Text Processing & Indexing

The process of parsing documents and creating a document index covers four modular stages in Terrier. Its architecture allows the usage of different plugins for each part of the process and thus ensures flexibility. The four stages of indexing are collection handling, document processing and parsing, term processing and transformation, and lastly the creation of index data structures. Since Terrier is intended as a shared, common research platform, it also contains plugins for processing standard TREC collections out-of-the-box. This contribution is two-fold, because researchers can focus on the development of new ideas and methods and at the same time the common chain of plugins for processing standard collections ensures reproducible experiments. For experiments that introduce a new collection, the first step in Terrier is to develop a new collection plugin. In the next step, documents have to be processed in order to extract textual information. Terrier supports several common document types like HTML, document formats of the Microsoft Office product family, and documents stored in PDF. Terms are modelled using three major attributes: the actual textual String, the position of the term in the document, and the fields, in which the term occurs, given that the documents are structured. Obviously, there is a significant overlap in how open-source search engines model general documents and terms. However, academic search engines like Terrier or Lemur model structured documents different than Lucene.

Transforming text into index terms is realised with *Term Pipeline* implementations in Terrier. Two standard *Term Pipelines* are provided by default: two variants of Porter's stemming algorithm, and a stop-word filtering *Pipeline* is used to remove common words. The last step of the indexing process is controlled by Indexer implementations that store the resulting data structures on disk. Here, Terrier provides a *BlockIndexer* and a *BasicIndexer*. In order to use term positions for retrieval, the *BlockIndexer* stores positions in blocks, where the size of the block defines the position accuracy. Every created index is stored in four data structures. The *Lexicon* stores terms and corresponding global information like document and term frequencies. The *Inverted Index* contains the posting lists for the terms covering document identifiers and term frequencies. Document numbers, document identifiers, and document length are stored in the *Document Index*. Finally, the *Direct Index* is used to store terms and term frequencies to facilitate easy and flexible query expansion during retrieval. In order to store the index data structures efficiently, the redundant information in the *Direct Index* and the *Inverted Index* are compressed using state-of-the-art encoding algorithms.

4.1.2.2 Ranking & Feedback

Terrier also supports flexibility in the retrieval stage. The plugin concept is adopted for all components that contribute to the final ranking of documents. Terrier features different state-of-the-art weighting models and it allows custom term and document scoring. For cross-comparisons of retrieval models, Terrier incorporates TF-IDF, BM25, Language Models and the Divergence from Randomness Framework for parameter-free probabilistic ranking. The latter model was developed at Glasgow University (see Section 3.2.4.5) and thus most of the weighting models in Terrier are implemented in this probabilistic framework.

Altering scores on document or term level is achieved by means of *TermScoreModifiers* or *DocumentScoreModifiers*. Tuning the score of particular terms is useful for structured retrieval to ensure that query terms appear in predefined fields. A *TermInFieldModifier* penalises documents that contain query terms in all but the desired document fields. Similar to Lucene, Terrier also supports static field boosting by means of the *FieldScoreModifier*. Another option to incorporate query-independent prior information is the *DocumentScoreModifier*. For example, in the scenario of searching web pages it could be used to include evidence from hyperlink structure analysis. In addition to that, *DocumentScoreModifiers* can also be used to alter the weighting model based on prior evidence. A final stage in Terrier's ranking model is post-filtering. Reducing the number of documents returned from a single web page in order to increase result diversity is a practical example. Another type of post-processing is query expansion in the sense of altering terms of the original query.

In order to support pseudo-relevance feedback mechanisms, Terrier uses its *Direct Index* data structure to extract the most informative terms from documents ranked at top positions. These related terms are then added to the original query. Terrier relies on its flexible term weighting model to re-weight the query, providing a richer set of retrieved documents. The success of automatic feedback models depends on various factors like query difficulty and the number of documents or terms that are used to expand the query. Again, the main design purpose of Terrier is achieved by supplementing feedback models that are based on the Divergence from Randomness framework with traditional models like Rocchio's feedback (see Section 3.3.1). This enables IR researches to design experiments that compare different automatic feedback algorithms. Furthermore, Terrier supports low-cost selective methods for query expansion. These facts show that the flexible term weighting model in Terrier enables IR researchers to relate new methods to a wide selection of effective state-of-the-art weighting and ranking models.

4.1.2.3 Evaluation

Comparative assessment of retrieval systems also requires test collections and metrics that measure relevance of documents. Evaluation campaigns like TREC provide test collections and metrics in annual cycles. In order to allow these experiments to be re-used, Terrier also provides a small framework that takes result lists and relevance assessments as input to calculate commonly used performance measures. Most of these metrics are based on precision (see Sections 2.3.1 to 2.3.4). Using Terrier's evaluation package, IR researchers are able to assess new methods without having to look at too many details. In combination with the rich set of incorporated state-of-the-art IR models, Terrier has become an important platform for IR research and practice. The developers claim that Terrier was downloaded several tens of thousands times and that it is used both for research and practical applications.

Terrier's major advantage is its flexible design. In spite of Lucene, which is another highly flexible open-source search library, Terrier focuses on flexible ranking and weighting. A minor limitation in comparison to Lucene lies in its term transformation concept. Although the *TermPipelines* allow flexible term transformation implementations, only a few algorithms are supported by default. A reasonable explanation is the focus and expertise of the research group at Glasgow University, which lies in IR research rather than in computational linguistics. However, being aware of the strengths of Lucene, Terrier, and Lemur, we can conclude that a clever combination of these platforms in a meta-framework may allow the covering of particular weaknesses and the potential multiplication of the advantages in a symbiotic environment.

4.1.3 Lemur Project

The *Lemur Project* is an open and collaborative work between the two US institutions University of Massachusetts, Amherst and Carnegie Mellon University. The project was initiated in 2000 and covers several contributions from students and staff members of both universities. All of the subsequent information was gathered from the project website (see reference in Table 4.1). At the time of writing the joint project contains five major components: Indri, Lemur, Galago, the Query Log Toolbar, and ClueWeb09. Instead of focusing on the details of the IR system implementation in Lemur, these five contributions are described briefly. The reason for setting a close examination of Lemur aside is that its development has been suspended recently and therefore it has lost its relevance for this dissertation.

Indri is a text search engine that combines inference networks with language modelling. It is primarily intended for academic research purposes and it supports indexing of large-scale text collections. Language models can be created for documents, queries, or sub-collections using the Indri API. Similar to other academic search engine projects, like Terrier or Xapian, Indri is complemented with API's for common programming languages, e.g. Java or PHP. By means of these interfaces it can be easily adopted for most search applications. Querying an index is achieved by using the Indri Query Language. It is a structured language that allows arbitrary complex queries. It originates from the widely-used InQuery query language. Based on the constant development in the past, it can be assumed that the query language is able to fulfil most of the requirements of modern search systems. Indri's query language supports different operators on term level. In addition to that, it is possible to use the belief operators presented in Section 3.2.4.6. Incorporating prior information on documents into the weighting algorithm is another feature of Indri. In contrast to other retrieval toolkits these document priors can be used in the query language, i.e. incorporating priors can be triggered dynamically depending on the contents of a query.

The Lemur toolkit is a collection of technologies that are useful for creating search engine applications. The software package includes tools for cross-lingual retrieval as well as document summarisation, filtering, and categorisation. It supplies several indexing formats that are optimised for the efficient processing of various collection sizes. The Lemur toolkit offers a number of retrieval models, e.g. Indri's inference network implementation, Okapi (BM25), TF-IDF, and others. It also supports many common document formats, like HTML or PDF. Due to scalability issues in the general architecture the final version of the Lemur toolkit was released in 2010.

A component that was added to the Lemur project recently is the Galago toolkit, developed at the University of Massachusetts, Amherst. In contrast to Indri and Lemur, Galago is intended for research purposes, because it focuses on the experimental features of a search engine. The main design goals for Galago are flexibility and scalability. Flexibility is achieved by means of customisable indexing and retrieval processes. In order to develop a scalable framework, a technique called *TupleFlow* was developed and implemented in Galago. *TupleFlow* is a method for distributed computation that extends *MapReduce*. In the *MapReduce* model a master node distributes subtasks that are partitions of the original problem to worker nodes, by transforming input items into key-value pairs. The worker nodes process these subtasks and pass the results back to the master, which aggregates the results by key and combines them to solve the original problem. *TupleFlow* allows the use of arbitrary data types (or tuples) to be exchanged between several distribution and aggregation stages. In addition, each of the stages can have arbitrary numbers of inputs and outputs. This flexible distribution set-up can be represented as an acyclic graph, where nodes are stages of the computation and edges represent the data flow. The data flow in such an acyclic graph accounts for the dependencies during the distributed computation.

Scalability and flexibility are conflicting requirements when designing a search engine toolkit. Flexibility means that a selection of retrieval models is available to be able to compare their effectiveness in different search scenarios. But different ranking functions require different information about terms and other features, which have to be included in the corresponding index representation. In order to create a retrieval toolkit that scales well both during indexing and retrieval, the index formats are typically optimised for efficiency. However, optimising several index formats for scalability is a complex problem. Galago tackles this problem by offering two customisation strategies. First, it allows the implementation of any custom ranking model and uses *TupleFlow* to obtain a corresponding index format that is very efficient for retrieval. Once such an index is built, the strategy for querying the index cannot be changed due to optimisation (represented in the *TupleFlow* graph). The second approach sacrifices efficiency for flexibility and makes use of a rich structured query language that is termed *Galago Query Language*. Galago's structured query language is a descendant of the query languages used in Indri and InQuery. A major limitation of Galago is that the current open-source implementation only supports single machines with multiple processors. For clusters of multiple machines there is no software available that makes use of the *TupleFlow* model, which makes it hard to compare it to other extensions of *MapReduce*.

Since 2005 the *Query Log Toolbar* is another important part of the Lemur Project. The Query Log Toolbar captures user behaviour and is intended to promote research in information seeking behaviour and related disciplines. It consists of two major components: a client that captures input data from a user, and a server that collects this input and provides access to the collected data via programming interfaces. Currently, clients are available as add-ons for commonly used web browsers, namely Firefox and Internet Explorer. Since privacy is a primary concern when dealing with user data, the client provides a simple blacklist filtering approach to make personal data anonymous before it is uploaded to the query log server. In order to allow rich analysis of the query log data, the toolbar can be configured to capture user queries as well as results from a set of predefined search engines. The Query Log Server uses a MySQL database and

Java Servlets to store the data it receives from registered clients. There are two options available to install the Query Log Server on a dedicated machine. Depending on existent services on the machine, one would either use the stand-alone server application that includes a container for the Java Servlets or one would install the application in an existing Servlet container. The flexible configuration of the toolbar enables researchers to design arbitrary experiments to study user behaviour in search scenarios. For that reason the Query Log Toolbar can be considered as the most unique feature of the Lemur Project.

A final component in the Lemur collection of information retrieval tools is the ClueWeb09 dataset. It contains about one billion web pages covering ten languages that were crawled in early 2009 by the Language Technologies Institute of Carnegie Mellon University. The dataset contains the contents of the web pages and the web graph, which includes URL's and outgoing links from the web pages. Web pages are stored in the WARC format that defines a standard way to store payload and control information from internet protocols like HTTP, DNS, and FTP. Due to the large size of the collection, the dataset is distributed in two categories. Category A refers to the full data set and category B includes about 50 million English-only web pages. The compressed storage size of the dataset is about five TB for the web pages and about 130 GB for the web graph (both values are for the entire dataset, i.e. category A). In the two years of its existence, the ClueWeb09 collection has already been used in several tracks at TREC, e.g. the Crowdsourcing Track at TREC 2011, or the Million Query Track at TREC 2009. This shows both the broad acceptance and the need for a research test collection of this type and scale. Another aspect that supports this claim is the variety of additional information that is available for the dataset. Carnegie Mellon University supplied a list of duplicate entries, a list of URL redirects for the category B subset, and their calculated PageRank scores for the entire collection. Other supplementary contributions from various institutions worldwide include spam scores, anchor text, and an anchor log. The latter can be used for query reformulation based on the assumption that anchor text for a link to a document may contain rich information on the target web page.

4.1.4 Overview on Further IR Frameworks

MG4J (an acronym from *managing gigabytes for Java*) is a toolkit for indexing large text collections at web scale that is being developed and maintained at the University of Milano [20]. It allows distributed indexing and supports querying multiple indexes by interleaving the results. Since its initial publication in 2005, it has been used in several research experiments ranging from ad-hoc retrieval to finding geographic locations and biomedical search. Flexible distributed indexing is a key feature of MG4J and thus it is predominantly adopted for experimental research on large data collections.

The Minion framework is dicussed next. It was developed at Sun Labs (now Oracle Labs) as a part of the Portal Server software system from early 2008 until the beginning of 2010. Minion supports standard document retrieval as well as Boolean, relational, and fuzzy querying. The architecture of the software was designed to be highly flexible. Unfortunately, no statistics were found on the usage of the software package. Deriving usage statistics from the Portal Server could be inaccurate, because Minion is just one of many components in that software framework. In general, Minion is very similar to Lucene (see Section 4.1.1), but there are two aspects that distinguish Minion from most other software implementations. First, Minion generates morphological variations at query time using its integrated morphological framework named LiteMorph. It is a lightweight collection of rules similar to those in rule-based stemmers. The purpose of incorporating generative morphological variance at query time is to simulate the tendency of users to over-generate their queries without losing any information during indexing. A second unique feature of Minion is that it provides ready-to-use methods for document clustering and classification. The latter can be done automatically or by programmatic definition of document sets, e.g. a specific result set. Besides the traditional ranking models TF-IDF and BM25 Minion also features a patented *Relaxation Ranking* algorithm.

The Sphinx search engine (an acronym from Sql Phrase INdeX) is an open-source implementation that serves as retrieval back-end for many web applications. Craigslist, Tumblr.com, GuteFrage.net, and CouchSurfing.org are just a few popular examples that serve large user communities. A key feature of the engine is performance in terms of efficiency, which includes good scalability as well as distributed indexing and searching. Based on the information from the Sphinx website, it can be concluded that the main purpose of the development is to commercially exploit the underlying technology in the context of large-scale web applications. Sphinx allows plain file indexing as well as SQL database indexing and it supports a wide range of RDMBS. It provides API's for standard programming languages like C and Java, but also for many other languages that are commonly used in web applications. Besides straightforward Boolean matching, Sphinx uses a phrase weighting algorithm that is based on the longest common subsequence (LCS). It also supports probabilistic ranking using the BM25 weighting scheme. The system provides five different ranking algorithms in total, which are combinations of Boolean, phrase and statistical ranking. To handle morphological variants of words, Sphinx includes the Snowball framework for stemming, which provides rules for many common languages. In addition to full text search, the application also supports selecting and ranking of non-textual data.

Another fully fledged search engine implementation is Wumpus from the University of Waterloo in Canada. Its primary objective is to study problems that arise during indexing of dynamic text collections, especially in the context of multi-user environments. A specific scenario is file system search, where the number of document updates can be expected to be larger than the number of user queries. Consequently, the implementation was carefully designed to allow maximum scalability. The program code is written in C and it uses a file system change notification package for Linux named *fschange*, which was developed by the author of Wumpus to overcome some shortcomings of other notification mechanisms. As previously noted, Wumpus proved its great scalability in a scientific comparison with Zettair and Lemur. Since the emphasis on efficiency in the defined scenario is the major advantage of the engine, it is not as flexible as other toolkits that allow the adjustment of many components of the entire system. Although this is not a limitation per se, it somehow restricts the adoption in search applications. The original implementation featured a ranking algorithm that is based on the vector space model. Later on, the widely-used BM25 algorithm was added for convenience. Wumpus was mainly used for research on large-scale text collections, e.g. in TREC's Terabyte task. There is no indication that it was used for other scenarios like commercial search applications.

Given the fact that it has been under development since 2001, and considering its long history before that period, Xapian is the most advanced framework in the present discussion. It was derived from Open Muscat, a commercial retrieval system developed by Martin Porter and his colleagues in the 1980s. The main objective for the development of Xapian (and its predecessor Muscat) was to create an efficient software implementation of the probabilistic IR model. Thus, it implements the widely adopted BM25 weighting scheme and allows arbitrary complex Boolean queries. A basic Boolean ranking method is also included in the system. Another feature of Xapian is its ability to rank terms, which can be used to automatically expand a query. Of course most of the recent search engine toolkits also support that kind of functionality. The developers of Xapian also offer a ready-to-use search engine application called Omega. Similar strategies are implemented for Lucene, Terrier, and Sphinx, which suggests that this is a convenient method to commercialise open-source information retrieval technology. Although Xapian is written in C++, which naturally requires more work to support platform independence than Java, it can be installed on machines operating on Unix, Windows, and OS/2. An important feature for commercial use is the support for data imports from various RDBMS implementations. Similar to most other open-source toolkits of its family, it also supports phrase search and faceted (or categorised) search. Xapian is used in a number of search and news websites, in web-based applications, and software frameworks. It also serves as the search engine in a few desktop applications, the OLPC (one laptop per child) project is the most prominent example.

A final implementation considered here, is Zettair, from the search engine group at RMIT University in Melbourne, Australia. Zettair is a fast text search engine that was formerly known as Lucy. It is designed to efficiently index and search HTML collections. Its main purpose is to provide a simple but fast command-line search application. The application is mainly used in research experiments dealing with large web collections. Zettair supports basic Boolean queries as well as phrase queries. Just like almost all of the other listed projects, it also ranks documents based on the probabilistic retrieval model and implements the BM25 weighting model. In addition to that the Dirichlet language model is also included. Although the software project has existed for a number of years, it has seen only little use outside the IR research community in Australia.

4.2 Excursus: Natural Language Processing Toolkits

Natural language processing is a field of research that is closely related to information retrieval. In particular, the overlap between the two subjects lies in morphological operations on low-level techniques like stemming, synonym extraction, and spelling correction. More complex operations combine methodologies from both fields at a higher level for Question Answering (QA), where an automatic system is required to answer a natural language question. Further details these systems will not be discussed, because current research in QA covers many topics that are not relevant for this work. However, some low-level techniques are potentially useful for information retrieval applications, as pointed out in Section 4.1.1. Here, a review of software frameworks that are frequently used in combination with search applications is provided.

A collection of tools for NLP is provided by the Stanford Natural Language Processing group[16]. They offer several Java-based software tools for parsing, POS tagging, named entity recognition, and text categorisation under the full GPL license. Along with the API's of the *Stanford Parser*, the *Stanford POS Tagger*, and the *Stanford Named Entity*

---

[16] `http://nlp.stanford.edu/software/index.shtml`, retrieved on March 1, 2012

Recognizer a number of trained models for several European languages, for Arabic, and for Chinese are also provided. This allows easy integration into applications as well as extending the provided models, or training completely new models.

The Stanford Parser is based on probabilistic modelling, i.e. it is based on prior knowledge from a collection of manually annotated sentences. The software implementation of the Stanford Parser supplies several state-of-the-art approaches: a basic lexicalised dependency parser, a lexicalised probabilistic context-free grammar (PCFG) parser, and a highly optimised PCFG parser. The output of the parser adheres to the Stanford Dependencies format, or alternatively, generates phrase structure trees. The Stanford POS tagger is based on a log-linear implementation and is distributed with several trained models. The Stanford Named Entity Tagger is only available for English and uses a conditional random field (CRF) implementation. Supplied models cover different classes of entities, with the three most popular types being location, person, and organisation, and in addition: miscellaneous, time, money, percent, and date.

The Stanford Classifier implementation is based on maximum entropy modelling. Here, no trained models are provided. In addition to the separate API's mentioned above, a suite of NLP tools is provided as the *Stanford CoreNLP* software package. It integrates all of Stanford's NLP group's tools for English and hence provides the basics needed for higher level language understanding applications. The CoreNLP package also includes a new deterministic coreference annotator. Coreference resolution is the process of identifying multiple expressions in natural language that refer to a single object or individual. It is a common problem in automatic analysis of dialogues or discourses. UIMA wrappers (see next but one paragraph) were contributed from outside of Stanford's NLP group for most of these briefly presented tools. This indicates that both the tools of Stanford and the UIMA concept are widely used in research and enterprise applications.

Another software toolkit that has to be considered is OpenNLP[17]. It is a collection

---

[17] `http://incubator.apache.org/`, retrieved on March 1, 2012

of open-source projects for natural language processing rather than a software framework in the traditional sense. At the time of writing, OpenNLP has also entered the Apache Software Foundation as an Incubator project. The software project is written in Java and was first released 2003. Over time OpenNLP has matured into a powerful toolkit that supports most of the common tasks in NLP, like sentence segmentation, POS tagging, named entity extraction, and coreference resolution. In addition, it also supports basic machine learning technology based on maximum entropy and perceptrons. The main goals of the project are twofold. First, it is intended to provide basic state-of-the-art approaches of NLP to solve more complex problems for text processing applications. And secondly, OpenNLP is used to develop and distribute models that are ready to use for most common languages. Similar to evaluation in IR, where re-using standard test collections and reproduction of experiments is a central issue, standard models are needed as baselines for assessing the quality of new approaches in NLP research. More recently, the developers added support to the Apache UIMA framework by wrapping OpenNLP components in UIMA Analysis Engines.

Apache UIMA[18] stands for *Unstructured Information Management Architecture* and was initially developed by IBM before it was made publicly available through ASF in 2006. As its name suggests UIMA, is a framework for the analysis of unstructured textual information. Its general architecture was accepted as industry standard. IBM's computer named *Watson* is probably the most widely known application that is based on UIMA. It competed in the quiz show Jeopardy! against two former champions in early 2011 and defeated both of them. In essence, UIMA allows to define analysis pipelines to process unstructured information that are called *Analysis Engines*. An analysis pipeline consists of *Annotator Components*, which consist of either a single, or an aggregation of components. Some text analysis components are directly provided with UIMA. Others are wrappers for projects such as OpenNLP. Two different models are supported to design a complete pipeline for an application. The *Collection Processing Manager* is an old model that is only kept for backwards compatibility. In that model a concept named *Collection Processing Engine* represents a single-

---

[18] `http://uima.apache.org/`, retrieved on March 1, 2012

threaded processing pipeline that starts with a *Collection Reader* and ends with a *CAS Consumer*. This linear structure impedes good scalability. UIMA *Asynchronous Scaleout* (or UIMA AS) fixes the problem and allows flexible analysis pipelines by means of a flow controller and a CAS multiplier. Additionally, UIMA AS supports a client-server architecture in order to allow clients to asynchronously access *Analysis Engines*. Again, a collection reader serves as the starting point of the processing. But instead of a fixed and single-threaded layout the *Analysis Engine* is accessed through a service wrapper that can be replicated. This allows for both more flexibility and scalability. UIMA has its own archiving format named *Processing Engine Archive* (or PEAR) that allows straightforward re-use of existing *Analysis Engines* in other UIMA applications.

In line with UIMA is GATE[19] (an acronym for General Architecture for Text Engineering), which is a software architecture and collection of toolkits developed by researchers at the University of Sheffield. The first version of GATE was published in 1995 and at the time of writing the latest release is version 6 published at the end of 2010. Similar to UIMA, it is written Java and is an empty framework, i.e. an architecture for processing text. GATE features a component-based architecture that contains three types of resources. *Language resources* are lexicons, text corpora, ontologies, or the like. Algorithmic processes like parsers, generators, and models are represented as *Processing resources*. Editing components that require visualisation are called *Visual resources*.

In GATE, all integrated resources are combined in CREOLE (Collection of Reusable Objects for Language Engineering). Several CREOLE resources are already built into GATE. Language Resources that are supported cover many document formats like mark-up languages (HTML, SGML, XML, etc.), office document formats covering Microsoft products and OpenOffice as well as PDF, and also UIMA CAS documents. One supported Processing resource is ANNIE (A Nearly-New Information Extraction System), which provides basic NLP tasks like sentence segmentation, tokenisa-

---

[19] `http://gate.ac.uk/`, retrieved on March 1, 2012

tion, POS tagging, and others. ANNIE is based on finite state machines and relies on the Java Annotation Patterns Engine (JAPE). GATE also contains various gazetteers, which are lists of named entities like geographic locations, currencies, etc. Further built-in processing resources include ontologies, machine learning resources, alignment tools, and several parsers and taggers. In general, GATE uses a pipeline structure that is similar to UIMA. Relying on its powerful component architecture, GATE also includes widely used NLP resources like OpenNLP, WordNet, and the Stanford collection of NLP tools.

UIMA and GATE are widely used in the NLP community. Nonetheless, other frameworks do exist that should not go unnoticed. The Natural Language Toolkit[20] (NLTK) was initially developed at the University of Pennsylvania and is distributed under the Apache Licence. It is implemented in Python and intended for teaching and research purposes. The toolkit is organised in several modules that cover different approaches for particular NLP tasks. The toolkit provides tokenizers, stemmers, taggers, chunkers, parsers, corpus readers and many more. In addition to that, NLTK also supplies document readers for a number of test corpora, several trained models, and grammars that can be used for testing. A limitation of the software toolkit lies in the Python language, which is an interpreted programming language. Thus, it is not the most efficient way to solve NLP tasks that process large amounts of textual data.

A project similar to NLTK is ClearTK[21], which is being developed at the University of Colorado, Boulder. ClearTK is a framework built on top of UIMA and thus, it is also implemented in Java. Its main purpose is to provide a framework to tackle NLP problems with common machine learning techniques. The main feature of ClearTK is its common interface to popular open-source machine learning (ML) toolkits like SVMlight[22], LIBSVM[23], OpenNLP Maximum Entropy, and Mallet[24]. In addition it

---

[20] `http://www.nltk.org/`, retrieved on March 1, 2012

[21] `http://code.google.com/p/cleartk/`, retrieved on March 1, 2012

[22] `http://svmlight.joachims.org/`, retrieved on March 1, 2012

[23] `http://www.csie.ntu.edu.tw/~{}cjlin/libsvm/`, retrieved on March 1, 2012

[24] `http://mallet.cs.umass.edu/`, retrieved on March 1, 2012

provides wrappers for NLP tools like the Stanford CoreNLP tools or the OpenNLP library. The combination of NLP and ML techniques is supplemented with collection readers for many common collections like Penn Treebank, CoNLL 2003, or TimeBank.

4.3 Summary and Directions

In this section a review of a selection of open-source software toolkits for information retrieval and natural language processing was provided. Although these frameworks were carefully selected, the list is not exhaustive. More popular toolkits were studied in more detail to be able to identify key differences between them. In addition, the focus of this investigation was on the needs of a researcher, being aware that a software developer, who is developing a particular search application would not consider all these aspects to be relevant for his work. In general, it has been found that most open-source IR projects consisted of at least two components. A search engine API or toolkit, which abstracts the complex process of the general model presented in the beginning of Chapter 3, is supplemented with an application framework that covers different search scenarios.

Although Lemur and Terrier are being developed in academic institutions, they do adhere to the two-fold API and application set-up. The Lemur Project is a successor of the InQuery system, which was one of the first experimental IR systems in the world. Given its long history and the fact that new state-of-the-art approaches were continuously integrated, Lemur has a strong reputation in the research community. Our investigation showed that Lemur incorporates a few IR models of the probabilistic family, ranging from Okapi (BM25) to Language Modelling and to the Inference Network Model. Another positive aspect is the Querylog Toolbar, which allows the collection of implicit user relevance feedback in web or browser-based search scenarios. The IR community also gives acclaim to the Lemur team for designing and collecting the ClueWeb09 test set of web pages for IR research. An impressive list of additional contributions regarding the ClueWeb09 collection shows the success of the community efforts. Due to the limitations in the architecture that impede good scalability of high-level search applications that deal with large data sets, the development of the Lemur Toolkit has been discontinued recently. This is a major drawback, because it is not clear whether there will be a replacement based on the Galago framework or not. Altogether the Lemur Project is an important collection of resources for IR research. But in comparison with other frameworks like Terrier or Lucene it has a limitation in terms of flexibility.

The Terrier platform is another academic framework. A central goal for the development of Terrier was to provide a standard framework to foster IR research. This aim was motivated by the fact that most of the leading institutions in IR research developed and used their own software framework for experimental evaluation. These frameworks rarely implemented more than a single IR model. Instead, they were usually based on a particular IR approach that was developed and promoted (in terms of scientific development and publication) at the respective institutions. In order to ensure maximum flexibility, Terrier implements a plugin concept for indexing and retrieval. Terrier supports various document formats and supplies parsers for many standard test collections used at TREC. Following the main design goal, it facilitates a flexible ranking approach that allows the selection of specific ranking models for an application. This unique feature is particularly appealing from a scientific perspective, because it also allows the comparison of fundamentally different IR scoring and ranking models in a common software test bed.

Its basic framework for the implementation of pseudo-relevance feedback algorithms is another aspect that should not go unnoticed. Similar to other search engine toolkits, there is a single model that dominates the software implementation. In case of the Terrier platform, this is the Divergence from Randomness model (see Section 3.2.4.5). Roughly half of all the ranking functions that are supplied in Terrier are implemented in the DFR framework. But in the case of DFR, this is not necessarily a negative aspect, because other ranking models like BM25 can be formulated in DFR. In fact, Terrier provides an implementation of BM25 in the DFR framework.

An evaluation package that contains a number of algorithms to compute state-of-the-art effectiveness measures completes the platform. Bearing in mind that the flexibility of a software implementation is usually sacrificed for efficiency in the implementation, this would result in questioning the scalability of the framework. Since scalability of the platform was another key aspect of the development, this is not problematic. In fact, Terrier supports parallel processing via Apache Hadoop, which implements MapReduce. There is, however, a limitation in the text transformation procedure, which does not support many different approaches. Summing up, Terrier is a very flexible framework for an easy entry into IR research without having to deal with details of IR approaches.

Apache Lucene is an open-source IR library maintained at ASF. From the toolkits that have been reviewed here, it has the largest developer and user communities, a fact that puts it clearly ahead of other frameworks, at least in terms of manpower. Over time a number of Lucene-related projects were initiated at ASF. A brief overview on these projects was given here. The variety of frameworks and applications that can either be coupled with, or are built on top of, Lucene proves the popularity of the project. The present investigation also demonstrated that Lucene features the most powerful and flexible text transformation framework. Due to the number of industry applications, it contains virtually every feature that a modern search application should support. In addition to that, Lucene uses a highly optimised index format. A great advantage of Lucene is its flexibility and its effective default configuration. It was also pointed out that its major limitation lies in the single scoring mechanism.

A close look inside current developments for the upcoming version 4.0 revealed that a major redesign of the indexing and scoring architectures is in progress. Flexible indexing with state-of-the-art index compression and flexible scoring with support for most of the state-of-the-art IR models would eliminate the limitations of Lucene. Only an evaluation package would be missing from the perspective of an IR researcher.

In summary, it can be concluded that the retrieval toolkits Terrier, Lucene, and Lemur all have specific strengths and weaknesses. When designing a scientific gedankenexperiment that takes into account all key parts of modern retrieval systems, one would expect a flexible framework which combines the text transformation capabilities of Lucene with the choice of scoring algorithms in Terrier, and the evaluation resources provided in the Lemur Project. Currently, the work of Lucene's developer community shows a clear intention to fulfil this need in the near future.

The architecture and flexibility of a modern search engine is only one side of the problem. More advanced techniques combine natural language understanding approaches to improve the subjective effectiveness of search engines. But more than that is what the IR community can adapt from NLP research: the brief review of NLP frameworks showed that frameworks like UIMA and GATE effectively manage baselines for the evaluation of new approaches. More sustainable evaluation will surely help to advance IR research. Bearing in mind that reproducible experimentation is still a key issue in IR evaluation, one could formulate a long term goal for IR research. The community would benefit from a framework that allows comparison across test collections, IR models, and search engine implementations. Formulating the vision of a common software framework that receives contributions from all researchers in the field could be the first step in the right direction. Lucene has the potential to realise such an ambitious goal. In order to facilitate this process, it might be useful to couple it with evaluation frameworks like TREC, CLEF, etc. However, whether such a vision is meaningful to IR research or not remains debatable. And even if it is, other possibilities may open up to tackle the central questions in sustainable IR evaluation.

5 Xtrieval Framework

Since 2007 the Xtrieval framework has been under development by the Chemnitz retrieval group which was formed in late 2005. The mainspring for the creation of Xtrieval was to design, conduct, and analyse IR evaluation experiments. This chapter covers two major areas. First, the design and implementation of Xtrieval are presented at an abstract level. A more detailed view demonstrates the combination of the most important state-of-the-art components of an IR system. The section on the architecture of the framework is complemented with a discussion of the prospect of further possible refinements that will only be outlined in this work.

The second part of the present chapter is intended to show the wide applicability and flexibility of Xtrieval. Note that the framework is being developed over several years. For that reason a number of publications that deal with different aspects of IR evaluation using Xtrieval, are referenced and summarised throughout the chapter. These efforts were either authored or mentored and supervised by the author of this dissertation. This chapter concludes with a summary of the observations from several empirical evaluations and a discussion on the limitations of the current evaluation methodology and possible ways to overcome these. The latter topic is then explored further in the next chapter.

5.1 Motivation

In order to describe the main purpose of the framework, it is necessary to summarise the initial situation. Starting with a first participation in an international IR evaluation campaign in 2006, namely CLEF, a task-oriented solution was developed based on Apache Lucene. It was the most flexible IR library that was publicly available at that time. A graphical user interface (GUI) was developed in order to allow easy configuration of the system. The user interface also featured a component for evaluation to load relevance assessments and visualise precision-recall graphs for the comparison of different system configurations.

The limitations in this first implementation soon became apparent. A major problem was the lack of a logical separation between the presentation in the graphical user interface and the actual programs for processing experiments. From the scientific perspective, another drawback was the sole reliance on Lucene as the core retrieval engine. Refer to [192, p. 28f] for a more detailed analysis of these limitations in this first prototype. The analysis led to the formulation of specific design goals for Xtrieval (see also [108] and [192, p. 29ff] for a detailed explanation). Other use cases and new search tasks in IR evaluation resulted in further enhancement of the individual goals. The following list summarises these goals:

- *Abstract Document Representation for Indexing*
  During indexing the documents are transformed into data structures that are stored in the file system. This process involves parsing the document collection that is used for the experimental evaluation. In order to keep the work of writing collection parsers as low as possible, an abstract *DataCollection* serves as template here. Due to many different types of document collections this abstract definition serves as flexible interface for the handling of various document types and structures. A class named *SimpleDataCollection* implements this abstract class in Xtrieval. It can be used for many structured document formats like

XML. Researchers that use the *SimpleDataCollection* only define a scheme by how they want the document structure to be represented in the index.

- *Interchangeable Retrieval API's*
  It was pointed out before that the first prototype was solely based on Lucene. Other toolkits like those discussed in Chapter 4 are based on the latest IR models. However, weighting and ranking is only one of the many elements of modern IR systems. Search effectiveness also depends on efficient text transformation models. In order to allow comparisons on text transformation level, stemming algorithms less popular than the Snowball implementation by Porter were also integrated. As Xtrieval was used for various cross-language evaluation tasks, more language-specific stemmers have been implemented into the system.

- *Simple but Flexible System Configuration*
  In order to support simple configuration, the system has been designed to allow changes of parameter values during runtime. The effect of this feature is twofold. On the one hand, experienced researchers can experiment with adaptive algorithms that tune parameters of system components according to observed features. And on the other hand, students without a strong background in IR core technology can acquire this knowledge in practical experiments by tuning these parameters manually from the graphical user interface. The core of the Xtrieval framework is organised according to the basic architecture of a modern IR system. Components like text transformation, document scoring, or automatic feedback are based on abstract classes. This makes it possible to quickly configure the framework according to a specific IR problem (see Section 5.4 for a list of examples).

- *Support for Retrieval Evaluation*
  One of the central objectives for the design of the Xtrieval framework is to support experimental IR evaluation in a straightforward way. A number of en-

hancements for Xtrieval that could be beneficial for IR researchers have been identified by conducting empirical experiments. Since precision-recall graphs contain more information about the effectiveness than typical summary measures like AP or MAP, a precision-recall graph visualisation was incorporated into the graphical user interface of Xtrieval. This allows the comparison of different system components and their configurations on individual queries or on a complete test collection. An interface to import relevance assessments from evaluation campaigns in order to realise the visual presentation of system effectiveness was also designed and implemented. In addition to that, the Xtrieval framework can also be used for collecting relevance assessments. Here, the graphical user interface is used to present a defined number of hits for a query, each of which can then be marked according to a given relevance model. The resulting relevance assessments are stored in the widely-used trec_eval format.

- *Reproducible Experiments*
  Over the past two decades, many test collections were created at evaluation campaigns. The test collections and the results were stored in archives. But the system descriptions were published in various forms. This raises the question of how those collected experiments relate to each other, and more importantly, how the systems and their component configurations relate to each other. The Xtrieval framework allows the storage of the complete system configuration covering all components and parameters. The data format that is used for this purpose is XML serialisation. System and experiment configurations are represented as Java classes. Xtrieval de-serialises them to XML retaining all relevant information, in order to be able to revert the process at a later stage. This allows re-running existing experiments, for example in cases where a systematic failure was identified in the course of the evaluation process.

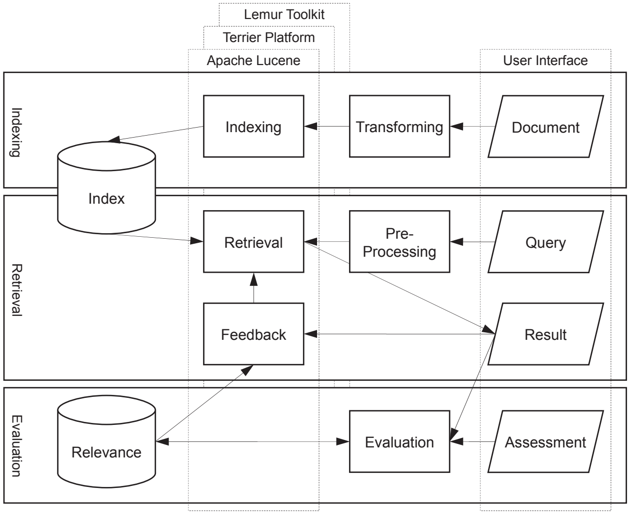


Figure 5.1: General architecture of the Xtrieval framework, redrawn from [108, p. 108].

5.2 Architecture

The Xtrieval framework is implemented in Java and consists of four major components. *Indexing*, *retrieval*, and *evaluation* [108] form the abstract and object-oriented system core of Xtrieval. The graphical user interface is the complementary fourth element. Implementations for the calculation of common effectiveness measures as well as import and export of relevance assessments are ready to be used. The graphical user interface serves as interface for the system and experiment configuration. It is abstracted as application layer and not necessary for experimentation, because the APIs of Xtrieval are typically accessed via programming code. Nevertheless, the graphical user interface can be useful to identify and understand particular problems, for instance topic-specific failures. The basic architecture of Xtrieval is illustrated in Figure 5.1 [192, p. 33], [108, p. 108].

In order to demonstrate how the formulated design goals were realised in Xtrieval, the most important enhancements are discussed in the following subsections. But first the emphasis is put on general modules and enhancements of the first prototype of Xtrieval. An aspect that illustrates the flexibility of Xtrieval, is the ability to process documents, queries, and ranked results, in controllable collections. This allows the implementation of flexible filters, which can be applied both during indexing and retrieval. A prime example is multilingual search where a *TranslationFilter* can be used to translate document or query collections. Xtrieval also supports a straightforward structural filter that allows the combination of multiple document fields into a single field according to a defined structure. The only decision that needs to be made when using such a filter is what field(s) should be combined into which new field(s). It has already been pointed out (see Section 5.1) that Xtrieval provides a substantial collection of stemmers that are based on different approaches than the frequently used Snowball stemming. This collection includes several stemmers for German, English, French, and Russian.

Data fusion algorithms [102, p. 69ff], [192, p. 40] and general purpose similarity functions [102, p. 23ff] represent further extensions of Xtrieval. The merging module (that is used for data fusion) is usually applied to test the result list fusion hypothesis, i.e. it combines search lists either on-the-fly or in a post-retrieval stage. The flexibility in configuring the system and its components in combination with different approaches to fuse the results, opens an entire field of empirical research. Currently, Xtrieval supports the following score-based fusion operators: a) SumScore, b) ProductScore, and c) Z-Score. The similarity module provides basic coefficients to measure the resemblance of vectors. Xtrieval incorporates implementations for the Dice, Cosine, Jaccard, and Ward coefficients. They were used to analyse whether clustering approaches can be used to select more diverse documents for automatic query expansion [104]. That study demonstrated that retrieval effectiveness can be slightly improved for the given test collection when a local document clustering approach is applied.

Another important feature of Xtrieval is its abstract feedback module. Due to the abstract illustration in Figure 5.1, it is not clearly shown how automatic or manual feedback can be activated. The framework only provides an interface for feedback, together with three different kinds of abstract feedback mechanisms. These are: (1) relevance feedback, (2) blind feedback, and (3) manual feedback. The latter is only available from the GUI and is implemented as a combination of relevance assessment and user-specific relevance feedback, i.e. the manual feedback is not interactive. The feedback algorithms are coupled with the corresponding IR cores. Terrier, for example, provides a number of different weighting schemes for pseudo-relevance feedback.

All of the modules presented so far constitute major elements of IR systems. The approach of the creation of a collection of these methods in order to compare them based on scientific principles is an important contribution. In fact it is comparable to the idea for the development of the Terrier platform. The difference between Xtrieval and Terrier is the level of abstraction. While Terrier focuses on the core models of IR, Xtrieval aims to unite all major tools of modern IR systems that are needed to solve any possible search task. For this reason, the integration of two widely-used IR frameworks, namely Lucene and Terrier, is discussed in the following subsections. Although the integration of Lucene was a major contribution from [192], it is included here in order to illustrate how different types of IR libraries are aggregated in Xtrieval.

Besides the Terrier platform and Apache Lucene, the Lemur toolkit has also been integrated in Xtrieval. However, the integration of Lemur had only little value for the formulated goals of Xtrieval due to different problems. Lemur is written in C++, but its major functionality is accessible through a Java API. Consequently, this interface was the basis for the integration to Xtrieval. Due to this Java to C++ wrapper, there is a severe limitation for simultaneous access to the data structures of the Lemur toolkit. The library can only handle a single index and a single search request at a time. Singletons were used to avoid side effects when accessing the library with multi-threaded components of Xtrieval. However, this was not only a problem regarding scalability. It did also limit the flexibility of our framework, which was designed to allow the change of the configuration of components during runtime. Because of this major problem and the fact that the development of the Lemur toolkit was terminated in the end of 2010, the support for Lemur was discontinued in Xtrieval as well.

5.2.1 Lucene Integration

It was pointed out before that one of the most important aims for the development of Xtrieval was to incorporate different IR toolkits and APIs. This was achieved by a redesign of the first prototype of Xtrieval, which was built on top of Lucene. Note that this initial design of the Xtrieval framework is a contribution of the work described in [192]. The following discussion is based on the current version of the integration of Lucene in Xtrieval. It resulted from continuous scientific experimentation with the Xtrieval framework and illustrates the advancement since its initial design and implementation. In order to keep matters clear, the emphasis of this presentation is put on the two major processes of all IR systems: indexing and retrieval.

As a first step concerning the integration of the indexing process of Lucene, the relationship of Figures 5.1 and 5.2 is established. Figure 5.2 provides an overview on the top part of Figure 5.1, where the major elements are denoted with the concepts *Indexing*, *Transforming*, and *Document*. The Xtrieval framework covers these elements with according classes and interfaces. The text transformation process is managed by implementations that access the text transformation architecture of Lucene which was described in Section 4.1.1. Next, the emphasis is put on the abstraction of document collections and the creation of retrieval indices. Figure 5.2 shows the main classes for the integration of these two elements. Each of the rectangles represents a single class. The colours indicate the type of the class: blue boxes are interfaces and green boxes are abstract or actual implementation classes. The names of the classes and the packages are presented in the top of each of the boxes. In the top of Figure 5.2 the Xtrieval interfaces *Indexer* and *DataCollection* are shown. It can also be seen that the central class for the integration of the indexing processes of Lucene is *LuceneIndexer*.

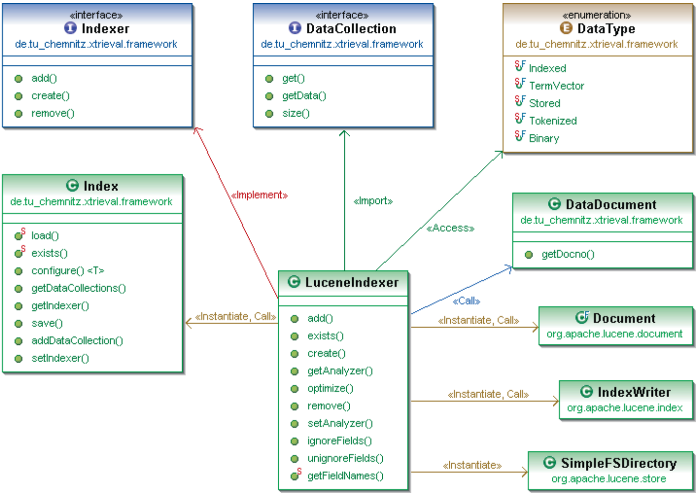


Figure 5.2: Lucene integration in Xtrieval – Indexing process.

The *Index* class manages the access to all underlying index data structures of Lucene or other IR toolkits. Note that the interfaces *Indexer* and *DataCollection* as well as *Index* and *DataDocument* represent the abstract classes for the integration of the indexing procedures. Thus, they will also appear in the corresponding figure in the following Subsection 5.2.2 describing the integration of Terrier. The classes *Document*, *IndexWriter*, and *SimpleFSDirectory* are implementation classes of Lucene which are accessed by methods of the *LuceneIndexer*. They are not illustrated in detail in order to keep the presentation clear.

The next part of this discussion is concerned with incorporating the actual search processes of Lucene into Xtrieval. It elaborates on the details of the middle element of Figure 5.1 that is denoted as the retrieval component of Xtrieval. Figure 5.3 illustrates the central classes that manage the retrieval process in Xtrieval. Again, there are two major types of classes to distinguish: process-oriented classes for search and classes

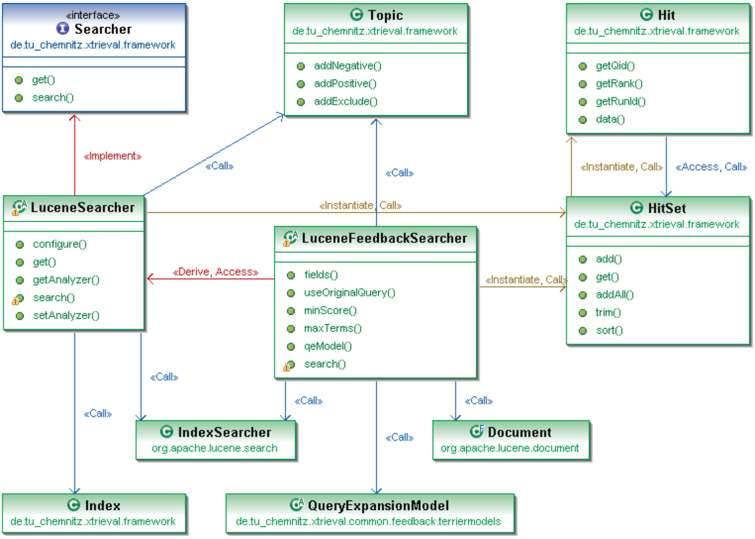


Figure 5.3: Lucene integration in Xtrieval – Retrieval process.

that manage the resulting data. Interface classes (in blue) and abstract and implementation classes (in green) are presented using the same colour-coding like before. The *Searcher* interface class in the top-left of Figure 5.3 inherits the search method to all underlying classes. All general settings for the retrieval process configuration based on Lucene is managed by the *LuceneSearcher* class. The *LuceneFeedbackSearcher* is the most convenient class to access Lucene via Xtrieval, because it also includes a generic feedback mechanism. It was pointed out before, that the actual feedback can be obtained from different sources, like explicit relevance assessments, or pseudo-relevance feedback. The classes *Topic*, *Hit*, and *HitSet* in the top and right part of Figure 5.3 manage the access and creation of the input and output data, i.e. queries and corresponding result lists. The illustration demonstrates that Xtrieval attaches all types feedback to the query representation in the *Topic* class. The classes that connect the retrieval mechanisms of Xtrieval with the actual implementations of Lucene are presented near the bottom of Figure 5.3. These include the *LuceneSearcher* class and the *Document* class. The connection to the classes *Index* illustrates that the configuration of the index is needed in order to start the retrieval process. It can also be seen that the feedback in Xtrieval is handled by the wrapper class *QueryExpansionModel*, which provides access to the feedback algorithms of Terrier. The integration of Terrier is described in more detail in the following subsection.

5.2.2 Terrier Integration

This section describes how the Terrier retrieval platform was integrated into Xtrieval. Terrier can be configured using key-value pairs. For that reason it was necessary to implement wrapper classes for the configurable components of Terrier in order to retain the possibility of flexible configuration during runtime. Figure 5.4 provides an overview of the main classes that realise this integration. Elements in the top-left of the illustration represent general Xtrieval interfaces or abstract classes for indexing. The elements in the bottom-right of Figure 5.4 are interfaces corresponding to Terrier. All remaining components, between those that are labelled with Xtrieval in the top and Terrier in the bottom, are needed to connect indexing interfaces of Terrier and Xtrieval. Following the layout of our abstract illustration in Figure 5.1, elements in the left generate and control index data structures, and elements in the right represent document-level data structures.

The *TerrierDocumentAdapter* was implemented to transform a *TerrierDocument* into an Xtrieval *DataDocument* and to adjust all document properties in the configuration of Terrier. A class named *TerrierCollection* allows us to gain control over the document processing during indexing. The actual processing of *TerrierDocuments* is handled by a wrapper class called *TerrierIndexer*. It encapsulates various indexing methods implemented in Terrier and controls the text transformation procedures. The resulting index data structure is an Xtrieval *Index*. The *Index* object represents the abstraction to access index data structures of different retrieval engines.

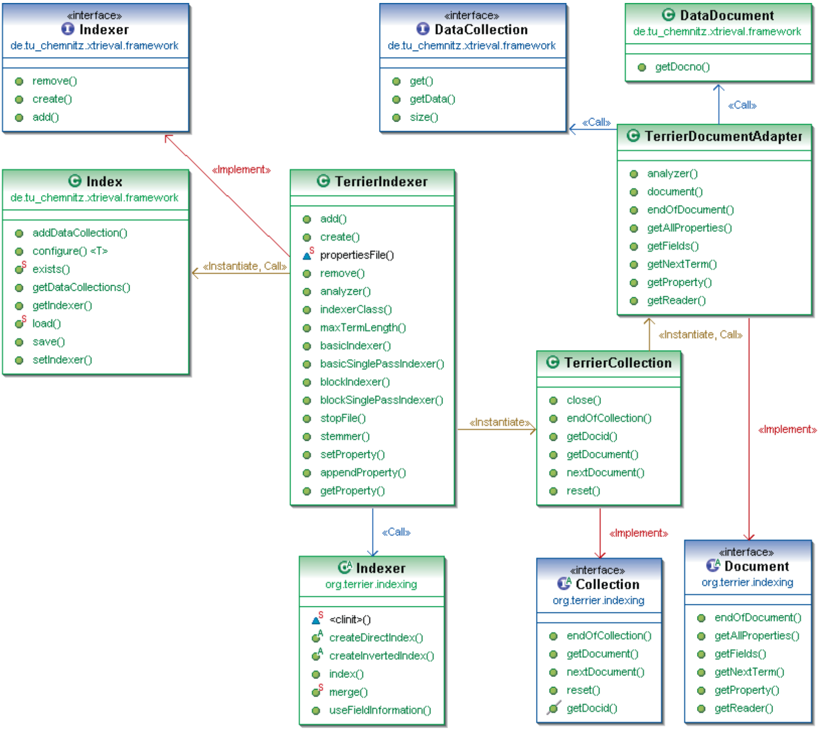


Figure 5.4: Terrier integration in Xtrieval – Indexing process.

In order to ensure easy integration of various IR toolkits or libraries, the *Indexer* interface was designed to be rather minimalistic. It covers only essential methods that are needed to create an index from a collection of documents. A *TerrierIndexer* can be configured to use either Lucene's text transformation framework by passing a Lucene *Analyzer* to the class constructor, or to rely on one of the *TermPipelines* supplied with Terrier.

In practice, the latter are rarely used in experimental evaluations with Xtrieval because Lucene's framework is more flexible (see Section 4.1.1). The implementation

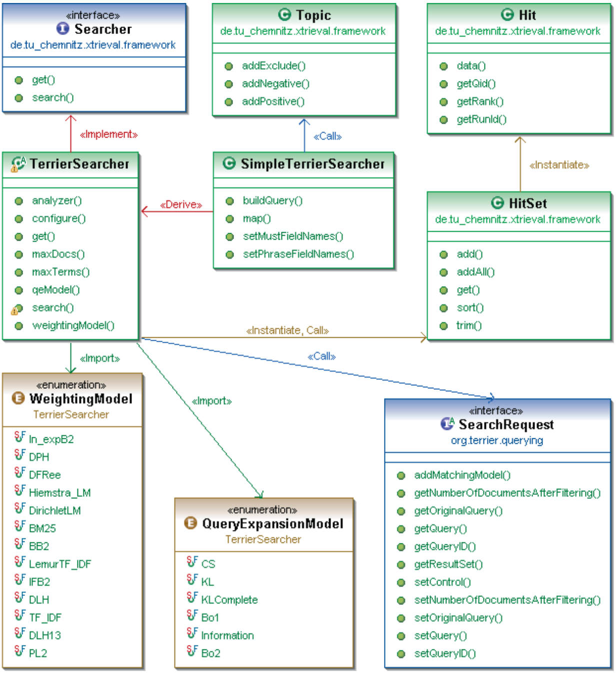


Figure 5.5: Lucene integration in Xtrieval – Retrieval process.

of the *TerrierIndexer* class is relatively complex because of all the different ways of indexing that are supported in Terrier. A limitation of this low-level integration lies in its maintenance. Any new release of Terrier that contains changes in the indexing procedure will not be supported unless the presented wrapper classes are adapted accordingly. Section 4.1.2 showed that Terrier's strength lies in its flexible weighting and ranking model. Considering the motivation for integrating Terrier, all of the IR models implemented in Terrier, should be available in Xtrieval.

Figure 5.5 illustrates classes and interfaces that were designed and implemented to integrate Terrier. Again, all the elements in the top of the illustration represent Xtrieval interfaces and classes. Terrier's flexible weighting and ranking algorithms can be accessed through a single interface named *SearchRequest*. A particular model that was chosen for retrieval can be passed as a parameter. For that reason, created two enumerations containing all weighting models for retrieval and feedback were created. They are part of the essential *TerrierSearcher* class, which is a wrapper for Terrier's *SearchRequests* and serves as the connector to the retrieval architecture of Xtrieval. In addition to the choice of the weighting model, both the selection and configuration of automatic feedback models are also controlled via *TerrierSearcher*. Xtrieval contains a further class named *SimpleTerrierSearcher*, which processes the queries and transforms them into the corresponding query object in Terrier. *SimpleTerrierSearcher* also controls the handling of document and query structures and it ensures that Xtrieval's structured query objects (Topics) are processed correctly. Retrieved documents are handled and represented as *Hits* or *HitSets* in Xtrieval. *HitSets* serve as temporary storage for returned document identifiers of search experiments. Since retrieved *HitSets* are an essential element of any evaluation experiment, they are part of the information that is exported and imported for experiment re-use.

5.2.3 Combining Lucene and Terrier in Xtrieval

This section illustrates how Lucene and Terrier are combined in Xtrieval in order to conduct the large-scale empirical analysis in Chapter 7 for the standard ad-hoc retrieval evaluation scenario. It has been pointed out in Section 4.1.1 that the strength of Lucene lies in its flexible and feature-rich architecture for the transformation of text into tokens. A flexible ranking and term weighting implementation is an outstanding feature of Terrier (see Section 4.1.2 for details).

Figure 5.6 demonstrates that these two advantages are linked together using Xtrieval. The illustration also shows the major components and corresponding instances that will be analysed later. The generic process of an IR process is shown in the top of Figure 5.6. It shows documents, queries, and results, which represent the input and output to this generic process. In this presentation the IR process is composed of three major components: text transformation, matching and ranking, and feedback. Each of these components is illustrated in more detail with the additional boxes in the bottom of Figure 5.6.

Text transformation is the first main component of the illustrated IR process. It can be observed that it consists of several filters that implement default operations to transform text into tokens. In this dissertation the main focus of text transformation is put on different stemming algorithms (see the corresponding box in the bottom-left of Figure 5.6). A detailed analysis of the integrated algorithms and their implementations was provided in Section 3.1. Matching and ranking represents the next key component with various algorithms that can be accessed via Xtrieval. Currently, Lucene provides only a single ranking model and Terrier covers the most widely-used ranking models (see Section 3.2 for the foundations of the different models). The optional feedback component consists of two thresholds that are controlled within Xtrieval and the actual term weighting algorithms are provided by the Terrier framework.

This experimental set-up covers the major components that are necessary for ad-hoc search tasks. It constitutes the general architecture for the large-scale empirical evaluation of state-of-the-art IR system component presented in Chapter 7. In contrast to that, current search applications used in practice are more complex. This complexity and the amount of additional components in these applications depends on the nature of the documents and the desired type of service that the operator of the corresponding application provides to the user. The abstract and flexible design of Xtrieval allows the insertion of further components or the creation of completely new component-based IR system architectures. As a result, it is possible to investigate more complex search tasks, given that the required components are integrated accordingly.

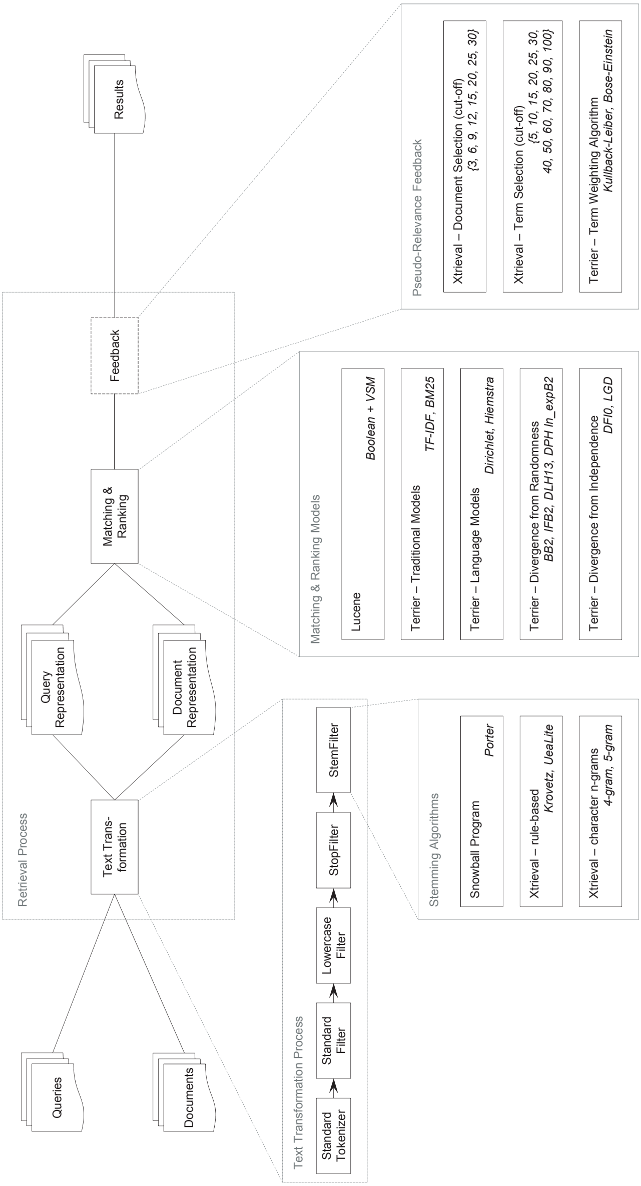

Queries
Documents
Retrieval Process
Text Trans-formation
Query Representation
Document Representation
Matching & Ranking
Feedback
Results
Text Transformation Process
Standard Tokenizer
Standard Filter
Lowercase Filter
StopFilter
StemFilter
Stemming Algorithms
Snowball Program Porter
Xtrieval – rule-based Krovetz, UeaLite
Xtrieval – character n-grams 4-gram, 5-gram
Matching & Ranking Models
Lucene Boolean + VSM
Terrier – Traditional Models TF-IDF, BM25
Terrier – Language Models Dirichlet, Hiemstra
Terrier – Divergence from Randomness BB2, IFB2, DLH13, DPH In_expB2
Terrier – Divergence from Independence DFI0, LGD
Pseudo-Relevance Feedback
Xtrieval – Document Selection (cut-off) {3, 6, 9, 12, 15, 20, 25, 30}
Xtrieval – Term Selection (cut-off) {5, 10, 15, 20, 25, 30, 40, 50, 60, 70, 80, 90, 100}
Terrier – Term Weighting Algorithm Kullback-Leiber, Bose-Einstein

Figure 5.6: Schematic overview of the IR system components that are covered by the combination of Lucene and Terrier in Xtrieval.

5.3 Future Directions

With the integration of widely-used IR core libraries and other extensions like various text transformation approaches, Xtrieval supports a considerable number of evaluation scenarios and application use cases. However, there are a few topics for research in IR that remain open for future work. Given the ever-growing amounts of digital information, scalability became an important issue in IR systems. Scalability in IR can be divided into different requirements regarding efficiency, which depend on the actual application. Most importantly, a modern IR system should be able to process large amounts of new (or updated) documents efficiently and be able to provide the answers to queries instantly. If a search application has to serve many users simultaneously, parallelisation becomes a third requirement. Since Xtrieval is designed for research, the first two requirements are a primary concern regarding scalability. Section 4.1 demonstrated that Lucene and Terrier are designed to be used in highly-scalable applications. However, there are a few constraints when the IR engines are used in Xtrieval. In general, scalability is an efficiency problem that is closely coupled with the design and the architecture of software. In the present case, it means that the IR core engines, which are accessed through the meta framework Xtrieval, apply strategies for scalability that are adapted to the underlying approaches and algorithms. As a consequence, no common approach that accounts for efficient indexing and retrieval exists in Xtrieval. Thus, if scalability does matter for a search application, one should revert to the optimised version of one of the underlying toolkits. In order to support highly-scalable evaluation scenarios and use cases, a concept for parallelisation at the meta-level of Xtrieval is needed.

Another option to extend Xtrieval is to incorporate NLP tasks as analysers in the text transformation component. A first step could be the integration of low-level NLP tasks like POS tagging or named entity recognition. A primary concern should be the storing of the information in a convenient format in the index data structures to be able to exploit such payload information for the retrieval process. According to all other components of Xtrieval, the design of NLP analysers should be flexible, i.e. the interface should allow the exchanging of the algorithm or toolkit that performs the actual task. Based on the brief analysis of state-of-the-art NLP toolkits provided in Section 4.2, the integration of OpenNLP and Stanford's CoreNLP library appears to be a promising step. Then, it would be possible to develop event-based IR models, which use POS or entity information on terms either to optimise retrieval effectiveness, or to improve result presentation.

More and more test collections are available for IR system evaluation, but there is no authority that organises all experimental results. Although the IR community is aware of the problem, only a few solutions have been proposed so far [8, 33]. Thus, a final aspect that should be considered as a valuable enhancement not only to Xtrieval, but to the entire IR evaluation community, is a persistent storage for empirical IR experiments. However, the problem is complex for several reasons. First of all, in order to re-produce the outcomes of experiments, a detailed and formalised description is needed. An obvious solution could be a structured XML format. The elements of such a standard experiment description should be discussed in the IR community to achieve a high level of acceptance and use. It might be beneficial to implement the standard at popular evaluation exercises like TREC in order to ensure its application. It is desirable that such a standard allows the description of information in existing experiment archives, like those from TREC or CLEF. This could be achieved by defining different levels of the standard, similar to the popular Dublin Core standard for libraries. The IR evaluation experiment description standard should include references to the system components and their configuration, either as abstract short description, or ideally with references to the actual algorithm, or its implementation. The amount of possible experiment configurations in Xtrieval would allow the study of the most important components of modern IR systems. Such an analysis could be used to initiate the creation of a general standard to archive IR evaluation experiments.

Other potential applications of Xtrieval include the development of further system components for specific search tasks, like Question Answering, or Opinion Mining.

Another intended future use case of Xtrieval is teaching. It could be used to explain the effects of different core retrieval models or other components. Here, the abstract design of the framework is an outstanding feature that allows students without background in IR theory to get to know the most important components of IR systems.

5.4 IR Evaluation with Xtrieval

So far the motivation, architecture, and potential enhancements of Xtrieval have been discussed. This section sheds light on the evaluation experiments that were completed with Xtrieval over the past years. In the subsequent sections, brief descriptions of the conducted experiments are given for the different retrieval tasks. Note that those experiments have been previously published along with their system descriptions and the empirical results. The experiments are replicated here in order to emphasise the key aspects of the hypotheses in this dissertation. The presentation of previous experiments is organised according to the type of the data collections that were used for the experiments.

In most cases these evaluation task have a close relation to a real-world search scenario. The following selection of empirical IR evaluation experiments demonstrates the flexibility of the Xtrieval framework. Since the main focus of this dissertation is on component-level comparison of IR system configuration on ad-hoc search tasks using English document collections, the experiments listed hereafter have been restricted to English document collections and tasks that are similar to ad-hoc search scenarios.

5.4.1 Domain-Specific Document Collections

From 2006 to 2008, the Chemnitz retrieval group participated in the Domain-Specific track at CLEF. Over time, the organisers assembled several document collections from the social science domain. The German Indexing and Retrieval Test (GIRT) database has been used in different versions since the start of the track in CLEF 2001, and the final evaluation in 2008. The GIRT corpus is a pseudo-parallel document collection in English and German. It contains extracts from the SOLIS (Social Science Literature) and SOFIS (Social Science Research Projects) databases from 1990 to 2000. German is the original language of the documents, and the most important fields were manually translated into English to create the corresponding pseudo-parallel corpus.

In addition to the GIRT collection, two Russian document collections were assembled for the Domain-Specific track. The Russian Social Science Corpus (RSSC) was used in 2005 and the INION/ISISS collection, covering Russian social science and economics publications, served as data collection from 2006 to 2008. The English part of the GIRT-4 collection was used for all experiments that are presented in this subsection. It contains about 150,000 document records with bibliographic information like title, author, publication year, as well as several manually assigned controlled vocabulary terms and last but not least about 17% of the documents contain abstracts in natural language. In 2007, 20,000 documents that were extracted from the Sociological Abstracts database from Cambridge Scientific Abstracts (CSA), were added to the English document collection. These additional documents cover publications from the years 1994-1996 and contain the document title, abstract, and subject keywords from a CSA thesaurus. [139]

5.4.1.1 Experiments

For the first participation in CLEF 2006, only monolingual experiments were submitted for evaluation. The main focus of the experimental set-up was put on diverse query expansion (QE) using clustering techniques [102]. It has been pointed out that the focus of this work is on monolingual English ad-hoc retrieval, which is the reason for reporting only those experiments here. Table 5.1 illustrates the results of the exper-

Corpus	Run ID[1]	Lang	Stemmer	QE	Fusion	MAP[2]	Rank[2]
GIRT-4	unineen2	EN	Savoy	PRF	yes	0.4303	1/8
GIRT-4	tucmigirten2	EN	Porter	PRF2 + LDC	no	0.3553	5/8
GIRT-4	tucmigirten4	EN	Porter	PRF1	no	0.3538	6/8
GIRT-4	tucmigirten1	EN	Porter	PRF2	no	0.3510	7/8
GIRT-4	tucmigirten3	EN	Porter	PRF2 + LTC	no	0.3450	8/8

Table 5.1: CLEF 2006 Domain-Specific track – Results for the English subtask.

iments submitted for official evaluation. A detailed description of the system set-up and the design of the experiment is provided in [104].

Two different QE approaches named PRF1 and PRF2 were implemented, where PRF2 was a two-step iterative approach with a blind feedback mechanism that was applied twice to reformulate the original query. LDC and LTC describe the clustering approach that was used for the second step, where the clustering to select terms for the reformulation was either done by clustering terms (LTC) or documents (LDC). The resulting MAP measure covers 25 topics, which is the standard size for the Domain-Specific track at CLEF. The results are compared to the best performing experiment, which is described in more detail in [163]. A relative difference of about 17.5% between the best configuration from Chemnitz *tucmigirten2* and the top-performing experiment *unineen2* can be observed in Table 5.1. Although no statistical significance test was performed, this is a considerable difference. Nevertheless, no systematic failure in the experiments was identified, which might have explained the large gap. The absolute effectiveness values for the experiments from the Chemnitz retrieval group show only little variance. An explanation for this result might be the fact that only the procedure for selecting terms for PRF was adapted. In fact, if a topic has many relevant documents in the collection, the selected approach is very likely to select the same terms that a standard PRF approach would return.

---

[1] From now on all experiment identifiers starting with *tuc* or *cut* refer to experiments that have been submitted for evaluation by the Chemnitz retrieval group, i.e. they represent either sole or collaborative work of the author of this dissertation.

[2] The MAP values and the experiment ranking were extracted from [51, p. 14].

Corpus	Run ID	Lang	Stemmer	Fusion	MAP[3]	Rank[3]
GIRT-4, CSA	unineen4	EN	Savoy	yes	0.3534	1/15
GIRT-4, CSA	cut_ds_en_merged	EN	Porter/Krovetz	yes	0.2985	7/15
GIRT-4, CSA	cut_ds_en_unstruct	EN	Porter	no	0.2952	8/15
GIRT-4, CSA	cut_ds_en_struct	EN	Porter	no	0.1850	15/15

Table 5.2: CLEF 2007 Domain-Specific track – Results for the English subtask.

In 2007, three experiments were submitted to the English subtask of the Domain-Specific track. The central lesson learned from the experiments in the previous year was that an IR system is a complex combination of configurable components. Thus, it was decided to investigate indexing as a central component of the complete retrieval process. A question of particular interest was how the structure of the document abstracts affects retrieval effectiveness. In order to investigate this problem, a straightforward approach based on two different types of indexes was implemented. The first index mapped every field in the document structure to a corresponding field in the index, i.e. the document structure was completely preserved. In contrast to that, the document structure was omitted in the second index, i.e. the contents of all fields were mapped to a single field in the index. A more detailed description of the experimental set-up is given in [109].

Table 5.2 summarises the outcome of these experiments and reports the best performing experiment for reference. The presented MAP values were accumulated using a new set of 25 topics. In contrast to the strategy of comparing and combining different index structures, the top-performing experiment combined results from different ranking algorithms and relevance feedback configurations [60]. Similar to the results in 2006, there is a considerable gap of about 15% between the best experiment from the Chemnitz retrieval group and the top-performing run. The hypotheses for the experiments were two-fold. First, it was expected that a structured index would result in stronger performance than a unstructured index. This assumption was based on the loss of information that occurs when the document structure is omitted during indexing. The experimental results on the given test collection indicate that this hypothe-

---

[3] The MAP values and the experiment ranking were extracted from [52, p. 47].

Corpus	Run ID	Lang	Stemmer	Fusion	Thes.	MAP[4]	Rank[4]
GIRT-4, CSA	cut_merged	EN	Porter/Krov.	yes	no	0.3891	1/12
GIRT-4, CSA	cut_merged_thes	EN	Porter/Krov.	yes	yes	0.3869	2/12
GIRT-4, CSA	uninedsen1	EN	Porter	yes	no	0.3770	3/12

Table 5.3: CLEF 2008 Domain-Specific track – Results for the English subtask.

sis is invalid, because of the low MAP values for experiment *cut_ds_mono_en_struct*. However, a complex optimisation problem exists for that particular experiment. Due to the number of different document fields and the tripartite topic structure, the selection and weighting strategy, that was needed to map the topic elements to document fields, may have markedly affected the retrieval performance. A third experiment was submitted in order to evaluate the second hypothesis that combining the results of both indexes would result in better effectiveness. Although, there is a significant difference between the basic experiments, a slight improvement for the combined experiment *cut_ds_mono_en_merged* was found. This result was also consistent across collections and languages.

In the following year, the Domain-Specific track was offered for the last time. The data collections remained unchanged and thesaurus mappings for GIRT-4 and CSA were provided in order to overcome differences in the technical language of different controlled vocabularies [139]. The experimental set-up was based on the fusion approach of the previous year. A detailed description of these experiments is given in [111] and a brief overview follows hereafter. Two indexes using the Porter and the Krovetz stemmers were created. Due to the bad results of the previous experiments, the structure of the documents was omitted in the index. The monolingual thesaurus mappings between the two English data collections GIRT-4 and CSA were used to extract additional terms from the topic. These terms were used to create the queries that were submitted to our indexes. The last step of the retrieval process consisted of a blind feedback approach using the top-k documents to extract frequent terms. In the final step of the applied experimental set-up, the results obtained from the indexes

---

[4] The MAP values and the experiment ranking were extracted from [53, p. 14].

were combined into a single result list. Only two experiments have been submitted to the monolingual English subtask of the track. Table 5.3 illustrates the outcome of these experiments.

In contrast to the previous years, the experiments submitted by the Chemnitz retrieval group achieved the strongest effectiveness in terms of MAP. These experiments resulted in similar retrieval effectiveness. In contrast to the expectation that using the monolingual thesaurus mapping for query formulation would improve effectiveness, a small decrease in MAP could be observed for the thesaurus-based experiment. A similar effect was found for the experiments with the other data collections of the Domain-Specific track [111].

5.4.1.2 Lessons Learned

In the course of the participation in the Domain-Specific track, the first prototype of the Xtrieval framework was designed and implemented. All experiments were based on Lucene and the effects of different indexing schemes and query (re-)formulation procedures on retrieval effectiveness were studied. The resulting MAP of the submitted runs varied between about 0.30 and 0.40. Although comparison across test collections might be untidy, the quality of the system has improved over time. An indication for this finding is the relative performance of the experiments in comparison to other participating groups. Although this conclusion is not robust, a few veteran groups, like the University of Neuchâtel and the University of Berkeley, can serve as reference baseline over the years. If the differences to the IR group from Neuchâtel, which used similar fusion approaches over the years, are considered as a good reference the Xtrieval framework was considerably improved.

Next, the approaches and their impact on retrieval effectiveness measured in MAP are recapitulated. Two approaches to reformulate original queries were studied: a cluster-based technique in 2006 and a thesaurus-based method in 2008. Both approaches had only little impact on search quality. While the local document clustering (LDC) method slightly increased mean average precision, the thesaurus-based expansion slightly decreased the retrieval performance. The experiments based on structured and unstructured indexing showed that omitting document structure does not decrease retrieval effectiveness in domain-specific retrieval, it even outperformed several experiments that made use of document structure. This effect is attributed to the complex problem of determining optimal weights for the different elements of the document structure during the query formulation process.

5.4.2 Open-Domain Bibliographic Collections

In 2008 and 2009, the Chemnitz Retrieval group participated in the Ad-Hoc track of the CLEF initiative. The document collection consisted of approximately one million library catalogue records from *The European Library* in English, French, and German language. This document collection marked a change from using standard news collections that contain large and well structured documents, to a sample from a real document corpus of a library. Instead of actual text documents, this corpus consisted of structured document descriptions. In addition to that, the data is very sparse, i.e. most of the records contain only a few important fields like a title or an abstract. The fact that the field contents of each document could appear to be in different languages made matters even worse. Actually, only about half of the documents were in the language that the data collection was labelled with, i.e. the English corpus that is referred to hereafter contains only about 570,000 documents in English. Those typical artefacts of a real-world multilingual document collection qualified the track as going beyond standard retrieval evaluation. Another significant difference to the Domain-Specific track is the larger size of the topic set, which consists of 50 queries here. TEL-BL is used to refer to the English collection from the British Library. An in-depth description of the track and the test collections is provided in [1].

5.4.2.1 Experiments

In 2008, the Chemnitz retrieval group participated in the CLEF Ad-Hoc track for the first time. The purpose was to investigate the track organisers' supposition that the knowledge about the multi-lingual nature of the documents could be exploited to increase retrieval effectiveness. This required a specific experimental set-up. In order to find documents that contain query terms in a language other than the language of the topic, the original query was translated to the ten most common languages (in terms of frequency of documents in the collection). As a consequence, a multilingual query was obtained. Due to huge differences in the number of documents for each language, a linear weighting scheme was implemented to compensate for this effect. Four experiments were submitted in total to the monolingual subtask on the English collection, where one served as the baseline reference and did not account for the multilingual documents, i.e. only the original query was used for retrieval. For the other experiments, different strategies were applied to account for the frequency of each language in the collection. The baseline experiment weighted (LW) all terms of the query equally, regardless of the language. In one of the remaining configurations, the frequency x of the documents in a each language was used to boost documents written in the most frequent language of the collection. Whereas $1 - x$ was used for our last experiment. This approach was based on the hypothesis that relevant documents might appear in any language of the collection and query terms in a rare language should be boosted in order to prefer those documents in the ranking. The general system configuration was based on the findings from preceding participations in the Domain-Specific track. The core retrieval system was Lucene. A language-specific stop-word list was applied and two indexes were created, one using the Porter stemmer and another one using the Krovetz stemmer. Again, the results were combined from both indexes and a blind feedback approach was applied, which reformulated the original query using the most frequent terms from the top-10 documents of an initial ranking. Further details on the system configuration can be found in [107].

Corpus	Run ID	Lang	Stemmer	Fusion	LW	MAP[5]	Rank[5]
TEL-BL	unineen3	EN	Savoy	yes	-	0.3753	1/37
TEL-BL	cut_merged_simple	EN	Porter/Krovetz	yes	-	0.3561	4/37
TEL-BL	cut_multi10_wx++	EN	Porter/Krovetz	yes	x	0.2484	30/37
TEL-BL	cut_multi10_w1-x++	EN	Porter/Krovetz	yes	1-x	0.1620	34/37
TEL-BL	cut_multi10_w1++	EN	Porter/Krovetz	yes	1	0.1453	35/37

Table 5.4: CLEF 2008 Ad-Hoc (TEL) track – Results for the English subtask.

Table 5.4 compares the best experiment to the results for the runs that were submitted by the Chemnitz retrieval group. The empirical evaluation reveals that the strategy to formulate multi-lingual queries was significantly outperformed by the baseline reference experiment *cut_merged_simple*. Nonetheless different weighting schemes for the multilingual queries terms did affect retrieval effectiveness considerably. Using equal weights for the multilingual query resulted in the worst MAP value. The experiment based on frequency-inverse weights for the language of the query terms (namely, *cut_multi10_w1-x*++) produced slightly better results, but it did not achieve even half of the effectiveness of the baseline experiment in terms of MAP. From the language weight-based query formulation experiments, the weights based on the document frequency of the language performed best. Nevertheless, there was still a significant difference to the reference experiment and the best experiment across all groups.

However, the consistent improvement of the Xtrieval framework, which can be attributed to the combination of different indexing approaches, stimulated further experiments on this particular test collection. In late 2008, the Xtrieval framework allowed the access to different IR toolkits for the first time. As a consequence, different ranking algorithms were compared by making use of the Lemur toolkit integration. The system configuration and the empirical results were published in [110]. These results are presented here, because the extension to exchangeable ranking models is an important foundation for this dissertation. The goal of this post-workshop evaluation was two-fold. First, an empirical verification of the technical implementation was needed. And second, there was an interest in studying the effect that different retrieval algorithms might have on retrieval performance. Again, two indexes were created us-

---

[5] The MAP values and the experiment ranking were extracted from [53, p. 16].

Corpus	Run ID	Lang	Stemmer	Ranking	MAP
TEL-BL	unineen3	EN	Savoy	-	0.3753
TEL-BL	cut_pws_1	EN	Porter	Okapi	0.3452
TEL-BL	cut_pws_2	EN	Krovetz	InQuery	0.3334
TEL-BL	cut_pws_m_e1e2	EN	Porter/Krovetz	Okapi/InQuery	0.3135
TEL-BL	cut_pws_3	EN	Porter	Lucene	0.3758
TEL-BL	cut_pws_4	EN	Krovetz	Lucene	0.3731
TEL-BL	cut_pws_m_e3e4	EN	Porter/Krovetz	Lucene	0.3880
TEL-BL	cut_pws_m_e1e3	EN	Porter	Okapi/Lucene	0.3800
TEL-BL	cut_pws_m_e2e4	EN	Krovetz	InQuery/Lucene	0.3837
TEL-BL	cut_pws_m_e1e4	EN	Porter/Krovetz	Okapi/Lucene	0.3908

Table 5.5: CLEF 2008 Ad-Hoc (TEL) track – Additional results for the English subtask.

ing the Porter and Krovetz stemmers. Four ranking algorithms, provided by Lemur, and the TF-IDF variant, implemented in Lucene, were used for retrieval. Due to the experimental state of the Xtrieval framework at that time, it was not possible to apply blind relevance feedback. Since the empirical results were generated with different retrieval cores, the results are reported in separate sections in Table 5.5. Again, the best experiment from the official evaluation is presented in the first row. A list of three experiments that were based on the Lemur toolkit follows. The experiments configurations *cut_pws_en1* and *cut_pws_en2* achieved the best retrieval effectiveness from the eight runs in this small configuration grid (consisting of two stemming and four ranking algorithms). A further experiment that combined different ranking and stemming algorithms is also presented. This run did not outperform the other experiments in terms of retrieval effectiveness.

The second group of experiments in Table 5.5 is solely based on Lucene as the retrieval core. Since the ranking model in Lucene is fixed, only the impact of two different stemming methods are compared here. Runs *cut_pws_3* and *cut_pws_4* show that both approaches result in similar retrieval effectiveness. The last experiment in that group combines results from both indexes. Here, a small increase in retrieval performance can be observed. Note that experiments *cut_pws_m_e3e4* from Table 5.5 and *cut_merged_simple* from Table 5.4 used identical indexes and the same result combination approach. The only the difference is that run *cut_merged_simple* was configured with blind relevance feedback and run *cut_pws_m_e3e4* without. The results indicate that omitting the blind relevance feedback slightly improved retrieval effectiveness in terms of MAP (0.3561 vs. 0.3880) on this particular test collection. The last section of Table 5.5 lists experiments that combine ranking models from the Lucene and Lemur retrieval toolkits. The results of the top-performing experiments from each retrieval engine were combined in a data fusion algorithm. All experiment identifiers of merged runs contain a reference to the experiments which they are based on. The results demonstrate that the data fusion approach improves performance in each of the test cases. Although there were only slight differences between the MAP values of merged experiments, a consistent improvement over the corresponding baseline runs was detected.

Bearing in mind the aim of the integration of different state-of-the-art retrieval models, Terrier was also incorporated into Xtrieval for the participation in the Ad-Hoc track of CLEF 2009, see Section 5.2.2. Except for a fresh set of topics and relevance assessments, the test collection remained unchanged for this sequel of the track. Based on further investigations of the previous strategy to exploit the multilingual content of the document collection, the weighting scheme was adapted to account only for the most common languages.

According to our previous observations (refer to Table 5.4), a weighting scheme based on the frequency of the language in the collection was applied. It was assumed that this particular experiment configuration would be superior to a standard monolingual retrieval approach. In addition to that, the effect of the *BB2 ranking model* from the divergence of randomness framework available in Terrier was studied. This experiment was also compared to a parallel system configuration that used Lucene for ranking. These two experiments were combined using our implementation of the Z-Score data fusion. The details of the system configuration and all experiments are discussed in [103]. In accordance with the previous results, Table 5.6 compares the outcome of our experiments to the top-performing run.

Corpus	Run ID	Lang	Stemmer	Ranking	MAP[6]	Rank[6]
TEL-BL	inesc.run11	EN	Porter/N-Gram	LM/Lucene	0.4084	1/46
TEL-BL	cut11	EN	Porter	BB2/Lucene	0.4071	2/46
TEL-BL	cut12++	EN	Porter	BB2	0.3914	7/46
TEL-BL	cut9	EN	Porter	BB2	0.3864	9/46
TEL-BL	cut10	EN	Porter	Lucene	0.3672	15/46

Table 5.6: CLEF 2009 Ad-Hoc (TEL) track – Results for the English subtask.

These results are analysed with respect to two hypotheses. First, the effect of the revised multi-lingual query formulation and weighting approach was examined. Experiments *cut9* and *cut12++* deal with this matter. Both experiments used identical system configurations, but differed in the query construction procedure. The multilingual query formulation method (cut12++) marginally improved retrieval effectiveness on this test collection. However, it might be argued that this small gain is not statistically significant. The second goal was to compare the ranking methods Lucene and BB2. The results for the corresponding experiments (cut10 and cut11) indicate that the BB2 method from the divergence of randomness framework is superior to Lucene. Experiment *cut11* verified our previous finding: that combining results from different ranking or stemming methods improves retrieval effectiveness. However, these merging strategies require more processing time in order to apply different ranking models. In addition, fusing different stemming methods obviously requires more storage space for indexing. A final comment needs to made on the relation to the best experiment *inesc.run11*. It can be seen that both the runs under examination and the best experiment merged different ranking models, but the resulting retrieval performance appears to be similar. This could be considered as an evidence that combining different retrieval strategies has a positive impact on retrieval effectiveness. Whether this advantage outweighs the additional requirements in terms of space and computation time or not, remains an open question.

---

[6] The MAP values and the experiment ranking were extracted from [54, p. 18].

5.4.2.2 Lessons Learned

Throughout the two years of experimental evaluation with the Ad-Hoc TEL collection, Xtrieval has been extended to support access to Lucene, Lemur, and Terrier. This allowed the comparison and combination of different retrieval models in actual evaluation settings. The obtained results indicate that different IR system configurations vary in retrieval effectiveness. Consistent improvements in retrieval effectiveness for combined ranking algorithms or indexes is another observation that can be made when analysing the empirical results. An interesting note on the fusion approaches is that an upper limit seems to exist regarding retrieval effectiveness measured in MAP. For both the presented test collections the “winning” strategy was to use a result list fusion process at the index and/or ranking levels. The experiments presented in this section support this claim since the merged experiments achieved similar performance values. Given the experiments on that particular test collection, a few questions remain unanswered. For instance, it is unclear whether the combination of different stemming techniques affects retrieval performance more than merging different ranking algorithms. In a realistic scenario, combining different ranking models might be infeasible due to computational restrictions. Then, it is important to identify the optimal system configuration that uses only a single index and ranking approach. Regarding the multi-lingual aspect of the present document collection, it was shown that a weighted query term translation approach can slightly improve retrieval effectiveness. The success of such a strategy does, however, depend on an appropriate weighting scheme that reflects the frequency distribution of languages in the document collection.

5.4.3 Searching Multimedia Descriptions

Rapid development of cheap still and moving picture capturing devices since the late 1990’s has led to huge amounts of available digital information. However, managing such collections of multimedia objects in order to retrieve desired pieces of information or an object itself, is more challenging than handling pure textual information.

The reasons are the challenges of comparing the content of multimedia assets, be it an image, an audio file or a video. Efficient information retrieval is based on the paradigm of comparing condensed document surrogates. But obtaining these surrogates is difficult even from an abstract perspective. A textual document like a newspaper article about a particular building in your home town will most likely contain its name and other contextual information like its location. A photograph of that building will not reveal this information, unless it was annotated by somebody in order to provide this context. Generating such contextual annotations for media objects is a task that can be accomplished in different ways. Textual annotations are, however, the basis of common multimedia retrieval approaches. The most straightforward way is manual description by a human. Of course, it is also the most expensive strategy. In contrast to that, automatic analysis of multimedia data is less expensive in terms of costs, but usually it is also limited in quality and accuracy. Switching from one form of media to another one (i.e. text) is a central challenge in multimedia retrieval, which is often referred to as *semantic gap*. In the following subsections three kinds of multimedia retrieval tasks are discussed. All of them are based on textual representations, but each of them investigates different challenges according to the type of the underlying media objects.

5.4.3.1 Image Retrieval

From 2006 to 2009 the Chemnitz retrieval group submitted contributions to several evaluation tasks that dealt with retrieving images. These are briefly discussed in the following. In 2006 and 2007 the ImageCLEFphoto track used a new collection named IAPR-TC12 Benchmark. This collection contained 20,000 coloured photographs of natural scenes provided by a travel company that organised trips to South America. The images were supplied with semi-structured annotations in English and German. These short image captions were organised in seven fields: (1) an image title, (2) a textual description of its contents, (3) the time and (4) the place the photograph was taken, as well as (5) notes for additional information, (6) the name of the photographer, and (7) an unique identifier. Since the collection was assembled to be realistic, only about 70 percent of the annotations contain the first five fields. The remaining captions were mostly supplied with location and date (20 percent), 10 percent also had a title and a further 10 percent did not have any caption. The topics for the ImageCLEFphoto tasks in 2006 and 2007 contained 60 queries that were carefully selected using a custom creation and administration system used at CLEF. A detailed description of the IAPR-TC12 collection and the topic creation process is given in [69].

The purpose of the participation in the ImageCLEFphoto task was to develop and test the predecessor of the Xtrieval framework. A specific requirement for image retrieval was to incorporate textual and content-based retrieval approaches in order to compare and combine them. Since 2006 marked the first year of participation in retrieval evaluation campaigns, the focus was to implement and test the prototype implementation that was based on Apache Lucene (version 1.4) for textual retrieval and the GNU Image-Finding tool (GIFT) for content-based image search. Unfortunately, it was not possible to finish the integration of GIFT in time. Instead, a content-based technique was implemented using Lucene. The MPEG-7 colour histogram descriptor was used to calculate low-level features for every image. This content-based low-level description was also indexed in Lucene by using binary fields. The main hypothesis concerning the empirical evaluation was to investigate whether content-based retrieval methods based on low-level descriptors can be utilised to increase retrieval performance. It is commonly known that comparing visual features is computationally more expensive than textual search. Even straightforward features like colour histograms generate large vectors that have to be accumulated and processed by the ranking or similarity function. This is arguably more complex than incorporating counts of words like in text ranking functions. For that reason, the content-based method was integrated in the query expansion procedure for the empirical experiments in 2006. This approach ensured reasonable response times and yet allowed to test the hypothesis, because the number of feature vector comparisons was small. Four monolingual English experiments were submitted in total. The index was created using a standard stop-word list and the Porter stemmer. The title and narrative fields from the topics were used to query the fields of the index and the topic description was abandoned. Terms in the title of the topics were searched in the title, description, location, and notes fields of the index and terms from the narrative topic description were queried against the description and notes. The baseline experiment *tucEEANT* was based on this configuration. All other experiments contained a pseudo-relevance feedback step, which extracted highly frequent terms from the top-20 documents in order to reformulate the initial query. The configuration *tucEEAFTI* combined textual and content-based features for the feedback step. The final score $score_{mixed}(d, q)$ (see Equation 5.1) was obtained by using a linear combination of the scores from the textual ranking $score_t(d, q)$ and the content-based similarity $score_{hist}(d, d_r)$ of the colour histograms of the retrieved documents and the histograms of the three true relevant documents d_r given in the query. The Euclidean distance was implemented as the similarity function for the reported configuration and the tuning parameter α was set to 0.3.

$$score_{mixed}(d, q) = (1 - \alpha) \cdot score_t d, q + \alpha \cdot score_{hist}(d, d_r) \tag{5.1}$$

Using the visual features of relevant images could be considered as bias to boost content-based approaches. In a real world application, these example images are usually not provided by the user. Nevertheless, it can be argued that an initial ranking generated by a text-only ranking mechanism, could easily be re-ranked by presenting the resulting images with an option for user relevance feedback. The last experiment dealt with the assumption that using relevance information can also be used to boost results of text-only image retrieval approaches. Of course this scenario is somewhat artificial, because such an user information need would consist of three parts: (1) a textual query, (2) three example images, and (3) annotations of the example images. There was an interest in comparing this scenario to the other experiments. Therefore, the most frequent terms were extracted from the captions of the examples images and the original query was expanded with these terms in experiment *tucEEAFT2*. In ad-

Corpus	Run ID	Lang	MOD	PRF	RF	MAP[7]	Rank[7]
IAPR-TC12	Cindi_Exp_RF	EN	mixed	yes	yes	0.3850	1/133
IAPR-TC12	tucEEAFT2	EN	text	yes	yes	0.2436	unknown
IAPR-TC12	tucEEAFTI	EN	mixed	yes	yes	0.1856	unknown
IAPR-TC12	tucEEAFT	EN	text	yes	no	0.1830	unknown
IAPR-TC12	tucEEANT	EN	text	no	no	0.1714	unknown

Table 5.7: ImageCLEF 2006 – Results for the Photo Retrieval task.

dition to the user relevance feedback, the pseudo-relevance feedback approach was applied as described before.

The obtained results are listed in Table 5.7. They are reported in terms of MAP and briefly described with the modality of the search (MOD), which can be either text, image, or mixed, whether the run was configured using pseudo-relevance feedback (PRF) and user (or true) relevance feedback (RF). In order to illustrate the effectiveness of the conducted experiments in comparison to the runs of other participating groups, the best performing experiment is listed in the top of Table 5.7. The analysis of the empirical evaluation confirmed the hypotheses. First, the baseline experiment *tucEEANT* was outperformed by all our other configurations. It can also be seen that using the content-based re-ranking approach based on example images, slightly improved retrieval effectiveness compared to the reference experiment *tucEEAFT*, which used only pseudo-relevance feedback. More interestingly, it was observed that the text retrieval approach which was based on user relevance feedback, performed significantly better. Table 5.7 also reveals a significant difference between the best experiment obtained with the prototype system and the top-performer. The limitation of the best experiment was that it relied on manual query generation and user relevance feedback, which explains the large gain in comparison to fully automatic experiments like ours. These results suggest that the content-based approach needs further exploration. A starting point for improvement was to incorporate other state-of-the-art visual content descriptors.

---

[7] The MAP value and the experiment ranking were extracted from [42, p. 8f].

Corpus	Run ID	Lang	MOD	RF	# RF docs	MAP	Rank[8]
IAPR-TC12	cut-EN2EN-F50	EN	mixed	yes	50	0.3175	1/239
IAPR-TC12	cut-EN2EN-F20	EN	mixed	yes	20	0.2846	unknown
IAPR-TC12	cut-EN2EN	EN	text	no	0	0.1515	unknown

Table 5.8: ImageCLEF 2007 – Results for the Photo Retrieval task.

The ImageCLEFphoto task in 2007 was a sequel to that of 2006 which used the same test collection with a few modifications in the documents and topics. In order to motivate a higher proportion of content-based approaches, the organisers removed the description field from the image captions. Another unique feature of this task was re-using the topic set, which is why returning groups were asked to not to apply machine learning approaches on the relevance assessments from the previous year. The organisers argued that re-using the topics could be omitted due to the changes in the document collection, which defined a new challenge for participating systems [68]. However, it might also be difficult to find a new set of unique queries for a small collection like IAPR-TC12, which also respects the features defined in [69]. Another important change to 2006 affected the topics, which did not contain a detailed description of the information need any more. This is another step to challenge the success of textual approaches to image retrieval and to motivate the development of sophisticated content-based image retrieval techniques. The Chemnitz retrieval group addressed this challenge by applying a thesaurus-based query expansion approach, which automatically expanded the short keyword query to a query with a pre-defined size. The content-based re-ranking approach was re-implemented and combined with the edge histogram and scalable colour descriptors from MPEG-7. The remaining system configuration remained the same. A complete description of the experiment and system configuration is given in [194].

Based on the observations from last years experiments, an automatic user relevance algorithm was implemented which used the relevance information from a given number of documents in an initial ranked list. This approach was chosen to simulate manual

---

[8] The experiment ranking was extracted from [67, p. 7f].

user relevance feedback on a ranking created with pure textual search. The results for the monolingual English experiments submitted to the ImageCLEFphoto task in 2007 are illustrated in Table 5.8. In this year, one of the experiments achieved the best performance out of 239 submissions in total. Just like the best experiment in 2006, the limitation of that experiment was the incorporated manual user relevance feedback. Due to the focus on cross-lingual experiments in the participation at ImageCLEFphoto 2007, the effects of all implemented components can not be separated. Nevertheless, it can be concluded that the simulated manual user relevance feedback strategy improved retrieval effectiveness significantly. Considering the goal of the organisers to penalise text-only approaches the runs *cut-EN2EN* from Table 5.8 and *tucEEANT* from Table 5.7 have to be compared. The effectiveness in terms of MAP shows an absolute difference of 0.0199, which indicates that the thesaurus-based query expansion strategy could not fully compensate for the absence of detailed image captions and query formulations.

In 2008, the Chemnitz retrieval group also participated in an ad-hoc image retrieval evaluation task named *wikipediaMM*. As the name suggest this evaluation exercise used a collection of images from Wikipedia and according unstructured annotations in English. The document collection had an approximate size of 150,000 images and the annotations usually contained a brief description of the image content and additional information like copyright information, which may be less relevant for retrieval purposes. Since the document collection was used for the INEX evaluation campaign in 2006 and 2007, the organisers provided additional meta-data, which was contributed by different research groups in the meantime. In particular, they supplied an image classifier with scores for 101 different MediaMill concepts and the full set of content-based feature vectors that were used to obtain these classification scores. A set of 75 topics was provided for empirical evaluation. Each of the queries contained a short title and a narrative description that defined the relevant images in more detail. In addition to that, a concept field contained the classification scores for the MediaMill concepts. Similar to the photographic retrieval tasks, a set of example images were provided in the form of links to Wikipedia. The evaluation task and its resources are described in more detail in [176]. The main purpose for the participation in the wikipediaMM task was to investigate whether using the large feature vector together with the previously used MPEG-7 low-level descriptors could improve image retrieval effectiveness. In total four experiments were submitted, where one was a text-only baseline experiment. For the text retrieval component an index using a positional stop-word filter and the Porter stemmer was created. In a second experiment the textual results were combined with the content-based re-ranking approach. The major difference to the previous experiments was the type and number of content descriptors, which were used for visual similarity comparison. Two further experiments were used to investigate the effect of query expansion techniques. Therefore, the thesaurus-based query expansion approach was compared to an experiment, in which the textual descriptions of the MediaMill concepts were used to expand the short keyword query.

Table 5.9 lists the experiments and the results in terms of MAP. From 77 experiments that were submitted for evaluation in total, the submitted runs were ranked 27, 30, 32, and 38. Both the ranks and the absolute MAP values show that there was only little variance in the results of these experiments. Considering that one of the experiments was the text-only baseline run, which was submitted to every image retrieval evaluation task, this is an unexpected outcome. The content-based re-ranking technique *cut-mix* slightly decreased retrieval effectiveness on this test collection. It is, however, not directly comparable to the other experiments, because a different feature set was used here. The thesaurus-based query expansion performed best, with slightly higher MAP values than the baseline experiment *cut-txt-a* and the according mixed-modality experiment *cut-mix*.

A large gap can be observed between the effectiveness of the best experiment and the best experiment submitted by the Chemnitz retrieval group. Even more astonishing is the fact that the top-performing run did not use any form of user relevance feedback, which was the case in most evaluations before. In the course of the evaluation of the wikipediaMM task, participants were asked to assess a small part of the topics for relevance. For most of the topics there were many relevant documents that had

Corpus	Run ID	Lang	MOD	PRF	MAP[9]	Rank[9]
Wikipedia	zzhou3	EN	text	yes	0.3444	1/77
Wikipedia	cut-mix-qe	EN	mixed	yes	0.2195	27/77
Wikipedia	cut-txt-a	EN	text	no	0.2166	30/77
Wikipedia	cut-mix	EN	mixed	no	0.2138	32/77
Wikipedia	cut-mix-concepts	EN	mixed	no	0.2048	38/77

Table 5.9: ImageCLEF 2008 – Results for the wikipediaMM task.

almost no visual similarity. This fact explains to some extent why the content-based re-ranking did not improve retrieval effectiveness in comparison to the text-only baseline run.

In 2009, the organisers retained the main focus of the ImageCLEFphoto retrieval task, but assembled a new document collection to facilitate the evaluation. This collection consisted of roughly half a million images with unstructured annotations in English. The collection was provided by the Belga Press Agency, a news agency from Belgium. Whether a priori information about the query clusters helps to increase diversity in image rankings was a particular research question the task organisers aimed to address in 2009 [138]. Therefore, they created two sets of queries, 25 topics referred to as *Query Part 1*, containing a cluster title and cluster description, and a further 25 topics tagged as *Query Part 2* without any cluster information. Participating groups were encouraged to make use of the additional cluster information to obtain a diverse result set. Although the queries were derived from actual search logs, the log files were not distributed to the participants. The target metric of the empirical evaluation focused on diversity, i.e. the organisers ranked systems according to the F_1-measure comprised of cluster recall and precision at rank position 10 [138].

At that time, the Xtrieval framework was ready to be used with different text ranking models from the Terrier toolkit. Due to the limited number of experiments that could be submitted for evaluation, only two different text-based approaches were compared.

---

[9] The MAP values and the experiment ranking were extracted from `http://www.imageclef.org/2008/wikimm-results#chemnitz`, retrieved on March 1, 2012.

Corpus	Run ID	Lang	MOD	PRF	Ranking	F_1	MAP[10]	Rank[10]
Belga	lig2_tct_txt	EN	text	yes	-	0.6393	0.5064	1/84
Belga	cut2_t_txt	EN	text	yes	BM25	0.5929	0.5041	4/84
Belga	cut2_tct_txt	EN	text	yes	BM25	0.6420	0.4821	10/84
Belga	cut1_t_txt	EN	text	yes	Lucene	0.5436	0.4468	17/84
Belga	cut1_tct_txt	EN	text	yes	Lucene	0.5717	0.4386	19/84

Table 5.10: ImageCLEF 2009 – Results for the Photo Retrieval task.

The target cluster fields were considered as a second variable parameter in the submitted experiments, which resulted in four different experiment configurations. A standard stop-word list and the Porter stemmer were applied to create the text index. A standard pseudo-relevance feedback strategy was applied by extracting the 10 most frequent terms from the top-5 documents in an initial ranking for automatic query expansion.

The results are presented in terms of MAP in Table 5.10. In line with the previous result discussion, the best experiment of the ImageCLEFphoto retrieval evaluation task 2009 was included for reference. Note that MAP was not the target metric of the organisers. Thus, the official evaluation resulted in a different ranking of participating systems. The ranking of our experiments shows that BM25 outperformed Lucene for this test collection. It is also obvious that the described experiments achieved strong performance in terms of MAP. The used diversity metric F_1, however, showed a different picture. Of the 84 experiments submitted for evaluation our runs were ranked 35th, 56th, 61st, and 64th. In fact, this is not a bad result, because the diversity was not addressed at all and our experiments were pure text retrieval runs.

A more interesting observation can be made when comparing the effect of using or omitting the cluster titles in *Query Part 1*. In terms of MAP, using the cluster titles to formulate a more diverse query results in lower performance both for BM25 and Lucene. But the opposite is the case when the runs are ranked according to the diver-

---

[10] The MAP values and the experiment ranking were extracted from `http://www.imageclef.org/2009/photo/results`, retrieved on March 1, 2012. Note that the experiment ranking is ordered by MAP.

sity metric F_1. At first glance this observation may appear odd. But given the definition of the F_1-measure [10, 138] it becomes clear that F_1 and MAP are not even remotely comparable. This is due to the fact that the MAP metric assesses the complete ranking of documents, whereas the present F_1-measure only accounts for the top ten documents. A detailed analysis of the metrics underlying F_1 revealed that all the presented experiments achieved a similar P@10, but the experiments which considered the cluster titles achieved markedly higher values for the diversity score CR@10, which is incorporated in F_1.

In order to conclude this section on image retrieval evaluation, general observations and findings are summarised. A central technical problem with searching visual objects is the size of features, which describe the image contents, and the complexity of similarity or distance measures. For that reason most of the content-based image retrieval approaches combine textual search using image annotations and content-based retrieval methods. Typically a ranking by textual annotations is used to constrain the number of feature vector comparisons. But this approach depends on the presence and quality of textual annotations as well as the effectiveness of the ranking algorithm. In the experiments discussed here, content-based features were used for re-ranking. The experiments in various evaluation settings demonstrated that the success of the multi-modal approach depends on the content-descriptors, i.e. the image features, and the visual similarity of the images in the collection. In general, there were always text-only experiments that achieved retrieval effectiveness similar to the more sophisticated multi-modal approaches. Another stream of empirical results suggested that user feedback increases image retrieval performance dramatically both for textual retrieval approaches and multi-modal techniques.

5.4.3.2 Question Answering on Speech Transcripts

A Question Answering system tries to find correct answers to questions that were submitted to the system in natural language. Traditionally, the systems use high quality, textual data sources to obtain answers. The QAST (Question Answering on Speech Transcripts) task at CLEF 2008 was designed to extend this challenge by using transcribed audio documents as resources. This is motivated by the fact that nowadays, a lot of potentially interesting information appears in multimedia documents like broadcast news, meetings, or telephone conversations [177]. In contrast to typical textual documents, transcripts of audio documents are far less precise in terms of syntax, i.e. there are only few or even no punctuations in this kind data. Moreover, speech is typically less fluent, which means it contains far more repetitions, restarts, and corrections. The organisers offered five different document collections covering lectures in English (CHIL corpus), meetings in English, news broadcasts in French, and debates from the European Parliament in English (EPPS corpus) and Spanish. In total ten different subtasks were offered by providing a manual and an automatically generated transcript for each of the document collections. The document collections were selected, because they covered the three most frequent classes of speech: (1) spontaneous speech, (2) prepared speech, and (3) semi-prepared speech. Two sets of documents and questions were created and distributed to the participating institutions, one for development purposes and a second set for evaluation. The development topic set contained 50 questions and the test set 100. The focus of the QAST task was on factual and definitional questions. In order to assess the systems according to these types of questions, each topic set contained about 70 percent factual questions, 20 percent definitional questions, and 10 percent NIL questions, i.e. questions that cannot be answered with the underlying document collection. Assessing the quality of Question Answering systems is more complex than evaluating standard ad-hoc text retrieval, because potential answers, besides being correct, should also be complete and short. The details of the evaluation procedure and used measures are given in [177]. In short, the participants were asked to submit up to five ranked answers to per question.

The main goal for the participation in the QAST task at CLEF 2008 was to test the extensibility of the Xtrieval framework. The general system set-up with all major components is illustrated in Figure 5.7. A Question Answering system is a combination of data mining software and an information retrieval system. This is also reflected in the presented modular system configuration. In order to enrich both the document collection and the questions, a pre-processing module that contained different state-of-the-art natural language processing tools was created. As a first step, a POS tagger, named CRFTagger[11], annotated the terms. In a second step, a Named Entity Recognizer (NER) was used to identify types of entities. The *Stanford Named Entity Recognizer* [63] was chosen, because it provided classes of entities that were closely related to those in the pre-defined classes of questions. The motivation for selecting this approach was the assumption that the system would be able to categorise the given questions based on the help of correctly recognised entities. Once the entities were known, the system could perform a keyword search on the documents in order to rank the most likely passages that might answer the question. In order to achieve this, a collection processing module was designed which used the POS and NER annotations to split the documents into meaningful passages. In this implementation the sentence splitter that was part of the POS tagger was used to obtain document passages. These passages were then indexed using the Xtrieval framework and Lucene as retrieval back-end by applying stop-word removal and Porter stemming.

The actual question answering process consists of three components: (1) a question processing and classification module, (2) the retrieval system, which ranks the passages according to a specifically formulated query obtained from the question processing module, and (3) an answer selection module that ranks the passages according to the type of question and selects any potential answer within each passage using its score. Question classification and query generation are the most crucial parts of the question processing module. The latter ensures the selection of appropriate candidate passages by means of text retrieval. In order to emphasise phrases in the question and they were submitted as phrases to Lucene to constrain the resulting passages. Question classification was then used to select candidate passages from the retrieved result set and to identify the actual answers within each passage. Both components are based on a complex set of rules and are described in detail in [113], [106]. Four experiments

---

[11] `http://crftagger.sourceforge.net/`, retrieved on March 1, 2012

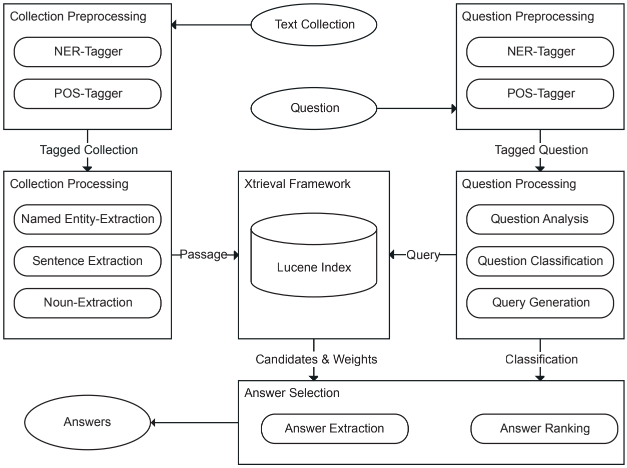


Figure 5.7: General system architecture for the QAST task at CLEF 2008, redrawn from [106].

were submitted for evaluation: two experiments for the manual transcriptions of the CHIL corpus and the same two configurations for the EPPS corpus. For the first of the two submissions for each collection the system was restricted to supply only one answer. In the second experiment a cut-off at three answers was implemented, i.e. the number of returned answer candidates could vary between zero (for an unsupported question) and three. Table 5.11 lists these experiments and the corresponding results. In order to critically assess the implemented approach, the best performing system was included in this presentation.

Table 5.11 lists the following figures: (1) the total number $\#Q$ of questions that were answered, (2) the amount of correctly answered questions $\#CA$, (3) the mean reciprocal rank MRR for all questions, and (4) the overall accuracy ACC of the systems under investigation. In general, the presented system was able to answer every fifth question correctly. In comparison to the best system *limsi1*, which achieved an overall

Corpus	Run ID	Lang	#Q	#CA	MRR	ACC
CHIL-manual	limsi1	EN	100	52	0.45	41.0%
CHIL-manual	cut1_t1a	EN	100	16	0.16	16.0%
CHIL-manual	cut2_t1a	EN	100	24	0.20	17.0%
EPPS-manual	limsi1	EN	100	56	0.42	33.0%
EPPS-manual	cut1_t4a	EN	100	21	0.21	21.0%
EPPS-manual	cut2_t4a	EN	100	23	0.22	21.0%

Table 5.11: CLEF 2008 – Results for the QAST task.

accuracy of 41 percent on the CHIL corpus and 33 percent on the smaller EPPS collection, these results appear to be poor, especially on the CHIL collection. However, the focus was to simulate a realistic question answering scenario, where a system should either return the correct answer, or nothing if it did not find any clues to answer a question. The experiment configurations *cut1_t1a* and *cut1_t4a* followed this assumption, which also explains the resulting figures with equal values for MRR and ACC. The restrictive design choice is also reflected in the proportion of NIL answers, where 53 and 60 were obtained on the CHIL corpus as well as 41 and 44 on the EPPS corpus. For the second configuration of the implemented system, about 70 percent of the questions still returned only one answer, whereas two or three answers were returned for 15 percent of the questions respectively. This fact explains the small gain in terms of MRR, when comparing these two experiments per collection. Interestingly, the difference between these experiment configurations and the best system is much smaller in terms of accuracy on the EPPS collection. In order to improve the rates of correctly answered questions for the described system, one could modify the query generation module in a way that all proper nouns are submitted to the passage retrieval engine.

5.4.3.3 Video Classification

The Chemnitz retrieval group participated in the VideoCLEF evaluation track in 2008 and 2009. In order to keep matters clear, the emphasis is put on the empirical evaluation in 2009. The VideoCLEF track 2009 offered three different evaluation tasks. The *Subject Classification Task* was an extension of the main task of the previous year, i.e. an empirical evaluation of automatically assigned thematic subject labels to videos. Two other tasks dealt with identifying narrative peaks (*Affect Task*) and finding topically related resources about the same topic, but in a different language (*Linking Task*) [115]. Because a main focus of the Chemnitz retrieval group is information extraction from videos, this review is centred on the subject classification subtask. The task re-used a dataset from TRECVid 2007 and 2008 and aimed at assigning subject labels that are broad enough to cover the topic(s) of an entire episode. Since the task was inspired by the manual work of documentation experts at the Netherlands Institute of Sound an Vision, a gold standard set of topical class labels could be used for evaluation. Participants were provided with the 46 class labels and 212 videos from TRECVid 2007 for development purposes, and 206 videos from TRECVid 2008 for testing. Although there was a huge variation, the average length of the videos was about 30 minutes. Zero or multiple labels could be assigned to each of the videos. Based on the approaches to the problem and the evaluation results of the previous year, the task was treated like an information retrieval task. In order to do so, the labels were simply interpreted as queries that can be submitted to a retrieval engine. The retrieval system ranks the video documents according to the query and the only remaining question is the definition of a document cut-off from which lower ranked documents will not be assigned with the label in question. The general workflow of the implemented system is illustrated in Figure 5.8.

In order to design the experimental set-up, a set of parameters was defined which was expected to affect the resulting performance. The most significant source of variance is the meta-data provided to describe the videos. There were two types of data available: (1) automatic speech transcripts asr and (2) archival meta-data descriptions md. Given these sources of annotations SF, it was decided to use both of them separately and also in combination. Another crucial parameter was the threshold level, which separated the documents that should receive a label and those that should not receive this label. An automatically estimated threshold would probably perform better than a

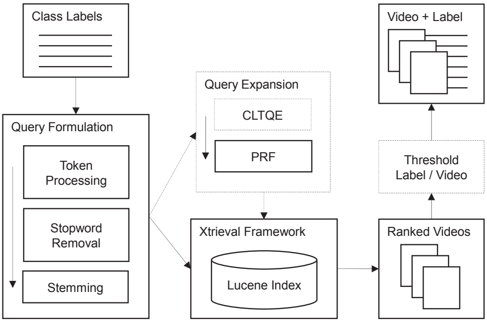


Figure 5.8: Abstract system design for the VideoCLEF classification task in 2009, redrawn from [105].

fixed threshold or no threshold at all. The formula given in Equation 5.2 was used to calculate an automatic threshold T_{LpD} to cut-off the video document ranking.

$$T_{LpD} = RSV_{avg} + 2 \cdot (\frac{RSV_{max} - RSV_{avg}}{D_{retrieved}}) \tag{5.2}$$

Technically, this implementation is different from defining a fixed threshold of allowed labels per document. But it achieves the same functionality in this experimental set-up, because it affects the absolute number of assigned labels directly. The given definition used two score-based measures, namely the mean score RSV_{avg} and the maximum score RSV_{max} of all retrieved documents $D_{retrieved}$ for a particular label (or query). Thus, it directly depended on the underlying ranking function, which was defined to be in the interval [0,1]. During the experiments with the development test set some labels were found to return no or only very few documents when they were submitted to our retrieval engine. In order to overcome this issue, a standard pseudo-relevance feedback (PRF) procedure and a cross-language thesaurus (CLTQE) module were implemented for query expansion. The latter was used to obtain similar English terms for the class label. The applied PRF method simply obtained the most frequent term from the top five documents of an initial ranking. Table 5.12 summarises the tested

Run ID	SF	QE	T_{LpD}	# L	MAP
cut1_l1_base*	asr	no	1	27	0.0104
cut2_l1_base	md	no	1	63	0.2003
cut3_l1_base*	asr/md	no	1	112	0.2541
cut7_l1_qe*	asr	yes	1	158	0.0904
cut8_l1_base	md	yes	1	196	0.2867
cut9_l1_qe*	asr/md	yes	1	196	0.2561
cut4_l0_qe	asr	yes	∞	1,571	0.1036
cut5_l0_base	md	yes	∞	1,933	0.4391
cut6_l0_qe	asr/md	yes	∞	2,276	0.4389
cut10_lx_base	md	yes	x	396	0.4115
cut11_lx_qe	asr/md	yes	x	482	0.4130

Table 5.12: VideoCLEF 2009 – Results for the Subject Classification task.

system configurations and the resulting performance in terms of MAP. Although using MAP as performance metric for a classification task appears to be questionable, this metric is reported here, because it was used for the official evaluation of the task [105], [115]. The total number of assigned labels L was included in the result table in order to illustrate the problem of using MAP as the evaluation metric.

Table 5.12 is organised in four sections to distinguish the used experiment configurations. All experiments that were part of the official evaluation are marked with *. The first section covers experiments that did not contain the described two-stage query expansion strategy. In addition to that, the number of labels per document (denoted in column T_{LpD}) was restricted to one. The results indicate that using only automatic transcripts results in very low performance, whereas using archival meta-data, both alone and in combination with transcripts performs significantly better.

The second section (runs *cut7_l1_qe**, *cut8_l1_base*, and *cut9_l1_qe**) shows the effect of the query expansion approach. It is obvious that it improves performance significantly when the transcripts and meta-data descriptions are used alone. But there is no gain when both types of annotations are used together. In fact, the comparison of the total number of assigned labels (# L) for experiments *cut8_l1_base* and *cut9_l1_qe**, reveals that both experiments assigned the same number of labels, but run *cut8_l1_base* achieved a higher precision.

The last two sections of Table 5.12 show the effect of the restriction threshold T_{LpD}. A value of ∞ indicates that there was no restriction to the number of labels for a single documents, i.e. each video could end up being tagged with every class label. One can clearly see that this approach increases performance in terms of MAP for all our experiments. It is, however, questionable whether assigning an average of 10 out of 46 labels to each document would be useful in a real-world scenario. The last section of the result table includes two experiments that used the automatic threshold approach (see Equation 5.2) to restrict the number of most likely wrongly assigned labels. When these experiments are compared to the corresponding reference experiments (*cut8_l1_base* and *cut9_l1_qe**) it can be observed that the automatic threshold achieves almost the same precision (MAP) as the non-restricted reference experiments, but it reduces the amount of assigned labels dramatically. In general, one can conclude that for this test collection and the given task, automatic speech transcripts cannot compete with manually created annotations in terms of classification performance.

Concluding the video subject classification evaluation experiments, one can state that treating the task as an information retrieval problem appears to be the most successful strategy. A promising approach to automatically restrict the number of subject labels for each video document was introduced and compared to assigning only a single label to every document, as well as to an exhaustive non-restrictive approach. Whether automatic speech transcripts or archival meta-data are the better source for classification seems to depend on various aspects of the data sets under investigation. Factors that might affect the performance in favour of one source of annotation could be the number of the class labels, their semantic relatedness, as well as the size of the document collection.

5.5 Summary and Implications

The present chapter tied in with the discussion of open-source IR toolkits and it showed that two particular toolkits, namely Apache Lucene and Terrier, can be combined in order to exploit the advantages of both. The design and architecture of the Xtrieval framework exploited the rich text transformation architecture in Lucene and the variety of ranking and feedback models provided by Terrier. Using Xtrieval to design and test retrieval experiments against several empirical evaluation tasks that used different types of textual document collections demonstrated the flexibility of the proposed method and its implementation. In addition to this flexibility, the availability of various toolkits in Xtrieval allowed the selection and adjustment of all system components according to the respective retrieval task. This approach resulted in very strong performance in terms of retrieval effectiveness at various evaluation tasks. A summary of all participations presented in the previous section, together with corresponding experiment configurations and the most important observations is provided in Table 5.13. It illustrates that using different system components and configurations improves the retrieval effectiveness and that the data fusion approach can be used to increase these results further.

The empirical results on the evaluation tasks presented in Section 5.4 demonstrated that the Xtrieval framework has been considerably advanced in terms of retrieval effectiveness. In 6 out of the 10 participations, the results submitted by the Chemnitz retrieval group have been ranked among the top-4 experiments and 3 of these experiments returned the best results overall. Each of these top submissions were achieved for different kinds of evaluation tasks which highlights both the flexibility and the quality of the Xtrieval framework.

Many system components and their parameters have to be tuned in order to maximise the overall search effectiveness for a specific task. It is clear that in a setting of an ad-hoc retrieval evaluation campaign, it is not possible to investigate all reasonable values of component and system parameters. This is one of the main reasons why IR evaluation typically ranks systems (or experiments) instead of groups of experiments that compare different instances of system components. This fact limits the usefulness of IR evaluation, because it hides an entire universe of valuable information and respective effects from scientific investigation. In the end of Chapter 2 recent approaches to evaluate IR system components for different search tasks have been presented. The following Chapter 6 picks up on this area of research and illustrates how the Xtrieval framework can be used to contribute a series of experiments to investigate the effect of, and relationships between IR system configurations.

Task	Year	Experiment Configurations	Observations & Findings	Ref.
Domain-Specific	2006	The basic system was Lucene. An English stop-word list and the Porter stemmer were applied. A sophisticated PRF approach that was based on clustering either terms or documents was the focus of the evaluation	The results showed that the implemented PRF strategy slightly improved the performance. The best experiment was ranked 5/8 and its relative difference to the best run was 17.5%	[102] [104]
	2007	The basic system set-up was the same as 2006. This time a standard PRF approach was applied and the focus was on the structure of documents	Omitting the structure of the documents resulted in almost doubled performance. The best experiment from Chemnitz was ranked 7/15.	[109]
	2008	First use of Xtrieval with the general set-up like 2007. Two indexes using the Porter and Krovetz stemmers were created for data fusion.	The data fusion slightly improved the effectiveness. The best Xtrieval experiment was ranked 1/12.	[111]
Ad-Hoc (TEL)	2008	First use of Xtrieval. A default stop-word list and the Porter and Krovetz stemmers were applied. Ranking with Lucene.	The data fusion run performed best and multilingual querying underperformed. Best Xtrieval experiment was ranked 4/37.	[107]
	2009	The basic system was the same as 2008, but data fusion of the ranking models of Terrier (BB2) and Lucene was used.	The data fusion of the ranking models improved retrieval performance over single model configurations. The best Xtrieval run was ranked 2/46.	[103]
Image Retrieval	2006	Bi-modal search based on Lucene. An English stop list and the Porter stemmer were applied for text captions. MPEG-7 descriptions were used for re-ranking in a standard PRF method.	The content-based re-ranking did not improve the performance over the text-only approach. The best experiment achieved only average performance in comparison to other participants.	[193]
	2007	The general system set-up was the same as for 2006. The PRF method was based on explicit feedback from example images.	The explicit relevance feedback from example images resulted in very strong performance. This run was ranked 1/239.	[194]
	2008	First use of Xtrieval with the exp. configuration of 2006. The processing of the content-based comparison was realised in PostgreSQL.	The content-based re-ranking slightly improved the performance. The best Xtrieval run was ranked 27/77.	[195]
	2009	Xtrieval was used for text-only experiments. An English stop list and the Porter stemmer were applied. Ranking models from Terrier (BM25) and Lucene were compared.	BM25 performed significantly better than Lucene. The best Xtrieval run was ranked 4/84.	
Question Answering	2008	Xtrieval was used for retrieval. Passage ranking was done with Lucene. A POS tagger and NER was applied on the test collection. A hand-set of rules was used for the classification of questions.	In general the approach did not work well. crafted Only definition questions were answered better than by other systems.	[106]
Video Retrieval	2009	The classification problem was interpreted as retrieval task. Xtrieval was used with Lucene as core. An English stop list and the Porter stemmer were applied. An automatic cut-off threshold was implemented. A bi-lingual thesaurus was used for QE and PRF.	The bi-lingual thesaurus and the PRF approach improved the performance in terms of MAP. The automatic cut-off threshold increased the rate of correct classifications. The best Xtrieval experiment was ranked 1/6.	[105]

Table 5.13: Overview of empirical IR experiments conducted with Xtrieval from 2006-2009.

6 Component-Level Evaluation and Empirical Analysis

Both the utility and the adaptability of Xtrieval have been demonstrated for a variety of traditional evaluation tasks. The present chapter aims to provide insights into a much more detailed analysis of empirical evaluation results in IR. First, the goals and findings of a closed workshop on *Reliable Information Access (RIA)* in 2003 will be presented and discussed. Bearing in mind the key observations regarding topic and system variance, which were analysed and published by the organisers of this workshop, the motivation and experimental set-up of grid experiments will be discussed by using the *Grid@CLEF* pilot task as an example. Second, a discussion of the results that were obtained using Xtrieval for the participation in *Grid@CLEF* will be provided. Finally, the ability of Xtrieval to reflect the state-of-the-art in IR research is combined with the grid experiment approach to component-level evaluation. Desirable properties and general principles for transparent and reliable assessment of important aspects in complex IR systems are derived based on the findings from these experiments.

6.1 Topic-Level IR Evaluation

The formulation of an information need is highly variant across different users of search applications. Empirical IR evaluations need to reflect this in the generated test collections. The TREC ad-hoc test collections emphasises the variance in topics well,

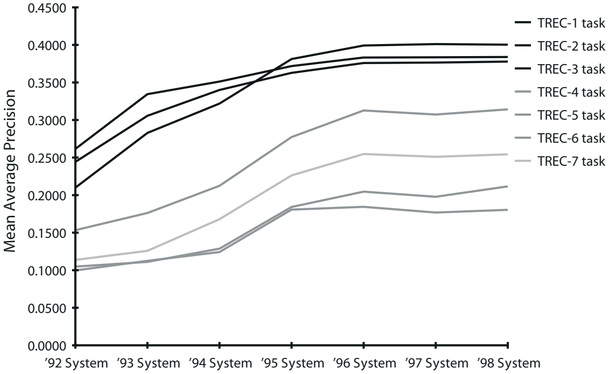


Figure 6.1: Retrieval performance in terms of MAP for different versions of the SMART system (1992-1998) on the TREC-1 to TREC-7 test collections, redrawn from [75, p. 616].

given the large number of topic sets that have been developed over the years. In the first years of TREC, the focus of research was to improve IR effectiveness on such a given set of test queries. But it has been pointed out in [75] that, within a period of seven years, the average retrieval performance reached a plateau. Figure 6.1 demonstrates this effect, by plotting the retrieval performance of the SMART retrieval system, which achieved high retrieval effectiveness across the first years of TREC. This *ceiling effect* motivated the IR community to investigate current state-of-the-art systems more thoroughly and led to the organisation of the *Reliable Information Access* workshop in 2003. A more recent study [9] which was briefly introduced in Chapter 2 found that even in the past decade no considerable improvements have been recorded on the TREC ad-hoc test collections.

Figure 6.2 illustrates the effect based on a literature review containing 32 publications, which used the TREC-8 ad-hoc test collection. An examination of this illustration suggests that the retrieval performance plateau is still present today. The authors of the latter study observed that although these results suggest that there has been no measur-

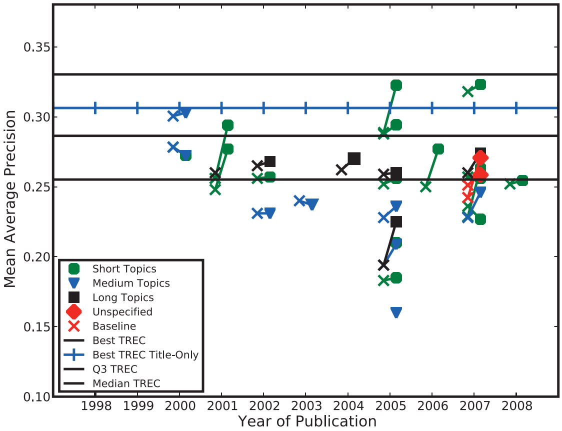


Figure 6.2: Retrieval performance in terms of MAP extracted from 32 publications (1998-2008) in comparison to original TREC-8 ad-hoc results, redrawn from [9, p. 605].

able progress for more than a decade, almost all of the analysed publications claimed that their presented approaches improved retrieval performance. This observation led them to the conclusion that this unexpected trend might be due to a general problem with the current methodology in empirical IR evaluation and the publication of research articles. To overcome this problem it was suggested that archives be created to give the opportunity for long-term evaluation. This would provide reasonable baselines for future experiments and allow any progress in effectiveness to be monitored. The author of this work fully agrees with this opinion. The outcome of the empirical evaluation supplied in Chapter 7 could serve as a meaningful addition to the efforts to overcome the performance ceiling issue.

A common explanation for the performance ceiling effect is the variance across topics, i.e. as stated in [75, p. 616]:

> “*[...] retrieval techniques that work well for one topic do not work well for others, leaving no improvement in performance on average*.”

Thus, a central goal of the RIA workshop was to investigate the topic variance by comparing the effectiveness of different state-of-the-art IR systems. Seven research groups with both theoretical expertise and ready-to-use IR systems were brought together for a period of six weeks. In order to be able to manage the complexity of analysing a large number of experiments in detail, the researchers decided to concentrate their investigation on techniques that most of the systems had in common. Prior to the actual workshop they agreed to focus their analysis on two key aspects of system and topic variability. Firstly, they conducted a massive comparative failure analysis across systems and topics to uncover where and why typical state-of-the-art systems fail, and to identify concrete problems which future investigations could focus on in order to improve retrieval effectiveness. Secondly, a series of strictly controlled experiments were carried out in order to allow a thorough analysis of pseudo-relevance feedback techniques (PRF). The reasoning behind this decision was based on the assumption that these techniques depend greatly on the actual formulation of an information need and could be highly correlated to variance across topics for this reason. Which elements of the IR evaluation process contribute to the variability of system effectiveness across topics will be discussed in the next subsection.

6.1.1 Types of Topic Variance

The variability of retrieval performance across different topics is affected by most of the elements of the evaluation process. A selection of possible factors that contribute to the difficulty of separating the pure topic effect was discussed in [75, p. 616f]:

- Variation in the average precision of the best performing system
- Variation in the performance across topics for a given system or system variant

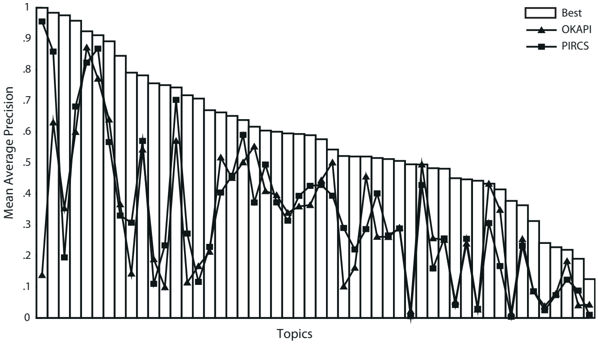


Figure 6.3: Variation in topic performance (AP) for three systems (Best, OKAPI, PIRCS) compared on the TREC-8 ad-hoc test collection, redrawn from [75, p. 617].

- Variation in the performance across topics of the effectiveness of particular system components, such as a PRF method
- Variation in the performance across topics due to features of the test collection

The first two problems can be observed for almost every evaluation task. Figure 6.3 [75, p. 617] shows the performance of three systems, which participated in the TREC-8 ad-hoc evaluation task. It illustrates that the performance of the best system varies from very high average precision (left) to values that are close to zero. A second observation is that the variance across topics does not seem to correlate well for different systems. In fact, the three depicted systems do not seem to agree on the difficulty of the topics from the TREC-8 ad-hoc test collection. This observation was one of the key aspects which provided the impulse for the experiments of the RIA workshop. An understanding of why systems do or do not agree on the ease or difficulty of a topic could allow the development of models that decide in advance, which system or model should be used given a particular topic. Despite a lot of researchers attempting to tackle this problem, no solution has been found thus far.

Further sources of variance across a set of topics can be illustrated by using a recent study on the limits of retrieval effectiveness [46]. The authors of that study conducted an experiment using a best match ranking algorithm (BM25), in which they investigated the effect of query length and query term selection on retrieval effectiveness. Their goal was to find optimal query formulations from complete information needs, i.e. using the description and narrative fields from TREC ad-hoc topics. A greedy algorithm that used relevance information was implemented to find the best query terms for a query with a fixed length. They compared the performance of this algorithm to the ability of a number of IR experts to find such an optimal query formulation. Their most surprising finding regarding topic variance was the following: the upper limit, i.e. the highest retrieval effectiveness measured in MAP, for optimal query extraction using their greedy algorithm, was about twice as high as using the entire information need as a query, or letting an average user (here: IR experts) decide which query terms should be used. Using their greedy algorithm to extract query terms from the entire information need, they were able to show that almost perfect retrieval performance can be achieved on small and medium sized TREC ad-hoc collections.

Figure 6.4 shows two important effects, by plotting retrieval performance in terms of MAP against query length. The illustration is based on two types of query formulation strategies: *SYSTEM LIMIT* refers to finding an optimal query to a system given an information need and the corresponding relevance assessments and *IN LIMIT* represents the selection of the optimal query terms from a given information need. Note the following difference between these strategies: the *SYSTEM LIMIT* defines the optimal query that contains all possible query terms, whereas the *IN LIMIT* defines a query that contains the optimal terms from an original information need. From the plots for the four collections, namely the Financial Times (FT), the Federal Register (FR), the Agence Press (AP), and the Wallstreet Journal (WSJ), a the large gap between the (theoretically) optimal system performance (SYSTEM LIMIT) and the optimal query formulation (IN LIMIT) can be observed. Based on these empirical results the au-

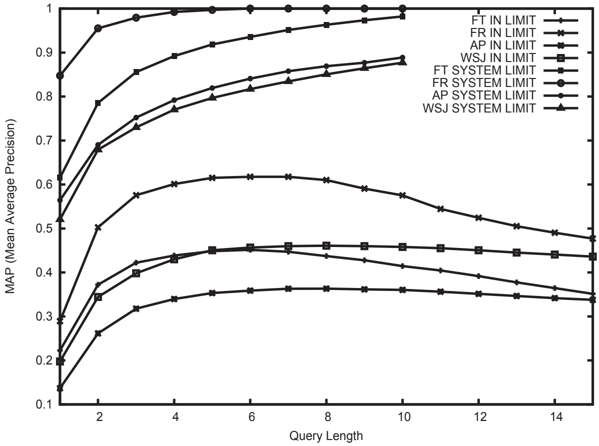


Figure 6.4: Upper limit of MAP based on two optimal query formulation strategies (*SYSTEM LIMIT* and *IN LIMIT*) on four test collections, redrawn from [46, p. 282].

thors of [46] concluded that it is more important to develop algorithms that choose an optimal query formulation strategy than optimising the IR system or its configuration.

In a state-of-the-art IR system, query formulation is one of many processes and can be considered as being a single, self-contained, component in an entire application. From a technical perspective, a query formulation algorithm is closely related to term weighting, which is part of the ranking algorithm in practice. In order to generate a document ranking that optimally satisfies the initial query, current ranking algorithms weight terms in documents based on document and collection features. In contrast to that, an *automatic* query formulation process will be trained on sampled evidence, i.e. complete formulations of information needs and respective optimal query derived from relevance assessments. The present work examines IR systems and components that do not rely on implicit or explicit relevance information. In spite of this decision, a generic framework for component-level evaluation, like the one discussed in Section 2.6, can be extended to incorporate components which adapt machine learning approaches to solve subtasks of the IR process.

6.1.2 Understanding Retrieval Variability

Two major conclusions can be drawn from the experiments that were presented in the previous section. Firstly, the performance of IR systems on a set of topics varies from query to query, but also the effectiveness of different systems or configurations on particular topics is highly variant. Secondly, the experiment presented in [46] suggests that even a particular system configuration may achieve almost optimal retrieval performance, assuming that the optimal query formulation is known beforehand. However, as can be seen from Figure 6.4 this optimal retrieval performance, i.e. MAP of 1.0, cannot be reached for all test collections. The experiment also suggests that there are other effects on retrieval performance, which must be due to the nature of the document or test collection.

The in-depth failure analysis of the controlled experiments at the RIA workshop revealed another interesting effect regarding topic and system variance. Although different systems (and different system configurations) retrieve different individual documents, they share general classes of failure documents, i.e. retrieved documents that are not relevant or relevant documents that were not retrieved. These system failures are typically due to the presence or absence of particular aspects of topics in the result sets. The researchers also found that the state-of-the-art IR technology from 2003 should be able to significantly increase retrieval performance on about half of the topics studied. The same conclusion is drawn from [46]: it is more important to discover which of the current techniques work well on which topics and why, than it is to implement new methods and models for general applications. [75]

There are two major problems in the approach to formulate optimal queries given an information need. First, the definition of this optimal query relies on relevance information about documents, which may not be available in general. Second, assuming the information about relevance of document is present or can be obtained, the optimal query may include terms that are not present in the formulation of the information need. Because of that it may be impossible to automatically generate the optimal query formulation. For these reasons the purpose of this work is to conduct a set of controlled experiments to investigate the effect of existing state-of-the-art IR system components on ad-hoc retrieval effectiveness. The reason for integrating key system components and their state-of-the-art instances is that all of those components have been shown to increase retrieval performance in laboratory experiments. However, most of these empirical investigations are limited, because researchers typically work with a single system, i.e. there is no possibility for them to separate topic and system variability in their experiments. As a consequence the results of experiments and any drawn conclusions should feature the same constraints and hence cannot be generalised unless they were validated across other types of systems *and* other document collections.

6.1.3 Implications for Component-Level Evaluation

Retrospective analysis of empirical IR evaluation experiments has shown that results are subject to variance of different kinds, although they are usually taken on average. From all types of variation, the difference between topics is the most crucial for evaluation, because it affects summary measures like MAP the most. In previous evaluation experiments, systems were treated as monoliths (or black boxes) without explicitly separating which components they have in common or to which extent they differ on particular parts of the complete IR process. Thus, it was almost impossible to study the difference between topic and system effects. Even if a system effect would have been identified using statistical tools, it would be of no value other than the knowledge of its presence.

One approach to component-level evaluation could be to continue the tradition of aiming to improve average retrieval effectiveness. This could be done by conducting straightforward TREC-like experimentation, but instead of comparing black box systems each instance of every system component has to be assessed individually. For components that can be configured using parameters, every (meaningful) configuration should be considered. As a direct consequence, comparing IR systems at component-level requires many experiments and may result in a very large parameter space. Assuming a simple model consisting of only two major components, where each component has only 5 state-of-the-art implementations, would result in 25 experiments in total.

In order to allow the thorough analysis of empirical results, a component-level evaluation task should concentrate on a particular set of components. The experiments at the RIA workshop focused on automatic (non-relevance based) pseudo-relevance feedback (PRF) mechanisms and produced about 105,000 data points. The test collection consisted of 150 topics, which were tested on seven different systems each running about 100 different configurations. This large number appears to be arbitrary, but in fact it can be simply explained by the two parameters that are used in most PRF techniques. Firstly, a number of (usually) top-ranked documents D, is assumed to be relevant. Secondly, a number of T terms is selected from these D documents in order to expand an initial query. Analysing the large amounts of resulting data is not only tedious, but it may also turn out to provide no answers to the targeted research questions. This was a problem that appeared during the RIA workshop, where the researchers expected to discover the reasons why particular PRF techniques fail on certain topics and do well on others. It turned out that the effectiveness of PRF is hard to analyse, because its success depends on both topics and systems. [75, p. 621]

6.2 Grid Experiments at CLEF

The Grid@CLEF pilot track was the first component-level evaluation task on ad-hoc IR test collections. In Section 2.6 the methodology of the task has been described and compared to other approaches. In this section it is described how Xtrieval was deployed to run experiments in the framework of the Grid@CLEF pilot task. The official results are analysed in detail in order to draw conclusions regarding multilingual information access. In addition to that, additional experimental results are also discussed. They were not part of the official evaluation due to the limited number of experiments that could be submitted by participating institutions. Finally, the observations from the grid pilot task are summarised in order to develop ideas on how Xtrieval could be extended to provide meaningful contributions to component-level IR evaluation.

6.2.1 Motivation and Goals

Similar to other component-level evaluation initiatives, the motivation for the Grid@CLEF task was to improve the understanding of IR technology. In particular, the organisers focused on multilingual information access and the effect of languages on the retrieval effectiveness of multi-lingual search applications, like digital libraries [61]. Modelling a framework for running controlled laboratory experiments and providing a platform for conducting large-scale evaluation by using a series of test collections are essential steps to gain these insights.

According to [61] the grid experiments were planned to be organised according to the Cranfield evaluation paradigm using existing document collections that were previously used at CLEF. The task itself was formulated as a traditional ad-hoc experiment. This abstract design investigated three major entities of the IR problem and their unknown interactions, namely *IR system component*, *language*, and *search/evaluation task*. Figure 6.5 illustrates the entities and the interaction effects in a straightforward fashion. It is commonplace in IR research that the contributions of these three entities on overall retrieval effectiveness are likely to overlap each other, as depicted in Figure 6.5. What remains unknown in classic system-level evaluation is to what extent these entities overlap and whether their contributions to retrieval performance can be combined in the sense of linear or more complex functions.

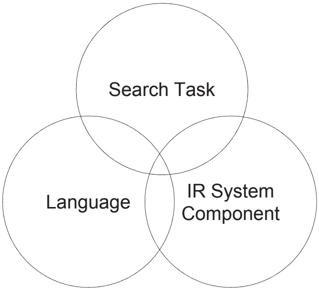


Figure 6.5: Three main entities examined at Grid@CLEF, redrawn from [61, p. 30].

6.2.2 Design of the Grid Evaluation Task

During the design of the Grid@CLEF task many IR system components were considered. See [61, p. 30] for a complete list of system components and respective subtasks in a multilingual IR system. In anticipation of the potential problem of a high-dimensional parameter space, which is both hard to visualise and difficult to interpret, the components were restricted to text pre-processing. Since these components are usually language-dependent, due to differences in syntax and morphology, they were supposed to affect multilingual IR experiments more than other components like retrieval models or relevance feedback approaches. It was noted before that the Grid@CLEF task was deployed as an ad-hoc retrieval task using the Cranfield methodology. This fact defined the first entity (the task) from Figure 6.5. The two remaining entities, language and IR system components, were also pre-defined for the task. Their effect on retrieval effectiveness was studied by the participating research groups individually. Each of these groups tested a few configurations of their own IR system using the provided test collection. The general set-up was restricted to IR system components that perform text pre-processing (see Section 3.1). The participants of the Grid@CLEF task were asked to use their component implementations in order to study the effect of different text pre-processing algorithms.

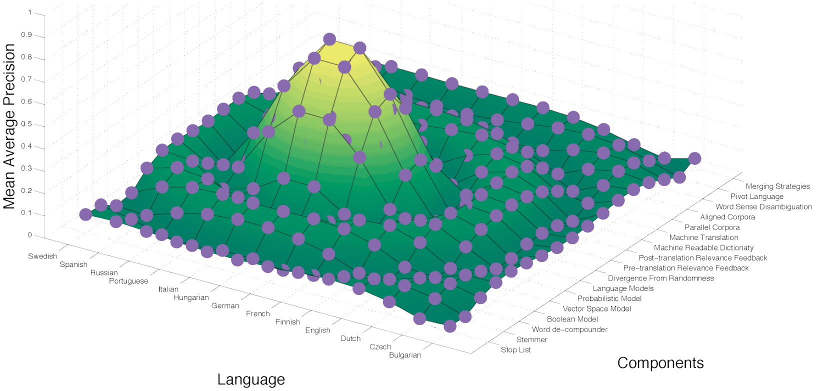


Figure 6.6: Hypothetical grid surface generated by comparison of the effect of different IR system components on retrieval effectiveness (MAP), redrawn from [62, p. 4].

The central aim of the task was described by means of the hypothetical illustration in Figure 6.6. It shows the various instances for two entities from Figure 6.5: languages and IR system components. These instances of different languages and system components, as well as the performance metric MAP, generate a “fine-grained grid of points” [62, p. 4]. Mixing different types of components on one of the axis is debatable, because “parallel” and “aligned corpora”, for instance, are part of a search task rather than IR systems. In contrast to the hypothetical illustration in Figure 6.6, the Grid@CLEF pilot task focussed on 4 specific components: languages, stop-word filtering, word de-compounding, and stemming. This is only as small fraction of the 17 components illustrated in Figure 6.6. In this work, 10 out of these 17 components are examined with consideration of various configurations for stemming, ranking, and PRF individually.

The methodology of the Grid@CLEF task was briefly described in Section 2.6. The organisers decided to develop a framework for exchanging the intermediate output of components from task participants in order to allow asynchronous experiments outside of the traditional evaluation cycles. This design allowed the creation of new system configurations across participants, i.e. connecting intermediary system outputs from different participants. These *distributed* configurations can be used to create additional data points on the grid surface described above. The Coordinated Information Retrieval Components Orchestration (CIRCO) was designed and implemented to create, exchange, and integrate the output from different text pre-processing components. CIRCO required the intermediate output to be stored in well-defined XML files, which can either be distributed among task participants or archived for re-use in other evaluation tasks that focus on other aspects of IR systems. For the Grid@CLEF pilot task, CIRCO implemented a pipeline architecture of the following components [62, p. 5]:

- *Tokenizer*: separate the input documents into a stream of tokens
- *Stop-word Filter*: removing stop-words from the token stream
- *Stemmer*: stem the tokens
- *Indexer*: store the tokens together with document and collection statistics

In order to deploy the designed task methodology, CIRCO was implemented using three components: *CIRCO Schema*, *CIRCO Web*, and *CIRCO Java*. *CIRCO Schema* is an XML schema, which defines the structure and content of the XML files to be exchanged from one element of the pipeline to another. *CIRCO Web* is an on-line system that is used to define the task model, i.e. the component pipeline. It also specifies the format for the description of components and manages the exchange of the actual data. *CIRCO Java* was supplied as implementation to facilitate the adoption of the framework and lower the threshold for participation in the task. [62, p. 6]

The Grid@CLEF pilot task re-used existing ad-hoc test collections from CLEF 2001 and 2002. The evaluation task was organised in five different subtasks concentrating on European languages (see Table 1 in [62, p. 9] for detailed statistics on the document collections). A further constraint was implemented to limit the number of subtasks. The Grid@CLEF task concentrated on monolingual experiments in contrast to most other evaluation tasks in the multilingual environment. The selected test collections have been extensively studied before and re-using them for a component-level evaluation allows the comparison of the results with the existing research literature. According to the organisers, Grid@CLEF concentrated on achieving two major goals. Firstly, the participants were asked to adapt their systems to comply with the CIRCO framework. Secondly, it would have been ideal to receive as many experiment submissions as possible in order to populate the grid surface. But similar to other evaluation initiatives, there was a limit of five experiments per subtask for each participants.

6.2.3 Experiments with Xtrieval

The main objectives for the participation in the Grid@CLEF pilot task were twofold. First, as it was noted above, the CIRCO framework had to be integrated into Xtrieval in order to be able to produce the required component outputs. The second goal was to provide challenging baselines that could serve as references for future evaluations using other types of system components [59]. Since Xtrieval and the underlying core libraries Lucene and Terrier are implemented in the Java programming language, there were no technical impediments to integrate the CIRCO framework. In fact, only few changes had to be made in corresponding wrapper classes of Xtrieval. For this reason it was possible to focus on the actual experiments.

Since the Chemnitz retrieval group did not participate in CLEF 2001 or 2002, the provided document collections were unknown. This means it was necessary to analyse the document structure and implement according parser programs to be able to process the collections. Due to limited resources it was not possible to create parsers for all five collections. Instead, the focus was on the English, French, and German collections. It is important to define which parts of structured documents will be indexed, i.e. which document sections of a present collection are likely to carry the information that is essential for effective retrieval. Table 6.1 lists the field names from the document structure of the respective collections that were extracted for indexing and retrieval.

Language / Collection	Indexed Fields
DE: Spiegel 1994/95	LEAD, TEXT, TITLE
DE: Frankfurter Rundschau 1994	TEXT, TITLE
DE: German SDA 1994	KW, LD, NO, ST, TB, TI, TX
EN: LA TIMES 1994	BYLINE, HEADLINE, TEXT
FR: Le Monde 1994	CHA1, LEAD1, PEOPLE, SUBJECTS, TEXT, TIO1
FR: French SDA 1994	KW, LD, NO, ST, TB, TI, TX

Table 6.1: Grid@CLEF 2009 – indexed fields per document collection, extracted from [59, p. 573].

Lang	Stemmer	# Docs	# Terms	# Distinct Terms	Index Size (MB)	CIRCO Output Size (MB)
DE	Snowball	225,371	28.71 M	3.37 M	743	15,695
DE	n-gram	225,371	63.12 M	0.84 M	1,137	19,924
EN	Snowball	113,005	20.21 M	0.69 M	448	14,293
EN	Krovetz	113,005	20.70 M	0.70 M	477	14,293
FR	Snowball	87,191	12.94 M	1.13 M	306	7,329
FR	Savoy	87,191	13.26 M	1.24 M	316	7,323

Table 6.2: Grid@CLEF 2009 – index and CIRCO output statistics created with Xtrieval, extracted from [59, p. 573].

In order to allow the validation of the conducted results it is important to report such information in empirical evaluation as it definitely affects retrieval performance.

Table 6.2 shows statistics on the terms, i.e. the processed tokens, and the resulting storage size for the compressed output from the CIRCO framework. The CIRCO implementation generates chunk files as intermediate component output. Each chunk file contains tokens for a maximum of 1,000 documents.

Due to different types of token stream processors, and different amounts and lengths of documents in the provided collections, the resulting data is highly variant in terms of size. Moreover, the total size of the compressed output produced with CIRCO, which covers three intermediate and one final result streams, is between 20 and 30 times larger than the actual sizes of the indices. Considering the relatively small size of the used collection this is a serious problem of the approach.

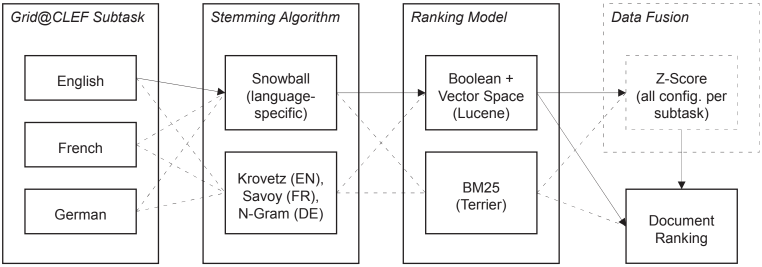

Grid@CLEF Subtask
English
French
German
Stemming Algorithm
Snowball (language-specific)
Krovetz (EN), Savoy (FR), N-Gram (DE)
Ranking Model
Boolean + Vector Space (Lucene)
BM25 (Terrier)
Data Fusion
Z-Score (all config. per subtask)
Document Ranking

Figure 6.7: Grid@CLEF 2009 – Xtrieval workflow for tested components.

As indicated above, the experiments were planned to serve as strong baselines for the Grid@CLEF pilot task. For this reason, a small grid of experiments was designed and implemented using Xtrieval. The analysis of previous empirical evaluations on test collections of similar size indicated that combining results obtained from different text pre-processing approaches and different retrieval models, usually improves retrieval effectiveness (see Section 5.4). Due to the limitation of five experiments per subtask, it was decided to investigate two state-of-the-art stemming techniques and two widely used ranking models as well as one data fusion experiment per subtask.

Figure 6.7 illustrates the workflow that covers the presented experiment configurations. These experiments resemble a mini-grid consisting of 15 MAP data points. In addition to the pictured components, a standard tokenizer for European languages was used to break the text stream into tokens. Moreover, Figure 6.7 does not show the standard pseudo-relevance feedback method, which was used for the four basic experiments with alternating stemmers and ranking models. The data fusion component was implemented as result list merging and it combined the results of all four basic experiment configurations per subtask.

Run ID	Stemmer	Ranking Model	PRF (d / t)	Fusion	MAP
cut_en_1	Porter	Bool + VSM	10 / 20	no	0.5067
cut_en_2	Porter	BM25	10 / 20	no	0.4924
cut_en_3	Krovetz	Bool + VSM	10 / 20	no	0.4937
cut_en_4	Krovetz	BM25	10 / 20	no	0.4849
cut_en_5	both(s.a.)	both(s.a.)	10 / 20	yes	**0.5446**
cheshire_eng_t2fb	Porter	Log. Regression	yes	no	0.5313

Table 6.3: Grid@CLEF 2009 – Results for the English subtask, partly extracted from [59, p. 574].

6.2.4 Results and Analysis

Table 6.3 lists all experiments for the English subtask. In total the subtask received only six submissions, five of which came from Chemnitz and one from Berkeley. Regarding the defined goal of putting as many data points on the grid surface as possible, this result is rather disappointing.

The best configuration on the English test collection was generated with the Xtrieval framework and it achieved a retrieval performance of 0.5446 in terms of MAP. The experiment from Berkeley achieved a MAP value of 0.5313, i.e. both groups achieved similar retrieval effectiveness. The top performing configuration *cut_en_5* was the data fusion experiment, which confirms the expectation that merging results from different indices and different ranking models improves retrieval effectiveness. In fact it clearly outperformed all of the four basic experiments. Considering the two components that were tested in these experiments, it can be observed that the BM25 ranking method from Terrier performed slightly worse than the combination of the Boolean and Vector Space models in Lucene for both of the tested stemmers. Examining the retrieval effectiveness of the stemmers shows that Porter performed slightly better than Krovetz for both of the tested ranking models.

The participation in the subtask on the French test collection was also low, i.e. again only six experiments were submitted in total with an identical composition of contributors. The experimental results for the French subtask are listed in Table 6.4. Here,

Run ID	Stemmer	Ranking Model	PRF (d / t)	Fusion	MAP
cut_fr_3	Snowball	Bool + VSM	10 / 20	no	0.4483
cut_fr_1	Snowball	BM25	10 / 20	no	0.4538
cut_fr_5	Savoy	Bool + VSM	10 / 20	no	0.4434
cut_fr_2	Savoy	BM25	10 / 20	no	0.4795
cut_fr_4	both(s.a.)	both(s.a.)	10 / 20	yes	0.4942
cheshire_fre_t2fb	Snowball	Log. Regression	yes	no	**0.5188**

Table 6.4: Grid@CLEF 2009 – Results for the French subtask, partly extracted from [59, p. 574].

the best experiments in terms of retrieval effectiveness as measured by MAP came from Berkeley. Again, from the experiments conducted with Xtrieval, the data fusion experiment achieved the best MAP value of 0.4942. Thus, the initial hypothesis that merging results from different components improves performance, was also confirmed on this test collection.

Analysing the effect of the tested components led to rather inconclusive interpretations, especially for the stemmers. While the implementation of Savoy performed slightly better than the Snowball implementation for French when it was used with the Boolean and Vector Space models from Lucene, the opposite is the case when BM25 was used for ranking. BM25 did slightly better than Lucene's ranking model on the index that was stemmed with Snowball and it clearly outperformed the ranking of Lucene on the index that was based on the stemmer by Savoy. These observations differ from those that were made on the English subtask, but with basically identical system configurations. This indicates that the language, or the test collection, affects the retrieval effectiveness of system configurations in a way that makes it hard to predict, which configuration should be used for optimal performance. Data fusion from different system configurations seems to be the only aspect which is independent of the test collection in the Grid@CLEF experiments, since these runs always outperformed any other configuration created with the Xtrieval framework.

Just like the English and French subtasks of Grid@CLEF, the German subtask received only six submissions from the two groups, Berkeley and Chemnitz. As can be

Run ID	Stemmer	Ranking Model	PRF (d / t)	Fusion	MAP
cut_de_1	Snowball	Bool + VSM	10 / 50	no	0.4196
cut_de_2	Snowball	BM25	10 / 50	no	0.4355
cut_de_3	N-gram Decomp.	Bool + VSM	10 / 250	no	0.4267
cut_de_4	N-gram Decomp.	BM25	10 / 250	no	0.4678
cut_de_5	both(s.a.)	both(s.a.)	both(s.a.)	yes	**0.4864**
cheshire_ger_t2fb	Snowball	Log. Regression	yes	no	0.4002

Table 6.5: Grid@CLEF 2009 – Results for the German subtask, partly extracted from [59, p. 574].

seen from Table 6.5, the best experiment came from Chemnitz. But in contrast to the two other subtasks there is a substantial gain in effectiveness compared to the best experiment from Berkeley. An examination of the other experiment configurations from Chemnitz reveals that all five experiment configurations outperformed the run from Berkeley. Assuming that both systems used identical configurations for all three subtasks, this observation suggests that there might be a notable test collection effect.

No additional experiments were made in order to explain this effect in more detail. Here, the CIRCO framework could have been used to explore the issue on component level. Given the present data, no definite conclusions can be drawn concerning the question of whether the difference between the results was due to a particular component in one of the two systems, or due to particular features of the test collection, i.e. topics that favoured results from Chemnitz, or a combination of both effects.

The results indicate that the implemented n-gram decompounding algorithm, which was specifically designed to handle German compounds, did slightly better than the Snowball implementation for German when using Lucene for ranking, i.e. the ranking is based on a combination of the Boolean and Vector Space ranking models. For the BM25 ranking model provided by Terrier, the Snowball algorithm was clearly outperformed by our custom n-gram decompounder. Comparing the effectiveness of the ranking models, the results show that BM25 is superior to the combination of Boolean and Vector Space models on this test collection.

Three key observations can be made from these experiments:

- Firstly, there was only little variance in terms of retrieval effectiveness between the BM25 and the combination of Boolean and Vector Space ranking methods, when using the Snowball framework for stemming on the Grid@CLEF test collections. This observation indicates that the two ranking approaches provide robust results for retrieval of small test collections in English, French, and German.
- A second aspect is related to the other stemming approaches that were tested. For languages with more complex morphology, like German and French, our results indicate that custom stemmers work better than the generic Snowball framework. On these test collections the variance between the two tested ranking methods (BM25 vs. Boolean and Vector Space) was much higher and in favour of BM25.
- The third observation was, that combining the four tested configurations in a late result list fusion approach based on the Z-score operator, always improved retrieval performance on the given test collections. This suggests retrieval performance can be improved, if storage and computational restrictions are not relevant for a specific search application. It can be realised by the creation of separate indices, using generic and specific stemming algorithms, querying those with different ranking models, and fusing the results into a final document ranking.

6.2.5 Lessons Learned

Besides the empirical analysis of the results of the system configurations submitted to the Grid@CLEF pilot task, further conclusions can be drawn from the design and implementation of the component-level aspect. The most apparent issue of the Grid@CLEF task was a lack of participation, which is fatal for the central goals of this type of evaluation. Two major reasons against the implementation of a common API for component-level experiments were identified in the course of the evaluation of the Grid@CLEF task [62]:

- First, the research groups were distracted from their main objectives in order to integrate CIRCO into their systems. Since the CIRCO framework was implemented for easy integration with Lucene, which is also implemented in the Java programming language, the burden to incorporate the framework in other IR systems was considerably larger. This resulted in a bias favouring those participants with systems based on Lucene, because they did not have to spend a lot of time on the integration of CIRCO.
- A second issue was the size of the intermediate output in terms of storage capacity and computation time. Compared to a standard ad-hoc IR experiment, which typically requires about two to three times the storage size of the test collection for an arbitrary large number of experiments, the design of the Grid@CLEF task took up to 30 times the size of the collection per experiment.

In spite of the unfruitful results obtained from the Grid@CLEF experiments, they provided further insights as to which specific problems future component-level tasks should concentrate on. It has already been pointed out in [75] that organising large-scale experiments which focus on specific components of IR systems is a challenging task. Moreover, a lot of unanticipated impediments may appear throughout the empirical evaluation process, just like the storage issue of Grid@CLEF. Therefore, it can be concluded that a well-designed experimental set-up for component-level evaluation should solve elementary problems prior to the actual evaluation. The key challenges from previous initiatives are the following:

- *Define key system components and their state-of-the-art instances*
  Both the Grid@CLEF task and the RIA workshop clearly defined which components should be examined by empirical evaluation. This is an important step in

determining the scope of the evaluation.Another crucial factor for component-level evaluation tasks is to investigate competitive state-of-the-art component implementations. The organisers of the RIA workshop used an invitation-only policy to restrict the participation to systems with a demonstrated track record in previous evaluation campaigns. This abstract selection scheme did not guarantee that each of the systems' components reflected the state-of-the-art, although it is very likely that they did. In contrast to that, Grid@CLEF had no restrictions. But the task defined a fixed component-level workflow which might have put off research groups whose systems did not fit the given workflow.

- *Formulate specific challenges to investigate*
  Motivating participation in new challenging search task is not a problem. It also explains the shift from traditional evaluation to more specific and diverse search problems [124]. A description of studies that supplied empirical evidence for only little, or even no, progress on traditional IR collections in the past decade was provided. Assuming that tools exist to increase performance on these test collections, or the underlying general search tasks, there was apparently not enough motivation to lead these new findings back to traditional test collections. Thus, a key issue for transparent component-level evaluation is to stimulate participation by defining new interesting goals. These goals have to be both challenging and specific enough to inspire the interest of the research community.

- *Define the level of detail for result analysis*
  Traditional IR evaluation concentrates on general retrieval effectiveness and thus is dominated by summary metrics like MAP or NDCG. In contrast to that, component-level evaluations focus on particular aspects of IR systems. Thus, it is necessary to study particular system or component failures in the greatest possible detail. This allows the identification of possible systematic failures, which – if corrected – will automatically increase retrieval effectiveness on average as well. Again, the actual level of detail for the evaluation depends on the design of the experiment, i.e. the search evaluation task itself.

- *Organise the distribution of data and code*
  Another important challenge for component-level evaluation in distributed scenarios are protocols for exchanging data or programming code. RIA used a simple but effective protocol by bringing both the experiment data and the system code into one place. In 2003, this was likely to be the most cost effective way. Grid@CLEF attempted to exchange component outputs through XML, but the redundancy of the intermediate output resulted in extremely large files. As a result, actual exchange of component-level output between participating groups did not happen. From this it follows that the decision as to whether exchange data or programming code, or to implement a combination of both, depends on the evaluation task. More specifically, it is dependent on the tasks which are completed by the components under examination. As a consequence, exchanging intermediate component output requires simple programming interfaces. In principle, synchronous protocols are possible, but for the sake of comparability in the future, protocols should allow asynchronous exchange of data.

- *Preserve the results in full detail*
  Current IR evaluation methodology also needs to deal with preservation of empirical experiments and their results. However, it is commonplace in IR research that experimental results are stored by the organisers of evaluation tasks. Experimental descriptions, summary figures about the results, and respective interpretation are published in research publications. This methodology is a major handicap for preservation and long-term analysis. Since the issue has already been pointed out for traditional IR evaluation, it is important to design and establish simple but powerful meta-data descriptions in the IR evaluation domain. This is particularly important for evaluation at the component-level. Because detailed experiment descriptions can serve as powerful tool to select appropriate baselines for the assessment of future system components.

The Xtrieval framework provides solutions for the conceptual problems of automated evaluation at component level. It facilitates an architecture to address the organisational issues. Its meta-level design for accessing and combining different state-of-the-art retrieval toolkits allows fine-grained empirical studies at the component-level. Incorporating the findings of previous initiatives, Xtrieval can be deployed to run a large series of grid experiments that provide a better understanding of the orchestration of components in modern IR systems. Xtrieval allows to focus both on overcoming practical limitations and addressing methodological issues without gathering research groups (RIA workshop), or implementing additional data exchange protocols (Grid@CLEF). The required resources are TREC-like campaigns that produce very valuable test collections and IR toolkits like Terrier, Lucene, Lemur, and others which reflect the state-of-the-art of specific aspects in IR systems.

6.3 Increasing Clarity in IR Evaluation

A central goal of Xtrieval is to make effectiveness evaluation more transparent. The previous section presented key challenges that need to be addressed in order to achieve this. These problems can be separated in two major groups. The first are organisational problems like exchanging the data that is needed to conduct empirical evaluation, which includes test collections, test systems, evaluation results as well as experiment designs. The second group of obstacles contains practical limitations, such as the number of experiments that can be submitted in traditional evaluation tasks, or the special focus of publicly available toolkits, which often result in IR systems combining highly-effective and innovative system components with other outdated components all into a single system.

It will be considered next, how Xtrieval can be used to address the challenges presented in Section 6.2.5 and which specific questions in the field of ad-hoc evaluation can be studied. First of all, we have to decide what relevant system components are, and which of their instances, i.e. implementations of different models for a particular component, reflect the state-of-the-art. Chapter 3 discussed central system components like text pre-processing and retrieval models, that are accessible through Xtrieval. Covering the complete state-of-the-art even for a single component of modern IR systems is a task that requires a large amount of resources. For this reason, Xtrieval integrates existing retrieval toolkits like Terrier and Lucene. Chapter 4 demonstrated that a large number of IR libraries have been developed in order to solve specific tasks and underlying problems. However, the specificity of these tools does not allow to take an holistic view on the technology. Xtrieval aims to fill this gap by deploying Terrier and its variety of ranking models as well as by integrating Lucene with its powerful framework for text pre-processing. This combination enables us to thoroughly assess the effect of state-of-the-art implementations for indexing and retrieval on retrieval effectiveness. The actual selection of component instances is presented in Chapter 7.

The formulation of specific challenges for component-level evaluation tasks is the next argument. This problem has to be tackled by the community as a whole and not by individual research groups. Nevertheless, there are a few arguments that deserve attention. For this reason the present empirical analysis emphasises these. The most dominant problem for component-level evaluation is the limitation of experiments that can be submitted for evaluation. This restriction has its roots in the traditional assessment strategy, which was inherited from the Cranfield paradigm and greatly enhanced through the TREC experiments. In this model, obtaining relevance assessments involves a great deal of human resources and is therefore expensive. Using 50 test questions (or topics) per task was shown to provide stable rankings of experimental systems at reasonable cost. However, this may not be the case for evaluation tasks that target at discriminating between thousands of different system configurations in order to create a better understanding of different component implementations and component interactions in modern IR systems. The Million Query track at TREC [2], [37] demonstrated that larger amounts of topics – in the order of thousands per task – can be evaluated with reasonable effort. The success of this track should motivate the IR community to create a similar track that conducts a deep pooling strategy, i.e. assessing as many documents as possible per topic. Obviously, it is out of the scope of this work to design and conduct the task here. Nevertheless, the experiments in the following Chapter 7 will evaluate the number of encountered non-evaluated documents with respect to the number of contributed experiments from different IR system configurations.

A particular problem for component-level evaluation across many research groups is the lack of a standard protocol for exchanging component output. Although, Xtrieval was not designed to solve this issue, the present experiments show that all that is needed already exists. In Chapter 4 a number of currently available IR toolkits and libraries have been compared. This analysis demonstrated that only a few frameworks are widely-used in the IR community. For this reason they are enhanced with new features and latest findings from research more frequently. Xtrieval integrates these libraries, namely Lucene and Terrier. But how does this relate to the lack of protocols for exchanging component-level output? First of all, it is necessary to consider which components actually need this kind of data flow. The RIA workshop focused on pseudo-relevance feedback approaches. This required the exchange of ten to hundred documents or terms between systems. Thus, there was neither a need for a complex protocol nor did it demand any architecture to manage the data flow. In contrast to that, the Grid@CLEF pilot task aimed to compare text pre-processing components during index creation. Here, the CIRCO framework served as the protocol, but due to the asynchronous nature of the protocol large amounts of data had to be created and exchanged. In retrospect, it might have been better to rely on widely-used frameworks like Lucene and Terrier and transfer the exchange protocol to an abstract meta level. Both Lucene and Terrier provide powerful and yet simple protocols to create and access their underlying index structures. With the help of Xtrieval both can be used through one common programming interface. This brings up a simple idea to mitigate the data explosion problem with CIRCO: creating index structures only for selected text pre-processing components, but for each of them separately. A selection is necessary to exclude trivial components like white space tokenizers, stop-word or other simple filters. These are better exchanged in the form of the few lines of programming or pseudo-code they need, instead of exchanging the transformed output. In fact, it could be beneficial to implement these straightforward components as web services with free access, at least for participating research groups. Remaining components that might be large or complex, like stemming algorithms, might stay closed and therefore the transformed output needs to be preserved. Here, relying on existing libraries and their resulting output requires the least amount of human effort. Thus, it is necessary to define a simple protocol which preserves the trivial components that were used to alter and filter the document collection before and after the more sophisticated closed source components like stemmers. Altogether this would drastically reduce the amounts of data that have to be exchanged and preserved for thorough analysis of text pre-processing components.

One key problem in empirical IR evaluation is the lack of detailed result analysis. It is related to the competitive character of evaluation campaigns, where systems are ranked based on summary metrics like MAP. Published papers that deal with IR effectiveness often rely on such test collections and many of them attempt to develop new techniques that increase average retrieval performance. In contrast to that our component-level IR evaluation approach is targeted at the details of system components and retrieval effectiveness. This approach includes analysis based on traditional effectiveness figures like MAP, but its most important aspect lies in careful examination of the variance across instances and configurations of selected components (see Sections 7.2.2 and 7.3). This variance analysis allows the interpretation of how robust each instance is in relation to other implementations. For instance, higher variance across different configurations of any particular component suggests that it is less robust compared to the remaining components of the system. Such information allows to make decisions on the trade-off between robust and solid retrieval performance, or a strong but error-prone configuration by adjusting system components accordingly. Studying topic-level retrieval performance, however, permits the investigation of the failures of particular components. Having at hand retrieval effectiveness values for comparable implementations of components provides the opportunity to estimate whether such failures are systematic rather than random (see Sections 7.4.1 and 7.4.2). The result analysis for the experiments in Chapter 7 is founded on these assumptions and considerations. These thorough analyses are intended to demonstrate that the traditional evaluation and result analysis paradigm only scratches the surface of understanding the full potential in state-of-the-art retrieval technology.

Making the results of empirical evaluation experiments available to the research community is the final aspect within the contributions of this work. A critical problem of traditional IR evaluation lies in the fact that experimental result data and experiment descriptions are being archived separately due to the duality between IR evaluation campaigns and the publication of research articles at major IR conferences. In line with the present work, a tool was developed [112], [196] to capture these resources (system configurations and empirical results) in order to visualise the effect of state-of-the-art components on retrieval effectiveness. It will be argued that such a tool could be established as the contact point for the selection of appropriate baselines for empirical evaluation of new IR system components (see Section 7.6). In order to achieve acceptance among the research community, such a tool has to be integrated into the infrastructure of evaluation campaigns like TREC, CLEF, or NTCIR. Future work in this direction could aim to consolidate the descriptions of IR experiments within a formal meta-data standard. Using such a standard in IR evaluation campaigns will considerably increase the utility of previous experiments. Finally, this would make it possible to keep track of the improvements in the field of IR effectiveness research.

7 Automated Component-Level Evaluation with Xtrieval

This chapter describes how Xtrieval has been deployed to run a series of ad-hoc experiments on small and mid-sized document collections in order to study the effect of system configurations on retrieval effectiveness. As a first step, the general set-up of the designed experiments is presented. A number of specific goals and deduced research questions serve as motivation for the selected empirical approach. They define the scope of this work and specify the variables of the experiment architecture. The scientific methodology for the experiment design, as well as the data analysis and interpretation, is presented with respect to the formulated research questions. The introductory section is completed with a detailed description of the test collections that were selected for our empirical analysis.

The key contribution of the present chapter lies in a detailed result analysis with respect to our selection of state-of-the-art IR system components. The results are examined and interpreted in accordance with the presented guidelines for transparent evaluation on component-level (see Chapter 6), in order to provide answers to the formulated research questions. Besides detecting the optimal system configurations for various test collections, the effect of test collections on the ranking of the tested system configurations is discussed. The variance of variables that affect retrieval performance in the experimental set-up will also be analysed.

Since one of the major points of criticism was the lack of an infrastructure for capturing both experimental set-up and results (see Chapter 6), a description of a straightforward tool that was developed to address the issues in Section 7.6 is provided. Firstly, a discussion of the basic functions of the visualisation tool is provided to point out how it could be integrated into existing infrastructures of evaluation campaigns like TREC, CLEF, or NTCIR. Secondly, all experimental results will be published by making them accessible in this web-based tool in order to encourage further analysis. Optimal configurations per system component instance are selected for publication on the platform presented in [8] in order to provide baseline references on component-level. Finally, the chapter is concluded by summarising the major contributions of the presented large-scale experiment.

7.1 Experimental Setup

It has been pointed out in the preceding chapters that the Xtrieval framework can be deployed to access open IR toolkits and libraries. It was argued that Lucene and Terrier are the most powerful tools for text transformation, indexing, and retrieval. In fact, due to different fields of application, wide applicability (Lucene) and coverage of important IR models (Terrier), the combination of the two appeared to be the most promising.

The potential benefit for IR evaluation lies in balancing the shortcomings of each framework with the strengths of the other. The best example is the combination of the powerful text transformation framework included in Lucene with the manifold IR ranking and weighting models implemented in Terrier (see Section 5.2). This approah allows the elimination of the drawback of Lucene, which relies on a combination of the classic Boolean matching and Vector Space retrieval models, but at the same time its powerful text transformation architecture is used to balance the less flexible term pipeline approach in Terrier.

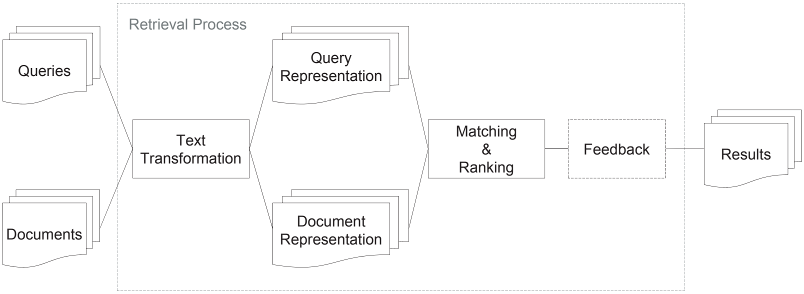


Figure 7.1: Abstract representation of the deployed experimental set-up.

It has been demonstrated that this meta-framework approach achieved strong retrieval performance with top-positions at several international comparisons (see Section 5.4). But more importantly, the Xtrieval architecture allows the study of three key components of standard IR systems: text transformation, matching and ranking functions, and relevance feedback algorithms. As a result, it is possible to enhance the experiments at the RIA workshop (see Section 6.1) and at the Grid@CLEF track (see Section 6.2). Although every possible state-of-the-art component of IR systems could be implemented and tested in Xtrieval, the focus remains on the three fundamental components for the sake of manageability. The number of instances of each component are limited for the same reason.

Figure 7.1 illustrates the overall workflow of the experiments, whereas all tested instances of system components are listed in Table 7.1. Note, that a fixed text processing workflow was implemented in order to create identical document representations for the use with Lucene and Terrier. This fixed tokenising procedure was adapted using Lucene. It preceded the variable stemming approaches presented in Table 7.1.

Figure 7.2 illustrates the data flow for breaking the document text into tokens. Most of the presented steps are standard components from Lucene, like *StandardTokenizer*,

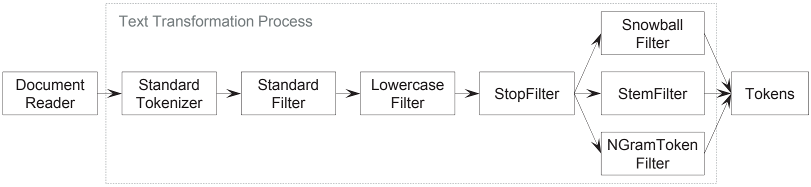


Figure 7.2: Implemented text transformation chain for the creation of document and query representations.

StandardFilter, *LowercaseFilter*, and *StopFilter* (see Section 4.1.1 for details). The class *StopFilter* requires a language-specific set of stop-words. A publicly available list[1] from the University of Neuchâtel was used for this purpose. The final step of the tokenising process is to transform all tokens into stems. Here the *SnowballFilter* class from Lucene is used to access the Porter stemming algorithm. The *NGramTokenFilter* class from Lucene was modified in order to convert all tokens in character n-grams with a maximum length of four (or five respectively). As a result, the token "all" results in the single character 4-gram *"all"* (which is in fact a 3-gram, but was still retained to avoid loss of information), whereas "characters" would produce the 4-grams *"char"*, *"hara"*, *"arac"*, *"ract"*, *"acte"*, *"cter"*, and *"ters"*. Additionally, a Lucene *TokenFilter* class named *StemFilter* was created which served as the interface for other implementations of stemming algorithms.

For the experiments presented hereafter, the Krovetz stemmer [101], and a light rule-based stemmer [88] named UeaLite for English from the University of East Anglia were implemented. They were chosen, because of their similarity to the Porter stemmer. These stemmers are rule-based, but they aim to avoid the problem of over-stemming by implementing less aggressive rules (UeaLite) or using a dictionary for check-up (Krovetz).

---

[1] `http://members.unine.ch/jacques.savoy/clef/englishST.txt`, retrieved on March 1, 2012

Component	Instance	From	Parameter
Stemming	none	-	-
	Porter	Lucene	-
	Krovetz	Lucene	word dictionary
	UeaLite	Lucene	-
	char n-grams	Lucene	n={4;5}
Ranking	Boolean + VSM	Lucene	-
	TF-IDF, BM25	Terrier	BM25:b=0.75
	LM (Dirichlet, Hiemstra)	Terrier	Dirichlet:μ=2500, Hiemstra:λ=0.15
	DFR	Terrier	{BB2, DLH13, DPH, IFB2, In_expB2}
	DFI0, LGD	Terrier	-
Feedback	none	-	-

	Kullback-Leiber	Terrier	d={3;6;9;12;15;20;30}
			t={5;10;15;20;25;30;40;50;60;
			70;80;90;100}

	Bose-Einstein2	Terrier	d={3;6;9;12;15;20;30}
			t={5;10;15;20;25;30;40;50;60;
			70;80;90;100}

Table 7.1: Tested instances of variable system components and their configuration.

Next, the details of the considered ranking algorithms will be discussed. Twelve different weighting and ranking algorithms were selected which can be grouped in four major classes: *Vector Space and classic Probabilistic models*, *language models*, *divergence from randomness models*, and *divergence from independence models*.

From the ranking models listed in Table 7.1, the Boolean matching (see Section 3.2.1) and Vector Space ranking (see Section 3.2.2), provided by Lucene, the TF-IDF ranking (see Section 3.2.3), as well as the BM25 ranking formula (see Section 3.2.4.3), belong to the first category of models. In the category of *language models* two implementations were included: Dirichlet and Hiemstra (see Section 3.2.4.4). The implementations BB2, DHL13, DPH, IFB2, and InExpB2 represent variations of the *divergence from randomness model* (see Section 3.2.4.5). The two recently published models DFI0 and LGD belong to the remaining group. Note that default parameters are used for all configurable ranking models in these experiments. All default values are listed in the rightmost column of Table 7.1 for reference. The reason for choosing this pragmatic approach is twofold:

(1) The test set-up is in line with traditional ad-hoc evaluation experiments and all of the configurable models default to this scenario.

(2) If the parameters of configurable ranking models are included and they do not necessarily have only one value, an additional dimension of the parameter space with an unknown and varying number of values would be introduced.

Two different feedback models have been included in the empirical evaluation, namely Kullback-Leiber and Bose-Einstein2, which are implemented in Terrier. The focus is on two parameters of the pseudo-relevance feedback (PRF) models:

(1) D, the number of documents that are used to collect feedback terms from.

(2) T, the maximum number of terms to be extracted from the set of selected feedback documents.

Automatic PRF in Terrier defaults to using the three topmost documents of an initial ranking and retrieves the ten most relevant terms according to the underlying metric, i.e. the selected feedback model. In general this is a safe setting in the sense that it can be expected to improve average retrieval effectiveness on a typical set of topics. It is commonplace in the IR retrieval community that automatic PRF with fixed document sets and maximum term numbers is a risky operation, because it is not known whether applying the technique will improve or decrease retrieval effectiveness on a particular topic. Moreover, the degree of the effect is also unknown beforehand.

Since both parameters are numerical values, the goal was to define a set of reasonable values to investigate in our experiments. In order to keep the resulting parameter space manageable it was decided to keep the overall number of feedback experiments per model below 100. Since most evaluation campaigns use precision-based effectiveness metrics like $P@n$, with typical values ranging from 1 to 20, the target range of documents for automatic feedback needed to be within these values. The maximum number of terms to expand the original query may be affected by the choice of stemmers. The limitation of the maximum length of a token by using character n-gram stemming during indexing also constrained the amount of information to be fed back into the query reformulation step of PRF. In order to investigate this effect, it was necessary to allow more terms than usual for standard stemmers. As a result, the interval $[0, 30]$ for parameter D and the interval $[0, 100]$ for parameter T have been obtained. In order to comply with the restriction to than 100 parameter combinations, the parameter sets presented in Table 7.1 have been selected.

In order to conclude this section on the general experimental set-up, the total number of generated experiments is determined. 6 different stemming algorithms generate the dimension of the first component. Bearing in mind, that Lucene and Terrier are accessed through Xtrieval results in twelve indices, which have to be created to conduct all experiments. The second main component is the ranking model which constitutes the second dimension of the evaluation. As can be seen from Table 7.1, 12 different ranking models were selected for comparison. The last system component is feedback and it constitutes the third and largest dimension of our evaluation. Using 7 samples for the parameter D and 13 samples for the parameter T results in 91 different pairs of feedback parameters. Since two feedback models are compared, it is necessary to multiply this figure by two. Finally, one should not forget each corresponding baseline experiment without feedback. As a result, there are 183 unique instances for the feedback dimension in the test set-up. Finally, the number of instances in each of the three dimensions can be multiplied which results in 13,176 unique system configurations in total.

7.1.1 Goals and Research Questions

In order to demonstrate the possibility to address the issues formulated in the previous chapter by adopting the Xtrieval framework (see Section 6.2.5), a few specific research questions were formulated. This is achieved by re-considering the general goals in the domain of IR evaluation and in component-level evaluation specifically. One of the most critical problems in IR evaluation is that test collections are re-used for verification of improved IR models, but at the same time there is a lack of commonly used and accepted baseline experiments. Since nobody is in the position to dictate, which baseline experiments should be used for reference when a particular test collection is being re-used, the situation can be improved by making baseline experiments easily accessible. However, some questions regarding these baselines remain. Who decides which experiment should serve as a baseline for reference, and what are the criteria for selecting it? The question about an eligible authority to implement the decisions may remain open. But one might argue that application-specific metrics exist to properly define the selection criteria. Thus, the first general goal is to provide experimental descriptions and results, that enable the community to make such decisions.

Another aspect of the empirical evaluation is to verify the functionality of the Xtrieval framework with respect to the combination of IR toolkits. The present experimental set-up allows the testing of two important interfaces: Lucene's generic text processing framework to create index structures with Terrier and applying Terrier's feedback mechanism to experiments from Lucene. Although these two examples are quite specific, they are considered to be a meaningful contribution to IR evaluation for the following reason. It was pointed out in Chapter 4 that Lucene and Terrier can be regarded as accepted standard tools in IR. The combination of their strengths within a single framework, like Xtrieval, allows to automatically simulate default configurations of state-of-the-art IR systems.

The experiments presented here were created in retrospect, i.e. they are based on test collections which include corresponding sets of relevance assessments. The field of application is restricted to ad-hoc retrieval. Thus, the most relevant system components were selected accordingly. But in general the approach can be adapted to any laboratory IR evaluation experiment. Assuming the question is to thoroughly investigate a specific field of application for IR technology, the IR community could define system components that are expected to be beneficial for the underlying task(s). These components can be included by integrating them in our general framework. As a final step, several instances for each component can be defined and the framework will provide results for all possible experiment combinations. This procedure allows the IR community to ensure that the current state-of-the-art is covered.

Besides these generic goals, there are a number of specific questions that have to be considered with the help of this empirical experiment. Naturally, they are related to the system components and configurations that were tested. The main focus lies in retrieval effectiveness measured using traditional metrics like average precision (AP) and corresponding summary figures, which is mean average precision (MAP) in this case. In the following list provides a number of key questions:

- *Stemming:*
  Stemming algorithms help to improve retrieval effectiveness. Given the five selected instances of stemming algorithms, the question of which of the implementations achieves the best retrieval performance on average (over a defined set of standard topics) can be addressed. Another subject of this investigation is the retrieval effectiveness at topic level. This is motivated by the assumption that some stemming algorithms may perform badly on average, but on a subset of the topics they might outperform the overall "winner". If this is the case and this subset also has some unique features, this information might be useful to improve average performance by choosing the optimal stemming technique on topic basis. Another specific investigation deals with the robustness of the component instances against the misconfiguration of other system components. Due to the experimental set-up, where each stemming algorithm is tested on the same set of system configurations, it is possible to compare identical subsets of configuration by altering only one component instance.
- *Ranking:*
  Ranking functions build the core of every modern IR system. Here, twelve different implementations that were separated into four major groups (see Sec-

tion 7.1) are studied in detail. Similar to the stemming component, the main purpose is to investigate which of the IR models performs best on standard test collections. Whether the models in the defined groups which are presented in a chronological order, improved over time is another question, i.e. it is expected that the vector space and classic probabilistic models be outperformed by the models in the other groups. There is also an interest in topic-level performance and its variance across the tested ranking models. The reasoning is the same as for the stemming algorithms: assuming that topic subsets exist on which some particular model(s) outperform the overall winner, it might be possible to exploit this to improve average retrieval effectiveness beyond the level of an experiment based on a single optimal model. A further aspect of this study concerning the ranking models is to determine the relative difference between groups of models. The review of the research on the theory of ranking models (see Section 3.2.4) demonstrated that some ranking functions, like TF-IDF, BM25, and Hiemstra's language model, are closely related from a mathematical perspective. It is expected that these theoretical assumptions can be verified by means of empirical observation in the experiment.

- *Feedback:*
  Pseudo-relevance feedback is the last component under investigation. Feedback instances provide many more different configurations than the other two components. But in accordance with the other components, the purpose is to find which of the tested feedback models, and which configuration of the parameters D and T result in best retrieval effectiveness over different standard test collections. The topic-level will also be studied by following the previous approach. But in contrast to the stemming and ranking models, it is not expected that it is possible to find particular configuration that perform consistently better on subsets of topics than the overall best feedback configuration. The effect of the feedback configuration on the performance of the other components of the system is a further topic of interest. Based on the assumption that top-ranked documents retrieved using different systems, or system configurations, will be

> similar in order to fulfil the relevance criterion of the query, it is likely that the optimal retrieval effectiveness is similar for these systems. How the feedback mechanism affects the performance of the instances of the other components is another important question. For example, if BM25 outperforms TF-IDF without PRF, will it still outperform TF-IDF with the optimal feedback configuration for both ranking mechanisms? And moreover, how will it affect the difference between the two in terms of some effectiveness metric? Another aspect that needs investigation is the relation between the two PRF parameters D and T. One would expect that parameter T should always be larger than D, i.e. more terms are extracted from more documents in order to achieve better effectiveness. Experiments, where $T <= D$ will also be conducted in order to validate this hypothesis.

Alongside the investigation of effects of particular components, the experimental setup allows the study of how the system components and configurations interact with each other and how these interactions affect retrieval effectiveness. Whether a stemming algorithm affects the order of the ranking models or not is one of the more specific questions. For example, one can assume that the Porter stemmer is the best choice for our first component. The ranking models under investigation will be in a specific order when they are used in combination with Porter stemming. The following question is then, if the order of the models will be the same for the other tested stemming algorithms, or not? This could help to further clarify the investigation of robustness of particular component instances. Of course, it will also be possible to answer the same question from the opposite perspective. The ranking model dimension will be studied to observe, how the relative performance of the stemming configurations changes, when different instances of ranking models are selected.

Different document collections and test sets that were generated for the evaluation of ad-hoc retrieval tasks (see Section 7.1.3 for details) are another topic of interest of the experiments. This allows the study of whether it is possible to observe general patterns across document collections or topic sets. For this reason specific document collections have been selected. The experiments will be carried out on several topic sets, which were aggregated to larger sets in order to smooth the effect of particular topic sets. It also allows us to investigate the questions presented above with respect to different original topic sets. This will help to understand, whether, or not, it is possible to predict the behaviour of system configurations on standard test collections. Since standard test collections for ad-hoc retrieval are being re-used here, it is possible to relate the results to the original results at the corresponding evaluation campaign. This allows to verify that the system configurations are working properly within Xtrieval. This test set-up also allows the determination of appropriate baselines for particular instances of system components, which can be used for testing and verifying future components on the deployed test collections.

7.1.2 Scope and Methodology

It has already been pointed out that empirical experimentation is the selected method to answer the formulated research questions. This section is intended to provide insights into the scope of the present experiments, and to discuss the limits of the research methodology. In general, the analysis of empirical evaluation is a standard tool to assess theories and hypotheses. But the design of the underlying experiments has to be adequate to the problems under investigation.

A typical problem in IR evaluation is that small topic sets limit the usefulness of empirical observation. Since traditional ad-hoc test collections are re-used in the experiments that are conducted for this work, the conclusions that can be drawn are limited to the ad-hoc retrieval domain. The purpose of combining several topic sets into a single experiment per document collection (see Section 7.1.3) is to derive more general insights from our empirical analysis than the single topic sets would allow alone. In order to do so, the following assumption needs to be made. Evaluation campaigns like TREC and CLEF use document pooling to create the relevance assessments for the topic sets. This pooling strategy is dependent on the submitted experiments per task. Since these submissions may differ from year to year due to different participants and systems, a hidden effect of the participating systems may affect the comparability of the topic sets for a corresponding document collection. In traditional IR evaluation it is commonplace to assume that this effect can be ignored. In order to be able to generalise the observations on combined topic sets for the selected document collections this assumption is also applied here.

Another possible limitation of the methodology in this work lies in the fact, that the number of components under investigation and their corresponding instances are limited. Although major parts of this work are dedicated to motivating the selection of algorithms to be included in the empirical analysis, the resulting choices can neither be complete nor cover all possible combinations or configurations of instances. Throughout the selection process a few choices had to be made to keep the experiments and the data analysis manageable. Nevertheless, these decisions are based on common sense in IR evaluation. The number of documents D and terms T to be used for PRF on standard ad-hoc test collections varies with typical values for D from 3 to 15 and from 10 to 50 for T. Both of these intervals are covered in the experimental set-up.

Another aspect that needs pointing out, concerns the potential impact on IR effectiveness evaluation. The lack of tools to select appropriate baselines for the evaluation of new IR methods is beyond dispute. Future evaluation tasks should be able to provide results for predefined subsets of standard system configurations beforehand. As a result, it is expected that these baselines could increase the challenge for participants, simply because they do in fact cover the current state-of-the-art in IR technology. At the same time, providing component-level configurations for standard test collections will allow the comparison of the new contributions with the corresponding reference implementations. This approach enables researchers to identify standard system configurations and components that might not be working as expected. It will also lead to a more thorough analysis of system components and their orchestration for specific use cases.

7.1.3 Selected Test Collections

This section is intended to describe the last and most important part of our experimental set-up: the test collection. It has already been indicated, that standard ad-hoc text retrieval collections are being used for evaluation here. Although the IR community has been tackling problems on a much larger scale in recent years (see Section 2.2), older and smaller document collections are being used here. The reason for this decision is twofold. Firstly, one of the goals is to provide further insights by assessing the component and configuration level. In order to maximise the usefulness of this investigation, it is necessary to run the experiments on the most frequently used test collections in IR evaluation. A recent study [9] examined peer-reviewed publications on IR evaluation over a period of ten years. One of the results was that the most widely used ad-hoc test collections are those from TREC-7 and TREC-8. Thus, this empirical analysis is also based on the corresponding document collection. The second reason concerns the feasibility of the experiment. The Xtrieval framework and the available computational resources are able to handle document collections at web-scale. But the nature of the designed experiment, which involves testing of tens of thousands of system configurations, does not allow another large-scale dimension, if the results are expected to be achieved within a reasonable time frame.

In addition to the document collections from TREC, some additional document collections from CLEF are also included in the evaluation. To stay in line with the ad-hoc retrieval scenario, the selected collections from CLEF were tested for ad-hoc evaluation tasks. A further requirement for inclusion in the present experiment was a minimum number of 100 topics per document collection. As a result, the GIRT-4 corpus (used for the Domain-Specific task for several years), and the TEL-BL collection (used for the ad-hoc retrieval tasks in CLEF 2008 and 2009), were selected. Choosing further

Doc. Collection	# Docs	Avg. Doc Length	Size (MB)	Topic Sets	# Topics
TREC disk 4					
Financial Times	210,158	412.7	564		
Federal Register	55,630	644.7	395		
TREC disk 5					
Foreign Broadcast IS	130,471	543.6	470		
LA Times	131,896	526.5	475		
TREC45-CR (total)	528,155	497.9	1,904	trec7-ah	50
				trec8-ah	50
				trec2003-robust-new	50
				trec2004-robust-new	50
CLEF TEL-BL	1,000,100	30.6	1,152	clef2008-ah	50
				clef2009-ah	50
CLEF GIRT-4	151,319	50.3	198	clef2003-ds	25
				clef2004-ds	25
				clef2005-ds	25
				clef2006-ds	25

Table 7.2: General document collection characteristics and topic sets.

document collections allowed the investigation of the effect of collection features on the defined test set-up. Evaluating identical topic set sizes on these different types of document collections is expected to reveal general trends regarding the effects of particular system configurations on retrieval effectiveness.

7.1.3.1 Document Collection Statistics

A general overview on statistical features of the document collections is presented in Table 7.2. All figures for the TREC document collection, which consists of TREC disks 4 and 5 without the Congressional Records collection, were extracted from [181, p. 3]. For the CLEF TEL-BL and GIRT-4 collections no information could be found on the average document length. As a result, the presented figures are estimates that were obtained from the indexes created for the empirical analysis in this work. The figures below demonstrate the differences between the collections. While the TREC collection features documents with an average length of about 500 words, the CLEF collections contain only about 30 (TEL-BL), and 50 (GIRT-4) words per document. This is due to the structure and content of the underlying document sources. The TREC

Doc. Collection	Indexed Fields
TREC45-CR	
Financial Times	CO, CN, DATELINE, HEADLINE, IN, PE, TEXT, TP
Federal Register	ADDRESS, AGENCY, DOCTITLE, FOOTNOTE, FURTHER, SUMMARY, SUPPLEM, TEXT, USBUREAU, USDEPT
Foreign Broadcast IS	ABS, F, FIG, H1, H2, H3, H4, H5, H6, H7, H8, TEXT, TI, TXT5
LA Times	BYLINE, CHA1, HEADLINE, KW, LD, LEAD, LEAD1, NO, PEOPLE, SUBJECTS, ST, TB, TI, TIO1, TITLE, TEXT, TX
CLEF TEL-BL	dc:abstract, dc:contributor, dc:description, dc:relation, dc:subject, dc:title, dcterms:alternative
CLEF GIRT-4	ABSTRACT-EN-HT, ABSTRACT-EN-MT, AUTHOR, CLASSIFICATION-TEXT-EN, CONTROLLED-TERM-EN, METHOD-TERM-EN, PUBLICATION-YEAR, TITLE-EN

Table 7.3: Indexed fields per document collection.

collection, which is denoted *TREC45-CR* from here on, consists of full-text natural language articles, whereas the CLEF collections represent extracts of library records. The total number of available test queries is 200 for the TREC45-CR collection, and 100 for each CLEF collection.

7.1.3.2 Specific Index Statistics

Next, the document statistics are discussed by presenting several index-specific figures. Table 7.3 lists the fields, that were selected for inclusion in the indexes. This reduction approach was implemented to limit the size of the indexes and to exclude tokens carrying no, or only little information. The indexing process and resulting figures are described in detail in order to ensure the utility of the experiments for future reference. This may enable other researchers to reproduce these experiments and observations. Table 7.4 lists a few index statistics on the number of tokens for each collection and stemming algorithm. A few interesting figures might serve as indicators for the nature of each collection and stemmer. First of all, the number of unique tokens in the rightmost column gives an idea about the vocabulary of the collection. It can bee seen, that TREC45-CR and CLEF-TEL-BL have similar figures, whereas the CLEF GIRT-4 has a vocabulary of about 15-20 percent the size of the other two. This is due to the fact that CLEF GIRT-4 is a sparsely translated version of the corresponding

Doc. Collection	Stemmer	Avg. Doc Length	# Tokens	# Unique Tokens
TREC45-CR	none	293.5	155,001,515	901,896
	4-gram	1,147.4	605,995,312	234,581
	5-gram	907.6	479,329,032	716,582
	Krovetz	293.5	155,001,515	823,035
	Porter	293.5	155,001,515	802,134
	UeaLite	293.5	155,000,500	785,534
CLEF TEL-BL	none	30.6	30,572,382	778,141
	4-gram	130.1	130,132,722	259,585
	5-gram	104.4	104,416,440	765,718
	Krovetz	30.6	30,572,382	732,006
	Porter	30.6	30,572,382	701,572
	UeaLite	30.6	30,570,380	728,957
CLEF GIRT-4	none	50.3	7,605,354	130,812
	4-gram	241.5	36,536,240	66,852
	5-gram	195.5	29,589,492	159,519
	Krovetz	50.3	7,605,354	113,288
	Porter	50.3	7,605,354	106,428
	UeaLite	50.3	7,605,336	116,126

Table 7.4: Index statistics in relation to stemming algorithms.

German collection. The total number of tokens could serve as a rough approximation of query processing time and storage size of the index, because it is directly related to the size of the posting lists. One can observe, that 4-gram stemming results in four to five times more tokens than rule-based stemmers. A similar picture can be obtained by analysing the figures for 5-gram stemming. Here, the factor varies between three and four.

Another interesting observation can be made by comparing the number of unique terms per collection for the rule-based approaches. From a theoretical analysis one would suggest that the UeaLite stemmer is less aggressive than the Porter stemmer. In fact, on the TREC45-CR collection the opposite is the case, whereas for the other collections the figures follow our supposition. A resulting question is, whether, or not, this fact might affect retrieval effectiveness. Since all ranking models rely on the numbers reported in Table 7.4 in one or another way, it is likely that the observed effect will influence the document ranking. What remains an open question at this point of

Collection	avg_l(t)	avg_l(d)	avg_l(n)	avg_l(t+d)	avg(num_rel)
TREC45-CR					
TREC7-AH	2.4	14.6	44.2	17.1	93.5
TREC8-AH	2.4	14.0	36.8	16.5	94.6
TREC2003-Robust	3.0	16.0	33.8	19.0	33.2
TREC2004-Robust	2.8	12.3	27.8	15.0	41.2
CLEF TEL-BL					
CLEF2008-AH	3.3	14.3	0	17.6	50.7
CLEF2009-AH	3.8	16.3	0	19.1	50.5
CLEF GIRT-4					
CLEF2003-DS	3.6	12.8	35.3	16.4	53.3
CLEF2004-DS	3.0	13.0	37.3	16.0	51.5
CLEF2005-DS	2.8	11.8	36.2	14.6	84.2
CLEF2006-DS	3.8	11.8	34.0	15.6	169.6

Table 7.5: Statistics for deployed topic sets.

the investigation is whether the difference in the document rankings will be reflected in the used retrieval effectiveness metric.

7.1.3.3 Topic Set Statistics

The test topics are one of the key elements in every empirical evaluation. They are necessary to be able to measure the quality of tested IR systems based on the returned results. Due to the importance of the test queries, their main features are examined here. Table 7.5 lists mean lengths of the topic fields (represented using the function $avg_l()$), and the average number of relevant documents.

Typical topic elements are abbreviated with the corresponding first letter of the topic structure name, i.e. t for title, d for description, and n for narrative. All presented figures are reported for the original topic sets from respective evaluation campaigns. Special attention is directed to the two rightmost columns of Table 7.5. The average length of the topic title and topic description, ($avg_l(t + d)$), is important, because it represents the terms that are used to create the queries in the experiments. The narrative topic part was not included for evaluation, because it was not available for the

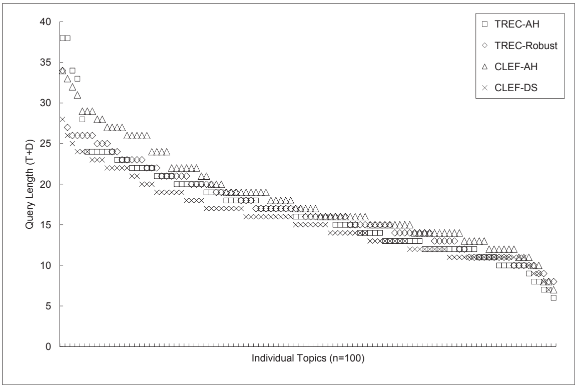


Figure 7.3: Query lengths for individual topics (n=100, order by decreasing query length) for each of the 4 aggregated test sets: TREC-AH, TREC-Robust, CLEF-AH, and CLEF-DS.

CLEF TEL-BL collection. It can be seen that the average query length varies between 14.6 and 19.1 on the selected sets. Considering the average number of relevant documents per topic, a large variance ranging from 33.2 to 169.6 can be observed. Finally, there is only little variation in the average lengths of the individual topic structures (title, description, narrative).

One of the formulated goals for the designed evaluation experiment is to be able to draw more general conclusions from the empirical analysis. In order to achieve this goal, the topic sets for the three document collections were aggregated to test sets of identical size. The assumption behind this approach is straightforward: the larger the topic sets are, the more reliable they should become, because more test samples cover the complete population of possible queries better. However, due to evolving topic creation and assessment strategies, the features of individual topic sets may not be remotely comparable, even when they are created for the same document collection. For the experiments in this work, four topic sets of identical size (100 test queries) were generated. These include two sets for TREC45-CR, with TREC7-AH and TREC8-AH being grouped together. For CLEF TEL-BL and CLEF GIRT-4, all individual topic sets were merged to form two additional topic sets containing 100 topics each.

In order to investigate the hypothesis that the comparability of the topic sets is increased by merging the test queries into sets of identical size, two major features are studied in detail. In Figure 7.3 and Figure 7.4, the length of the queries and the number of relevant documents per query are plotted for the merged topic sets. The query lengths for the merged topic sets are illustrated in decreasing order for each of the four test collections. Similar to the average query lengths presented in Table 7.5, there is only little variance across the different test collections. Within each of the topic sets, most queries consist of 10 and 25 terms with only a few outliers on both ends of the range. As a result, it can be concluded that the topic sizes and their distribution are homogeneous for the merged test query collections.

The number of relevant documents per topic in Figure 7.4 are also presented in decreasing order for each of the deployed topic sets. In contrast to the figures in Table 7.5, the topic sets feature quite similar distributions of relevant documents for each individual topic. However, the level of these distributions, which is represented by the area under the curves, is quite different across our four merged test collections. In fact, the distributions for TREC-AH and CLEF-DS are very similar with just a few deviations. The CLEF-AH test collection has a similar distribution, but it has only about half as much relevant documents for all topics.

A similar picture can be observed for the TREC-Robust test collection, which also shares the general distribution across all topics with only about 40 percent of relevant documents per topic in comparison to TREC-AH and CLEF-DS. The conclusion of this analysis is mixed. On the one hand the general distribution of the number of relevant documents per topics is similar for all test sets. But on the other hand, there are quite different absolute numbers across the collections. The latter argument may appear disturbing for the pupose of reliable component evaluation. However, it is com-

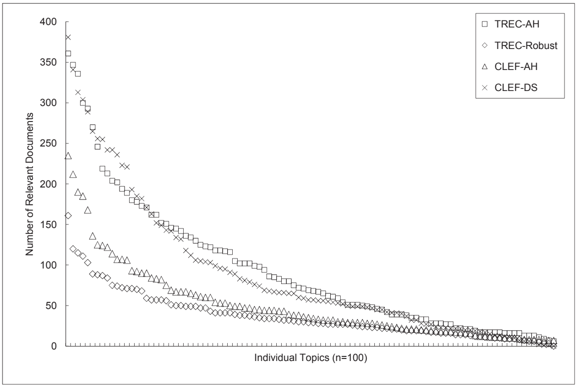


Figure 7.4: Number of relevant documents for individual topics (n=100, order by decreasing number of relevant documents) for each of the 4 aggregated test sets: TREC-AH, TREC-Robust, CLEF-AH, and CLEF-DS.

monplace in IR evaluation, that there is no correlation between the number of relevant documents per topic and the retrieval effectiveness.

In order to gain deeper insights on how combining the individual topic sets into single sets changed or retained the features of the original topic sets, the corresponding figures for the distribution of the query lengths and the number of relevant documents were also captured. From the observations, that can be made in the corresponding figures in Appendix A, one can conclude, that the original topic sets for the TREC7-AH and TREC8-AH, TREC2003-Robust and TREC2004-Robust, as well as CLEF2008-AH and CLEF2009-AH, are homogeneous in terms of both features. In contrast to that, the individual topic sets for the CLEF GIRT-4 document collection vary significantly in terms of the number of relevant documents. The topic set combination normalised the distribution of relevant documents in that particular case.

7.2 Result Analysis and Interpretation

This section reports the results of the designed large-scale empirical experiment. In order to discuss these results, an appropriate technique to illustrate the large amount of results ($n = 13,176$ observations) needs to be selected. In fact, the total number of tested experiment configurations is $n = 14,274$, due to two different implementations for the Hiemstra language modelling approach (see Section 3.2.4.4). A detailed description for this problem is given in Section 7.4.1. The corrected implementation of Hiemstra's language model ranking was used for the following investigation.

Due to the number of experiments conducting traditional comparisons of the retrieval effectiveness of systems is impossible. Instead, an exploratory data analysis is provided to investigate the effect of different component implementations on retrieval effectiveness. First of all, a description of the applied visualisation techniques is given. Then, each of the system components is studied separately by examining the distributions of the tested software implementations. The corresponding results are reported for all of the four test collections described in Section 7.1.3. This allows to gain general insights into how the system configurations take effect on different types of test collections.

Another aspect of the result analysis is to study optimal system configurations. But instead of only reporting the single best system configuration for each of the test collections, optimal configurations are presented and discussed component-by-component. Thus, the results for a selected subset of system configurations are presented and discussed. The obtained optimal system configurations are compared to the best contributions of the original evaluation campaigns in order to allow an objective evaluation of our results.

The scale of the present experimental set-up allows the study of the effect of the test collections and more specifically, the impact of individual topics on the evaluation task as a whole. Since the actual configuration of all systems is known in the present experiment, analysing the variance of topics on selected subsets of experiments will provide a better understanding of the interaction between individual queries and the IR systems (and its component configurations). The variance of the systems on individual topics can be used to identify topics that are more useful for evaluation, because they discriminate the rankings of system configurations better than others. From a more general perspective, this approach could also be used to assess the utility of a test collection for evaluation. Another use for the analysis of individual topics is finding the optimal system configuration for each of the topics. A theoretical upper limit for the mean average precision over a considered test collection is determined by using a subset of optimal configurations.

A further aspect of the result analysis concerns the utility of Xtrieval for automated evaluation. An additional experiment is designed to investigate this by means of introducing another component of Xtrieval. This component is based on the late data fusion approach and is intended to increase the retrieval effectiveness by combining results from individual system configuration. This experiment is chosen for the purpose of establishing the validity of our approach to automated component-level evaluation. Thus, a selected subset of optimal, but also distinguishable system configurations, is used for a pair-wise data fusion approach. The results of these experiments show, that it is possible to improve retrieval effectiveness measured in terms MAP beyond the level of the best automatic IR systems from original ad-hoc evaluation tasks at TREC.

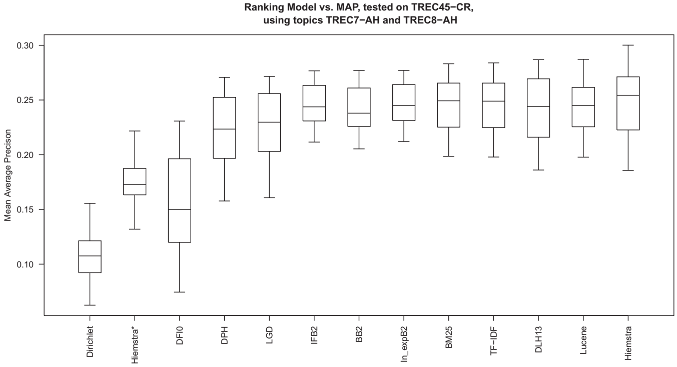
7.2.1 Exploratory Analysis: General Observations

In traditional IR evaluation, a standard approach is to assess the statistical significance of the difference between two systems (see Section 2.4). Obviously, for the experiments presented and discussed in this work, this is not only infeasible for all (or even selected) pairs of systems configurations. But moreover, it would not provide any insights into the effects of components and how their different implementation affect retrieval performance. Thus, other techniques from the field of statistics are applied in order to obtain a graphical visualisation of our experimental results. These visualisations serve as the basis for the interpretation of the results in terms of specific research questions.

It is a common approach to graphical analysis and interpretation of large empirical data sets to study characteristic statistical features. Mean, median, maximum, and minimum of the observations are only a few examples of such figures. The five-number summary [86] is a specific tool for visual interpretation of univariate variables. It relies on the following figures: minimum, lower quartile, median, upper quartile, and maximum. In the present experimental set-up these univariate variables are the components themselves, which appear as different instances (here: software implementations). In fact, from an abstract perspective the pseudo-relevance feedback component is multivariate, because it depends on the used model, the number of documents, and the number of terms. Since all three contribute to the desired effect, and can be quantified independently, they are considered as being three separate components in order to examine their impact on the target metric. As a result, the outcome of the experiments can be examined in terms of these typical statistical figures to summarise the distribution of the observations.

The most common visualisation of a five-number summary is a boxplot [142]. Nowadays, there are many variants of the traditional boxplot visualisation. Most of them differ in the representation of the lower and upper end. A standard boxplot (see Figure 7.5, top) usually exchanges the minimum for the 9th percentile and the maximum the 91st percentile.

In spite of the concise visualisation of characteristic features of the underlying distributions, boxplots do not provide any graphical interpretation of the probability density. This might be problematic in situations, where the density function of the data provides further insights into the effects of the studied variables. The research literature in



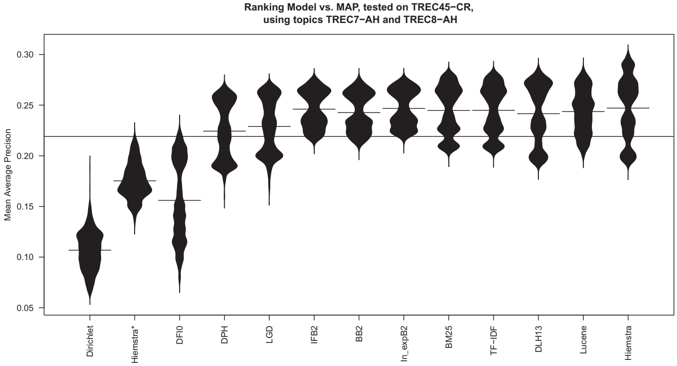


Figure 7.5: Comparison of a boxplot (top) and a beanplot (bottom) visualisation for the results of the test configuration based the TREC45-CR document collection and the combined topic sets TREC7-AH and TREC8-AH. In this example, the univariate factor under examination is the *ranking model*, which has 13 different instances.

the field of statistics provides two solutions to the problem of the comparison of probability density functions of factors in large data sets: violin plots [85] and beanplots [92]. Both of them replace the non-informative lines on the side of the boxes with kernel density functions in order to provide a visual interpretation of the data. The violin plot is based on a fixed kernel function for density estimation and an adjustable, but constant, bandwidth parameter which enables a researcher to tune the presentation according to the data set. But the latter is also a limitation of the technique, because the adjustment of the bandwidth parameter might fail on some particular data sets. In contrast to that, the beanplot presentation features a number of kernel density estimation functions and it also supplies different bandwidth estimation procedures. In this particular scenario, where the probability density functions for different types of factors (the software implementations of different IR system components) are unknown, the latter feature is most important. For this reason, it was decided to base the exploratory data analysis on beanplots.

Figure 7.5 illustrates the distribution of various ranking model implementations evaluated on one of the test collections, in terms of boxplot and beanplot visualisations. This allows a comparison of the advantages and drawbacks of both types of graphical presentation. The boxplot visualisation provides an overview of the relation of the five-number summaries for all tested instances of the ranking model component. Outliers are shown as circles if present (see the distributions for the Dirichlet and Hiemstra* models in the left). However, two specific goals of the present experiments are the analysis of related groups of models, and to estimate the reliability of component instances in relation to the complete system configuration. In order to answer these questions, the beanplot visualisation allows the interpretation of the results based on the graphical presentation of the probability density functions.

Next, the visualisations of the In_expB2 and Lucene models in the boxplot and beanplot presentations are examined in detail (see Figure 7.5, number eight and twelve, from left to right). In the boxplot presentation, both distributions look fairly similar, except for the general expansion which is slightly higher for Lucene. In contrast to that, the beanplot presentation reveals an interesting difference in the density function for these two models. The probability density function for Lucene is almost uniform which suggests that other parameters of the system configuration do not affect the model much. A different picture can be observed for the In_expB2 density function. Here, one can see a variant of a bimodal distribution. This observation suggests that some other parameter is likely to affect the retrieval performance. This effect will be elaborated in the following subsection. Similar observations can be made when comparing respective presentations for Dirichlet and Lucene.

7.2.2 Visual Comparison of Component Instances

This section provides a general overview on the effect of IR system components on retrieval performance measured in terms of MAP. In order to study these effects thoroughly, each of the major components is considered separately. The following figures are used to interpret these effects: minimum, mean, and maximum, as well as the average, across all instances. Moreover, the probability density functions are used to relate the different groups of models for particular components, namely stemming and ranking. The illustrations in the following subsections are organised in specific sequences of the factors of the system components:

- *Stemming algorithms*: none, 4-gram, 5-gram, Porter, Krovetz, Uealite
- *Ranking models (vector space and probabilistic group)*: TF-IDF, BM25, Lucene
- *Ranking models (language model group)*: Dirichlet, Hiemstra*, Hiemstra
- *Ranking models (DFR group)*: BB2, IFB2, In_expB2, DPH, DLH13
- *Ranking models (DFI group)*: LGD, DFI0
- *Feedback models*: none, Bo2, KLCorrect

This kind of presentation allows straightforward comparison of the effectiveness across the considered test collections. The two numerical factors of the PRF component, namely the number of documents and terms for PRF, increase from left to right in order to illustrate the effect of the respective parameter values.

In order to conduct a meaningful exploratory analysis of the experimental results, it is necessary to make sure that the underlying data is free from systematic errors. Therefore, Figure 7.5 should be considered again by focusing on this particular aspect. The illustration clearly suggests that three ranking models, namely Dirichlet, Hiemstra*, and DFI0, do not perform well in comparison to the rest of the ranking models. This observation can be made for each of the four test collections (see Appendix B.1.2).

The observation that these models systematically fail on all test collections suggests that the configuration is not correct, or the software implementation has errors (see Section 7.4.1 for more details). Thus, it was decided to exclude these ranking models from the exploratory analysis and to narrow the focus on the working configurations. Nevertheless, all corresponding figures were also created for the sake of completeness (see Appendix B.1).

7.2.2.1 Stemming

Figure 7.6 illustrates the impact of the five tested stemming algorithms on retrieval effectiveness and relates this impact to the same set of configurations tested without any stemming. Each of the sub-figures summarises one of the four test collections. The total number of observations is $n = 10,980$. Considering the effectiveness of the n-gram stemming, it can be observed that these approaches perform consistently worse than all of the other approaches, even in comparison to the configurations without stemming.

On the full-text TREC45-CR test collections the performance of 4-gram and 5-gram stemming is fairly similar. In contrast to that, on the library record collections CLEF TEL-BL and CLEF GIRT-4, the 4-gram approach outperforms the 5-gram technique on average. Following the outlined course of action, these observations suggest, that the system configurations using n-gram stemming should be examined separately.

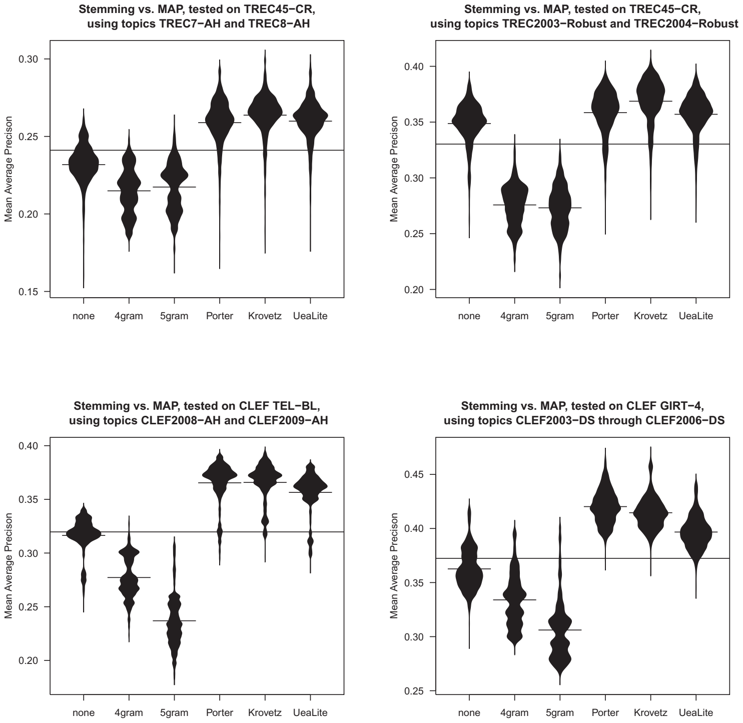


Figure 7.6: Beanplot visualisation using different *stemming implementations* as factors ($n = 10,980$ observations per sub-figure). Each figure illustrates the data for one of the merged topic sets created for evaluation.

Therefore, the exploratory analysis of the remaining system components is based on the subset of configurations that use the Porter, Krovetz, and UeaLite stemmers. The corresponding figures for the analysis of configurations based on n-gram stemming can be found in Appendix B.2.

Another point of interest of the visual analysis and interpretation concerns the probability density function of the n-gram stemmers in comparison to the other algorithms.

The curve progression for the n-gram stemmers looks like a mixture of multi-modal and uniform distributions, whereas the other algorithms produce shapes that are similar to normal distributions. This suggests that the n-gram stemming influences the effectiveness of other components of the entire system.

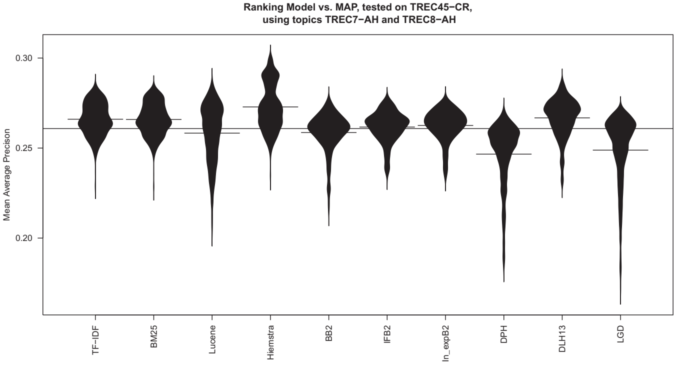
Next the emphasis is put on the comparison of the rule-based stemming algorithms Porter, Krovetz, and UeaLite. The Porter and UeaLite implementations show similar average performance on the TREC45-CR document collections, but on the CLEF collections Porter achieves better effectiveness than UeaLite. Krovetz is the best of the three on both TREC45-CR test collections. Its performance is similar to Porter on the CLEF TEL-BL test collection and slightly worse than Porter on the CLEF GIRT-4 test set. These observations suggest that the Krovetz implementation is preferable on full-text collections and that Porter is a good choice on sparse collections like library records.

Another interesting aspect is the small difference between configurations using the rule-based stemmers, and the configurations without stemming, on the TREC45-CR test collection with robust topic sets. Here, there is an especially small gap between the no stemming and rule-based stemming. However, no robust conclusion can be drawn based on this observation without the examination of the other system components. A final comment concerns the total expansion of the MAP values on the different test collections. On the TREC45-CR test collections the range from minimum to maximum performance is about 0.15 MAP regardless of the absolute MAP values. In contrast to that, this variance is only 0.10 measured in MAP. This implies that the risk of system mal-configurations for full-text collections is higher than for collections of library records.

7.2.2.2 Ranking Models

The following exploratory analysis of the effect of ranking models on retrieval effectiveness is based on a subset of $n = 5,490$ observations. In order to allow meaningful interpretations, the graphical illustrations presented hereafter focus on system configurations that use Porter, Krovetz, or UeaLite stemming. Furthermore, ranking models that are suspected to be configured erroneously, were also excluded from the presentation. All corresponding figures for the analogue investigation on a subset based on the character n-gram stemming configurations ($n = 3,660$) can be found in Appendix B.2.1.

Figure 7.7 shows a comparison of the ten remaining ranking models on the test collections for the TREC45-CR corpus. The corresponding presentation of the results on the test collections from CLEF is illustrated in Figure 7.8. Again, the exploratory analysis is focused on general characteristics of the different implementations of the ranking component. The sequence of the ranking models used for presentation is in chronological order of the publication of the models. Given this perspective, one would expect the beanplots to be in increasing order from left to right. In fact this is not the case for all collections used here. This observation suggests that newer ranking models do not perform better than classic models on the tested collections. It can be observed in Figure 7.7 and Figure 7.8 that the empirical observations demonstrate the close relationships within the defined groups of ranking models. Next, the presentation of the results for the TREC45-CR ad-hoc test collections is studied in more detail. It is very clear that the ranking models TF-IDF and BM25 from the traditional group are very similar in terms of minimum, mean, and maximum performance as well as in the shape of their probability density function. A similar picture can be seen for BB2, IFB2, and In_expB2 from the DFR group of ranking models. However, in this group DPH and DLH13 differ significantly in terms of the minimum, mean, and maximum performance, but they still share a very similar density function on the experiment subset under examination.



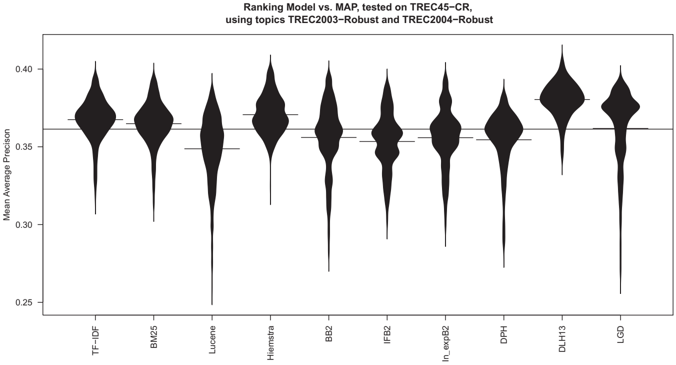
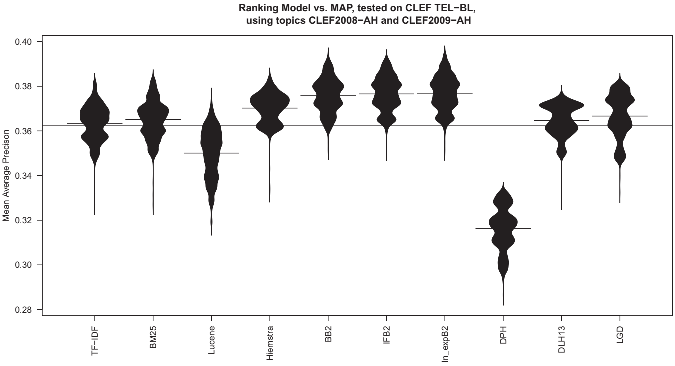


Figure 7.7: Beanplot visualisation using different *implementations of ranking models* as factors ($n = 5,490$ observations per sub-figure). Each figure illustrates the data for one of the merged topic sets created for evaluation.

Another aspect that is worth noting, is the shape of the probability density functions with respect to the different ranking models. An optimal density function would be funnel-shaped, i.e. most of the system configurations would perform very well, and the more the performance decreases the less configurations should appear. However, this optimal shape criterion for the density function should only be applied to resolve ties of models showing similar maximum and mean retrieval performance.



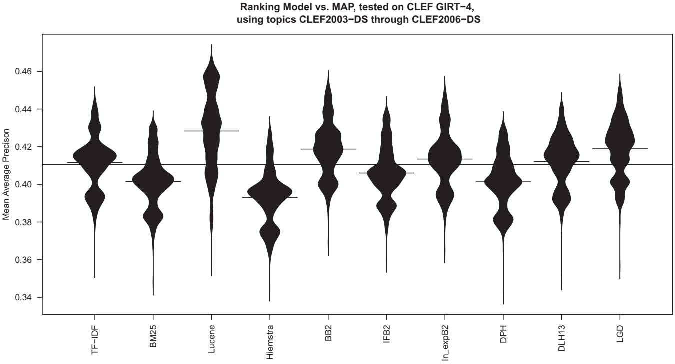


Figure 7.8: Beanplot visualisation using different *implementations of ranking models* as factors ($n = 5,490$ observations per sub-figure). Each figure illustrates the data for one of the merged topic sets created for evaluation.

The density functions for Lucene and Hiemstra differ significantly from the distributions of the other models on the TREC45-CR ad-hoc test collection (see Figure 7.7, top). Instead of a normal distribution, the density function for Lucene looks more like the shape of a uniform distribution. Considering the subset of the system configurations, which represent the distributions, this suggests that Lucene is more affected by the configuration of other components than the models with normal distributions. This conclusion is also supported by the fact, that for Lucene the difference between minimum and maximum performance is larger than for most of the other ranking models.

Yet another distribution can be observed for the Hiemstra ranking model, which looks like a bimodal distribution. This indicates, that the performance of the model might be dependent on two factors in one of the other system components, i.e. the stemming algorithm or the feedback mechanism.

A comparison of the performance of the ranking models across the four different test collections follows next. Figure 7.7 demonstrates how using different test topics on the same document collection changes the relationship between the models. However, the defined groups of ranking models still show similar performance on each test collection. There are two significant differences to be observed. Firstly, the DLH13 ranking model clearly outperforms all other models on the TREC45-CR robust collection (see Figure 7.7, bottom), whereas the Hiemstra ranking model showed best performance on the TREC45-CR ad-hoc test set. Secondly, the difference between the minimum and maximum among all ranking models is significantly larger on the robust topic set than on the ad-hoc test collection.

Lastly, the analysis of the effect of the ranking models is considered on the collections from CLEF (see Figure 7.8). They differ from the TREC45-CR corpus in the average length of the documents. The differences in the nature of these document collections are clearly visible. Obviously, they affect the probability density functions for the ranking models. The shapes of the probability distributions are multi-modal for most ranking models on the CLEF TEL-BL test collection with the exception of Lucene and Hiemstra. For the CLEF GIRT-4 test collection most ranking models show a bimodal distribution on the selected subset of experiments. These observations suggest that there is another system component with two factors on the CLEF GIRT-4 collection. For the CLEF TEL-BL collection the interpretation of the probability density function is even more diffuse. It indicates that other components severely affect retrieval performance on this test collection. Bearing in mind the distributions of the stemming components for CLEF GIRT-4 (see Figure 7.6, bottom right), the bimodal distributions can be explained with the influence of the stemming algorithms. Porter and Krovetz show similar, high performance and account for the upper peak in the ranking model distributions. The lower peak can be attributed to the UeaLite stemming component, which achieves worse retrieval effectiveness than Porter and Krovetz on this collection. A conclusive explanation of the observations for the ranking model probability distributions on the CLEF TEL-BL cannot be found without the consideration of the effect feedback component.

7.2.2.3 Pseudo-Relevance Feedback

The third and last key component examined by means of exploratory data analysis is pseudo-relevance feedback. It was noted before, that the configuration of PRF is a multivariate variable due to the different parameters of the component. However, these parameters can be separated into three variables, which can be quantified individually. In order to study these, the results are reported and analysed in terms of these variables. The selected experiment subset consists of $n = 5,490$ configurations covering the Porter, Krovetz, and UeaLite stemming algorithms, as well as the ranking algorithms presented in the previous subsection (see Section 7.2.2.2). All corresponding beanplot visualisations for the experiment subset ($n = 3,660$ observations) covering the n-gram stemming approaches, are listed in Appendix B.2.

The first variable considered for comparison is the PRF model used to select the terms for query reformulation. Due to the large number of different configurations for PRF, the total number of experiments using a specific PRF model ($n = 2,730$) differs significantly from the number of experiments without PRF ($n = 30$). As a result the probability density functions for the latter factor ("none"), cannot be directly compared to the distributions of the actual PRF models.

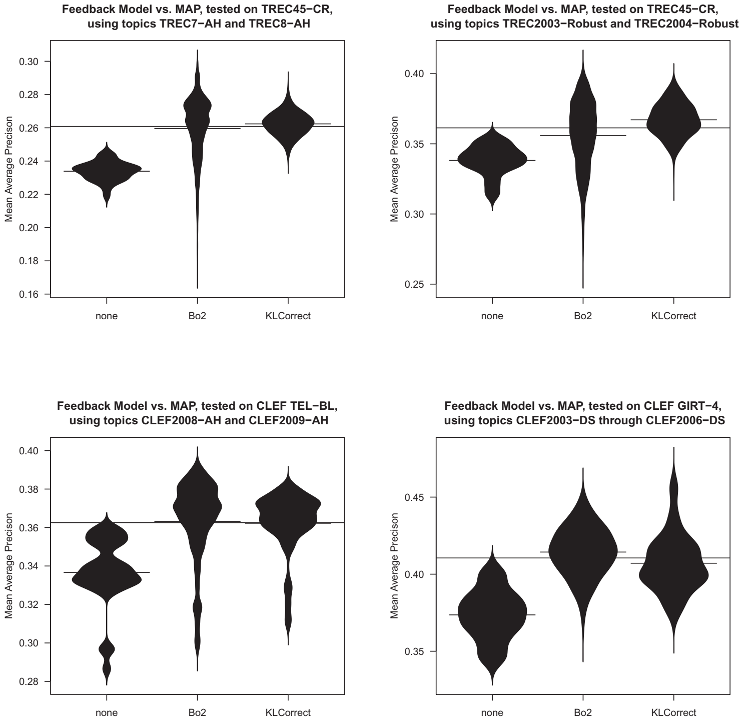


Figure 7.9: Beanplot visualisation using different *PRF models* as factors ($n = 5,490$ observations per subfigure). Each figure illustrates the data for one of the merged topic sets created for evaluation.

Figure 7.9 illustrates the results on the selected experiment subset by means of beanplots. The presentation clearly shows that both PRF models improve retrieval effectiveness on average, and on all test collections. The presentation as beanplots reveals another interesting aspect of the PRF component. As the focus is sharpened on the TREC45-CR document collection (see Figure 7.9, top), it can be observed that the two PRF models feature very different probability density functions. While the Bo2 implementation produced a distribution that might be characterised as a uniform distribution, the KLCorrect implementation resulted in an almost perfect normal distribution for the ad-hoc test collection, and in a slightly biased normal distribution for the robust test set. Another aspect concerning the different probability density functions is the variance across the experiment subset under examination. The Bo2 implementation achieved a higher maximum performance on both test sets of the TREC45-CR document collection than its counterpart implementation. But at the same time the minimum performance was much lower than the minimum performance of KLCorrect. As a result, the KLCorrect implementation works better than the Bo2 algorithm on average for the TREC45-CR document collection. This observation suggests that KLCorrect as PRF model is more reliable than Bo2, but the latter has the potential to perform significantly better than its counterpart if it is used with an optimal system configuration.

The last question of interest is how the document collections affect the PRF models? In order to address this it is necessary to compare the observations for the CLEF document collections (see Figure 7.9, bottom) with the previous findings. On the CLEF TEL-BL collections an interesting pattern can be seen for the probability density functions. All instances share a similar tail, which suggest that some mis-configuration may have appeared on this collection, or that a particular component fails systematically. Recalling the visual comparison of the ranking models on this experiment subset and test collection (see Figure 7.8, top) reveals that the component in question is likely to be the DPH ranking model.

The specific shape of the density functions on the CLEF TEL-BL collection indicates some other systematic effect. This will be detailed when the other parameters of the PRF component are graphically examined and interpreted. The illustration of the results for the CLEF GIRT-4 test collection reveals that the observations on the TREC45-CR collections are reversed here. This indicates that the Bo2 algorithm is

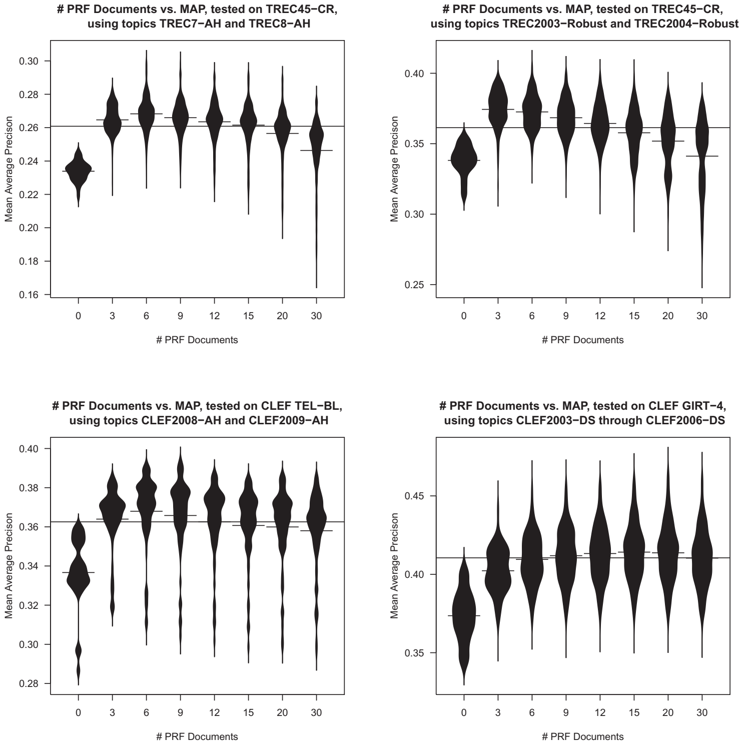


Figure 7.10: Beanplot visualisation using different values for the *number of PRF documents* as factors ($n = 5,490$ observations per sub-figure). Each figure illustrates the data for one of the merged topic sets created for evaluation.

more reliable than the KLCorrect implementation for sparse collections like library records.

Figure 7.10 demonstrates the impact of the number of documents used for PRF on retrieval effectiveness. The underlying data set is identical to those from Figures 7.7, 7.8, and 7.9. The visual presentation demonstrates that the standard configurations of modern IR systems are fairly reasonable. Both the average and maximum retrieval

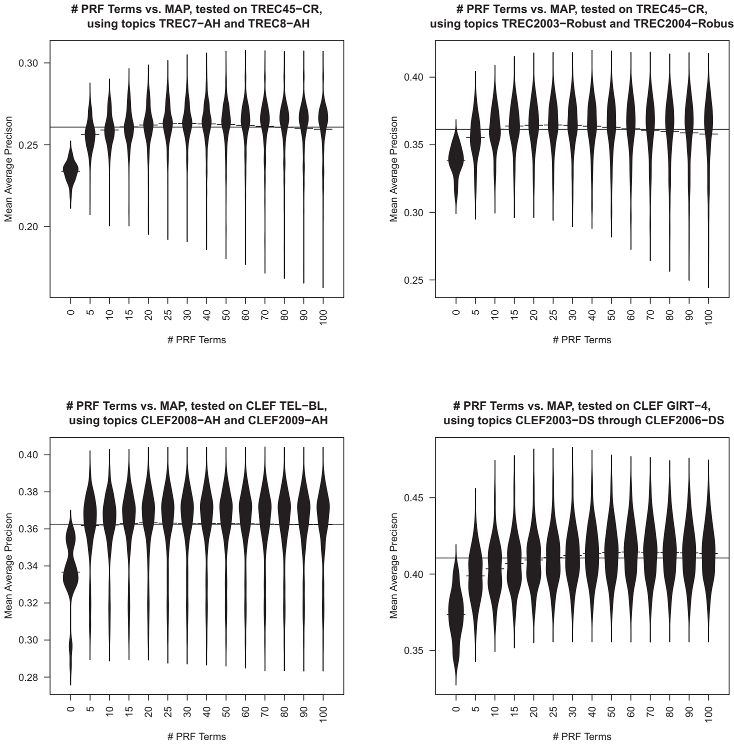


Figure 7.11: Beanplot visualisation using different values for the *number of PRF terms* as factors, ($n = 5,490$ observations per sub-figure). Each figure illustrates the data for one of the merged topic sets created for evaluation.

effectiveness increase with the number of documents used for PRF up to a certain threshold, after which using more documents decreases the performance more and more. For the test collections used in this work this threshold varies from 3 to 9 documents, with the exception of the CLEF GIRT-4 collection. Here the threshold is about 20 documents, which differs significantly from the other test collections. As a final comment on Figure 7.10, we focus once more on the probability density functions for the instances of the variable. It can be observed that for all collections except CLEF

TEL-BL, these functions look like stretched normal distributions. Moreover, the lower end of the distribution gets longer for higher numbers of PRF documents, i.e. the minimum is lower for these values. This observation confirms, that the more documents used the higher the risk that the PRF operation will fail. The examination of the shape of the probability density function for the CLEF TEL-BL collection indicates that other components affect the impact of the PRF component. Note, that the variation at the low end stems from one particular ranking model, (DPH).

The impact of the last variable of the PRF component, the number of terms used for PRF, is studied next. Figure 7.11 illustrates the distributions for the selected experiment subset and presents the results in terms of the number of terms for PRF. The visualisations demonstrate that the effect of the number of terms used for PRF is very similar to the effect for the number of documents for PRF. This indicates that the two variables could be correlated, but testing this hypothesis is beyond the scope of this work. The illustrations show that, on average, the optimum number of terms to use for PRF varies between 20 and 40 for all tested collections, except for CLEF TEL-BL. For values larger than 50 the average performance of the PRF component decreases, as it also does when larger numbers of documents are used for PRF.

Why the number of terms used for PRF shows an unexpected effect for the CLEF TEL-BL collection is the next subject of this analysis. One can see that using a few terms increases retrieval effectiveness in comparison to configurations that did not use PRF. However, for larger values of the variable there is almost no change in retrieval performance any more. Bearing in mind the expected effect of the number of terms for PRF, this is an indication of a systematic failure. In order to verify this hypothesis, the results were analysed by re-running a subset of the experiments. It turned out, that the observed failure is indeed systematic. But in contrast to the expectation that it might be due to an erroneous system and/or component configuration, the actual problem is collection-specific: most documents in the CLEF TEL-BL collection are very short (about five to ten terms after stemming). As a result, the actual threshold for the number of terms to be selected for PRF is never met, due to the constraint on the number of documents for PRF. Finally, this results in the effect observed for the CLEF TEL-BL test collection (see Figure 7.11, bottom left).

Finally, it should be noted once again that the results in this section were constrained on rule-based stemming algorithms in order to allow a more meaningful exploratory data analysis. However, one of the formulated research questions was concerned with the effect of the different stemming algorithms. The actual question was whether n-gram stemming algorithms need more terms for the PRF mechanism to work optimally, than standard stemmers like Porter or Krovetz. In order to answer this question, the results in Figure 7.11 need to be compared to the corresponding graphical presentations on the n-gram stemming experiment subset. These results are illustrated in Appendix B.2.5. The beanplot illustrations demonstrate the expected effect. The average performance of the experiment subset increases with increasing numbers of terms used for PRF on all test collections except CLEF TEL-BL, for which the collection effect discussed in the previous paragraph is also visible. The graphical presentations in Appendix B.2.5 also show that the gain in performance when using more terms for PRF increases slower than on the experiment subset focusing on rule-based stemmers. Moreover, the tested values for the number of terms used for PRF do not allow the conclusion that our selected maximum value of 100 terms results in the maximum gain in retrieval effectiveness when using PRF. In fact, the results indicate that the average optimal configuration of the variable is larger than 100 for the TREC45-CR ad-hoc and the CLEF GIRT-4 test collections. For the TREC45-CR robust test collection the graphical illustration shows that on average there is no gain in retrieval effectiveness for more than 50 terms used for PRF.

7.2.2.4 Conclusion

In this section the different aspects of the IR system components and test collections that were investigated by means of the exploratory analysis of large sets of experiment configurations are briefly summarised. Here, the emphasis is put on the analysis of the key facets, the most important observations and their interpretation. Corresponding implications are provided for cases where the results allow to draw robust conclusions.

- *Stemming Algorithms*:
  The visual presentation of the results demonstrated that the n-gram stemming techniques were clearly outperformed by the rule-based approaches on all selected test collections. Even the configurations without stemming consistently outperformed both tested n-gram implementations. This is a straightforward observation. Another important observation is that Krovetz stemming outperformed Porter on full-text test collections like TREC45-CR, and that it achieved similar retrieval performance (in comparison to Porter stemming) on sparse document collections like CLEF TEL-BL or CLEF GIRT-4. The difference between Krovetz and Porter on TREC45-CR can be attributed to the dictionary used in Krovetz' approach. The light stemming implementation UeaLite resulted in retrieval effectiveness similar to Porter on the TREC45-CR collection, but was outperformed by the other two rule-based techniques on the library record collections CLEF TEL-BL and CLEF GIRT-4. Due to the large difference between the two key techniques for stemming (n-gram and rule-based) the experiment sets were split in order to enable a clear interpretation of the other components.
- *Ranking Models*:
  In this category the graphical illustrations indicated that some of the tested ranking models, namely Dirichlet, Hiemstra*, and DFI0, failed systematically on all test collections. Furthermore, the analysis of our results showed that the DPH ranking model failed on the CLEF TEL-BL collection. The reasons for these failures were not analysed in detail except for the Hiemstra* ranking model (see Section 7.4.1). In general the visualisations and interpretation presented in Section 7.2.2.2 demonstrated substantial variance in retrieval effectiveness for the different ranking models. In contrast to the expectation that recent models would consistently achieve better performance than traditional ranking models, no upward trend was found on the used test collections. Furthermore, the

analysis showed that the implementations of the models in defined categories result in fairly similar probability density function. This is an empirical verification of the theoretic relationships between the models within each specific category. Another important implication is the fact that some ranking models are more sensitive to other component configurations. As a result, they incorporate a higher risk to result in poor retrieval effectiveness.

- *Pseudo-relevance Feedback Configurations*:
  The considered PRF component configuration had three major variables: the number of documents, the number of terms, and the actual PRF model used for the reformulation of an initial query. Starting with the latter variable, the analysis indicated that the average and maximum optimal performance is collection-dependent. For the full-text collection TREC45-CR the KLCorrect implementation worked better than the Bo2 algorithm on average. However, the PRF models are ranked vice versa in terms of maximum performance on this collection. The number of documents used for PRF was the next parameter of interest. The analysis of the experiment sets revealed that the optimal number of documents varies between 3 and 9 (on average), depending on the test collection. Another aspect of this analysis is the observation that more documents decrease the retrieval effectiveness. The last variable that was considered for evaluation, was the number of terms used for PRF. The visualisation of the experiment result sets indicated similar results as for the number of documents. The more terms are being used the better the effectiveness of the PRF component will be. A collection-specific threshold using more terms decreases the effectiveness. This threshold varies between 20 and 40 on three of the four tested collections. A further disadvantage of using more documents than this threshold is a higher risk of poor retrieval performance.

- *Test Collections*:
  The exploratory result analysis demonstrated that the retrieval effectiveness of some of the system components are collection-specific. In particular the relative performance of the stemming techniques, the ranking models, and the PRF

> models, varied across the test collections used for evaluation. A very interesting aspect was that some of the ranking models failed on all test collections and the DPH failed on a single test collection. This indicates that the effectiveness of ranking models may be dependent on the test collections. Furthermore, the ranking models in the defined groups (see Section 7.2.2) achieved similar retrieval distributions on the evaluated experiment sets for each of the test collections. This implies that using only one representative from each group may be sufficient for the purpose of automated component-level evaluation of other retrieval tasks. The investigation of the configurations of PRF components confirmed the common knowledge that the optimum configuration for retrieval effectiveness is hard to guess, because it is collection-specific. In general, using more documents or terms results in higher risk of PRF failure.

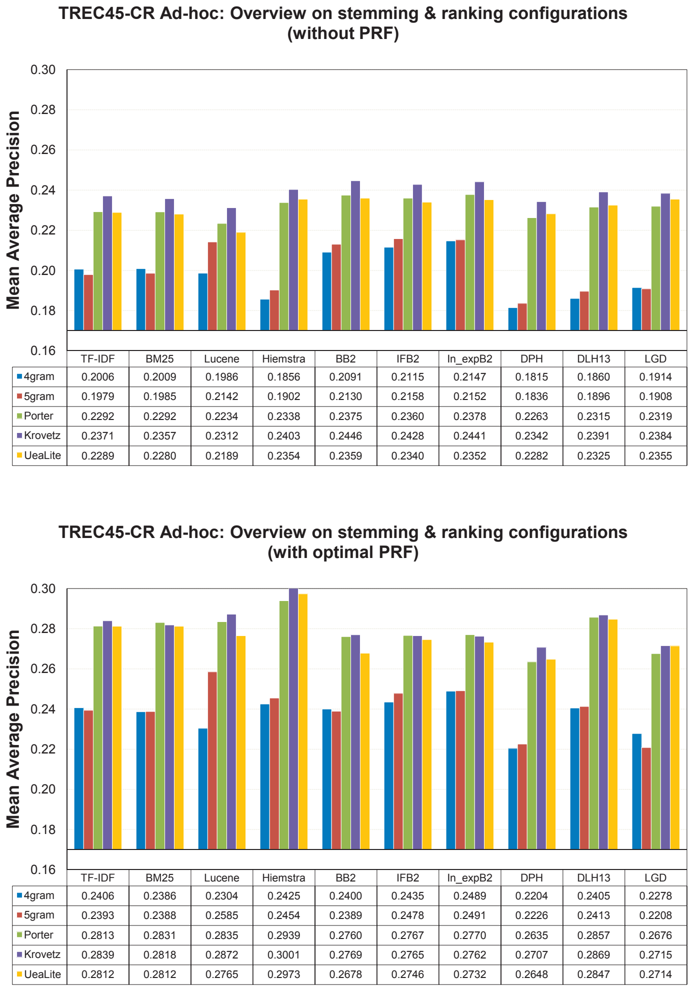
On the one hand, the presented observations and their implications provide answers to the general questions from Section 7.1.1, but on the other hand more specific questions remain open. This is due to the applied methodology. The exploratory data analysis was useful to identify problems in the general system configuration. This kind of result interpretation also allowed to find specific component failures by means of an empirical comparison of individual component implementations on several ad-hoc test collections. As a result, the focus of this investigation was on average retrieval effectiveness. Another important question concerns the maximum retrieval effectiveness and, more specifically, how the tested system components and their configurations can be tuned to achieve optimal performance.

7.3 Optimal System Configurations

Finding the optimal system configurations in the pool of empirical experiments is a straightforward task. Due to the large number of different configurations of the PRF component (183), the entire experiment set is biased. However, finding the optimal configuration for a PRF component is tricky, because of its variance in terms of retrieval effectiveness across topics. In order to provide a plausible answer to the question of the optimal overall system configuration per test collection, the PRF component configurations are divided into two groups. First, the optimal experiment configurations without PRF are selected for analysis. Second, for each of these experiment configurations, the optimal PRF configuration is selected in order to compare it to the corresponding baseline.

The exploratory data analysis in Section 7.2.2 demonstrated that the component configurations need to be restricted due to systematic failures of particular component implementations. As a result, the ranking component contributes only 10 out of 13 implementations to the study of optimal configurations. All stemming algorithms are included in order to investigate their effect in relation to the ranking models. This results in two sets of 50 optimal configurations, which cover all key components of the empirical evaluation set-up. In the following paragraphs, the results on the four test collections are presented and discussed based on visual presentations and the retrieval effectiveness values listed in two-dimensional matrices.

Figure 7.12 illustrates these two sets of optimal experiment configurations for the TREC45-CR ad-hoc test collection, which consists of 100 topics. Each of the illustrations for the test collections (see Figures 7.12, 7.13, 7.14, and 7.15), presents the set of the configurations without PRF in the top and the configurations with optimal PRF in the bottom. Starting with the configurations without PRF, it can be observed that the Krovetz stemming implementation gives the best results for all presented ranking models. The performance of Porter and UeaLite is very similar across the ranking models. Some models prefer Porter over Uealite and others vice versa. The same is the case for the 4-gram and 5-gram implementations. BB2 is the optimal ranking algorithm on the TREC45-CR ad-hoc collection with PRF switched off. However, without the actual MAP figures below the bar charts, it would be hard to determine this model as the winner. In fact, IFB2, In_expB2, and Hiemstra achieved similar effectiveness.



	TF-IDF	BM25	Lucene	Hiemstra	BB2	IFB2	In_expB2	DPH	DLH13	LGD
🟦 4gram	0.2006	0.2009	0.1986	0.1856	0.2091	0.2115	0.2147	0.1815	0.1860	0.1914
🟥 5gram	0.1979	0.1985	0.2142	0.1902	0.2130	0.2158	0.2152	0.1836	0.1896	0.1908
🟩 Porter	0.2292	0.2292	0.2234	0.2338	0.2375	0.2360	0.2378	0.2263	0.2315	0.2319
🟪 Krovetz	0.2371	0.2357	0.2312	0.2403	0.2446	0.2428	0.2441	0.2342	0.2391	0.2384
🟨 UeaLite	0.2289	0.2280	0.2189	0.2354	0.2359	0.2340	0.2352	0.2282	0.2325	0.2355

	TF-IDF	BM25	Lucene	Hiemstra	BB2	IFB2	In_expB2	DPH	DLH13	LGD
🟦 4gram	0.2406	0.2386	0.2304	0.2425	0.2400	0.2435	0.2489	0.2204	0.2405	0.2278
🟥 5gram	0.2393	0.2388	0.2585	0.2454	0.2389	0.2478	0.2491	0.2226	0.2413	0.2208
🟩 Porter	0.2813	0.2831	0.2835	0.2939	0.2760	0.2767	0.2770	0.2635	0.2857	0.2676
🟪 Krovetz	0.2839	0.2818	0.2872	0.3001	0.2769	0.2765	0.2762	0.2707	0.2869	0.2715
🟨 UeaLite	0.2812	0.2812	0.2765	0.2973	0.2678	0.2746	0.2732	0.2648	0.2847	0.2714

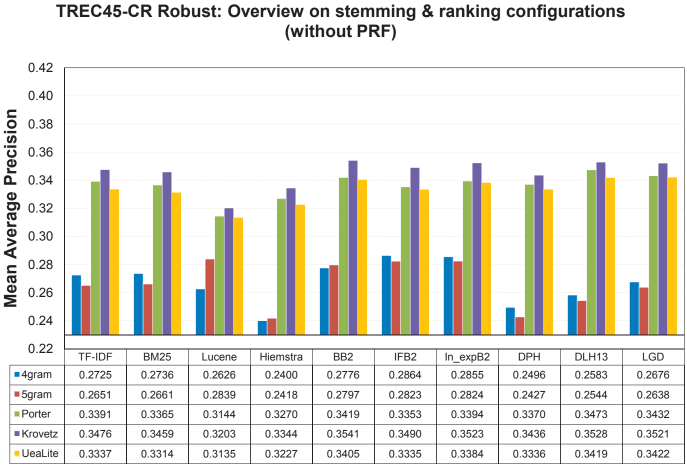
Figure 7.12: Illustration of the best system configurations for each of the components under investigation, tested on the TREC45-CR ad-hoc collection. Baseline configurations without PRF (a) are compared to the respective configurations by selecting the best PRF configuration for each of the 50 combinations for stemming and ranking (b). Note that the actual PRF configuration for each of the bars (or each value in the matrix) might be different for this reason (see Appendix C.1 for details).

Next, the emphasis is put on the corresponding experiments with optimal PRF set-up in order to investigate how the PRF configuration affects the effectiveness of the configurations of the two other components. It can be seen that the absolute gain from PRF varies between 0.04 and 0.06 in terms of MAP. Also, the gain of the Porter and UeaLite stemming implementations is larger in comparison to the algorithm by Krovetz.

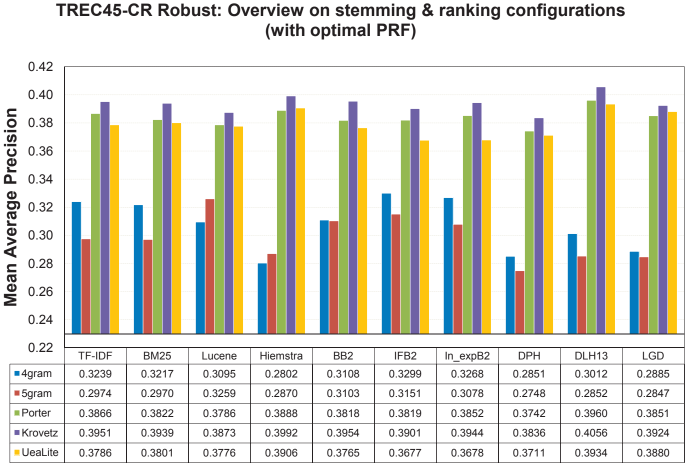
Some ranking models, like Hiemstra, DLH13, and Lucene, improve retrieval effectiveness more than others, like BB2, IFB2, and In_expB2. The variance across ranking models is larger for configurations with PRF, than without. In contrast to that, the variance across different stemming configurations is smaller for the experiment configurations with PRF, than without. Whether this effect is specific for this test collection will be examined by the analyses of the optimal results on the remaining test collections.

The illustration in Figure 7.12 shows that the optimal configuration with PRF consists of the Krovetz stemming algorithm and the Hiemstra ranking model implementation. In this particular case, the PRF configuration is based on the Bo2 PRF model and 60 terms selected from the top-6 documents. The complete configurations of the 50 selected experiments are listed in Appendix C. After considering the optimal PRF configurations on this test collection it was found that Bo2 works better than KLCorrect, and that the optimal number of documents varies between 3 and 12 depending on the actual configuration of the other components. These observations suggest that the optimal PRF configuration for a test collection depends on the actual configuration of other system components.

How a different set of topics tested on the same document collection affects the observations regarding the optimal system component configuration is studied next. Figure 7.13 illustrates these optimal system configurations for the TREC45-CR robust test collection. First the focus is on the different stemming algorithms. The bar chart visualisation demonstrates that the Krovetz stemming implementation performs best across all ranking model implementations.



	TF-IDF	BM25	Lucene	Hiemstra	BB2	IFB2	In_expB2	DPH	DLH13	LGD
🟦4gram	0.2725	0.2736	0.2626	0.2400	0.2776	0.2864	0.2855	0.2496	0.2583	0.2676
🟥5gram	0.2651	0.2661	0.2839	0.2418	0.2797	0.2823	0.2824	0.2427	0.2544	0.2638
🟩Porter	0.3391	0.3365	0.3144	0.3270	0.3419	0.3353	0.3394	0.3370	0.3473	0.3432
🟪Krovetz	0.3476	0.3459	0.3203	0.3344	0.3541	0.3490	0.3523	0.3436	0.3528	0.3521
🟨UeaLite	0.3337	0.3314	0.3135	0.3227	0.3405	0.3335	0.3384	0.3336	0.3419	0.3422



	TF-IDF	BM25	Lucene	Hiemstra	BB2	IFB2	In_expB2	DPH	DLH13	LGD
🟦4gram	0.3239	0.3217	0.3095	0.2802	0.3108	0.3299	0.3268	0.2851	0.3012	0.2885
🟥5gram	0.2974	0.2970	0.3259	0.2870	0.3103	0.3151	0.3078	0.2748	0.2852	0.2847
🟩Porter	0.3866	0.3822	0.3786	0.3888	0.3818	0.3819	0.3852	0.3742	0.3960	0.3851
🟪Krovetz	0.3951	0.3939	0.3873	0.3992	0.3954	0.3901	0.3944	0.3836	0.4056	0.3924
🟨UeaLite	0.3786	0.3801	0.3776	0.3906	0.3765	0.3677	0.3678	0.3711	0.3934	0.3880

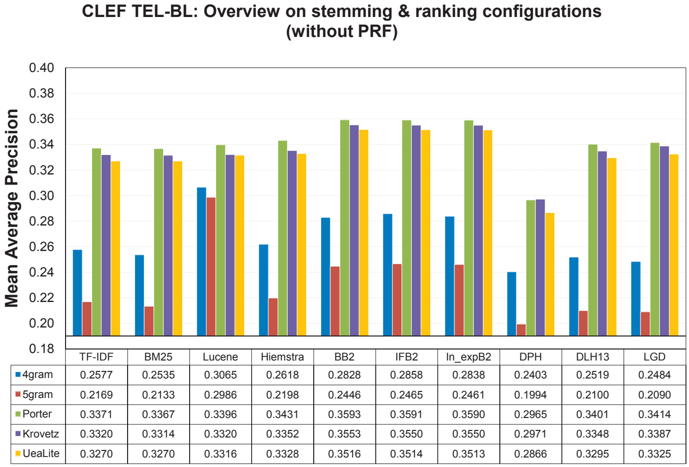
Figure 7.13: Illustration of the best system configurations for each of the components under investigation, tested on the TREC45-CR robust collection. Baseline configurations (a) without PRF are compared to the respective configurations by selecting (b) the best PRF configuration for each of the 50 combinations for stemming and ranking. Note that the actual PRF configuration for each of the bars (or each value in the matrix) might be different for this reason (see Appendix C.2 for details).

The Porter and the UeaLite stemming implementations achieve similar retrieval effectiveness. Furthermore, there is only little variance in the optimal performance of the different ranking models. In contrast to the previous observation (see Figure 7.12) on the experiment set with the optimal PRF configuration, the Krovetz stemming algorithm remains at the top position here (see Figure 7.13, bottom). The 4-gram stemming outperforms the 5-gram stemming in combination with most of the ranking models except for Hiemstra and Lucene.

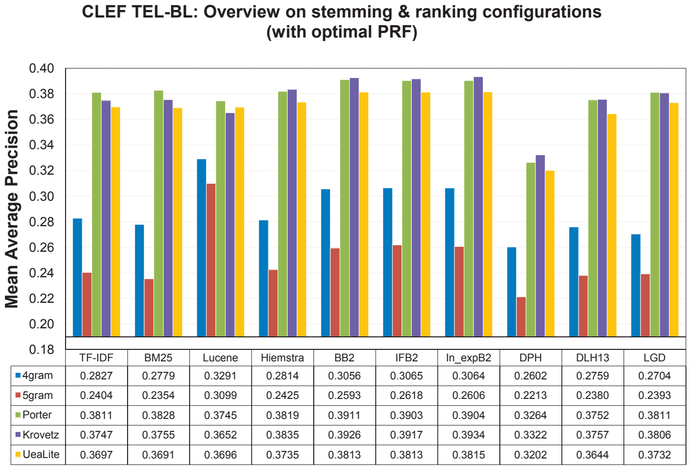
Considering the variance across the ranking models, one can see that it remains at a low level for configurations with PRF. The variance across the different stemming algorithms is similar for configurations with or without PRF.

The best overall retrieval effectiveness for the TREC45-CR robust test collection was achieved by the Krovetz stemming algorithm, the DLH13 ranking model, and the Bo2 PRF model using 40 terms from 6 documents for automatic query expansion. The complete descriptions of the configurations of the optimal experiments for the TREC45-CR robust test collection are presented in Appendix C.2. These results demonstrate that the Lucene ranking model always works best in combination with the KLCorrect PRF model whereas most of the other ranking models favour the Bo2 PRF model. In order to investigate whether these effects are due to differences in the system components or caused by the composition of the test collections, the optimal configurations on the CLEF TEL-BL and CLEF GIRT-4 have to be analysed.

Figure 7.14 presents the results for the described selection of optimal system configurations per component on the CLEF TEL-BL test collection. The illustration of the experiments without PRF (see Figure 7.14, top) reveals that the Porter stemming algorithm works best for all ranking models except DPH.


CLEF TEL-BL: Overview on stemming & ranking configurations (without PRF)

	TF-IDF	BM25	Lucene	Hiemstra	BB2	IFB2	In_expB2	DPH	DLH13	LGD
🟦 4gram	0.2577	0.2535	0.3065	0.2618	0.2828	0.2858	0.2838	0.2403	0.2519	0.2484
🟥 5gram	0.2169	0.2133	0.2986	0.2198	0.2446	0.2465	0.2461	0.1994	0.2100	0.2090
🟩 Porter	0.3371	0.3367	0.3396	0.3431	0.3593	0.3591	0.3590	0.2965	0.3401	0.3414
🟪 Krovetz	0.3320	0.3314	0.3320	0.3352	0.3553	0.3550	0.3550	0.2971	0.3348	0.3387
🟨 UeaLite	0.3270	0.3270	0.3316	0.3328	0.3516	0.3514	0.3513	0.2866	0.3295	0.3325


CLEF TEL-BL: Overview on stemming & ranking configurations (with optimal PRF)

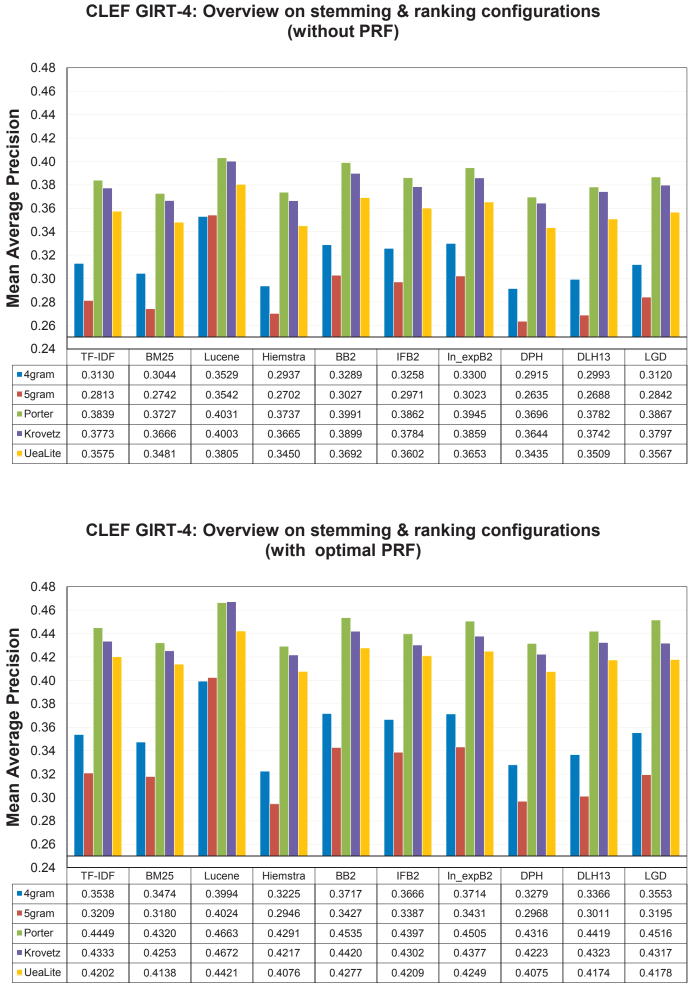
	TF-IDF	BM25	Lucene	Hiemstra	BB2	IFB2	In_expB2	DPH	DLH13	LGD
🟦 4gram	0.2827	0.2779	0.3291	0.2814	0.3056	0.3065	0.3064	0.2602	0.2759	0.2704
🟥 5gram	0.2404	0.2354	0.3099	0.2425	0.2593	0.2618	0.2606	0.2213	0.2380	0.2393
🟩 Porter	0.3811	0.3828	0.3745	0.3819	0.3911	0.3903	0.3904	0.3264	0.3752	0.3811
🟪 Krovetz	0.3747	0.3755	0.3652	0.3835	0.3926	0.3917	0.3934	0.3322	0.3757	0.3806
🟨 UeaLite	0.3697	0.3691	0.3696	0.3735	0.3813	0.3813	0.3815	0.3202	0.3644	0.3732

Figure 7.14: Illustration of the best system configurations for each of the components under investigation, tested on the CLEF TEL-BL collection. Baseline configurations (a) without PRF are compared to (b) the respective configurations by selecting the best PRF configuration for each of the 50 combinations for stemming and ranking. Note that the actual PRF configuration for each of the bars (or each value in the matrix) might be different for this reason (see Appendix C.3 for details).

The Krovetz stemmer is the second best and performs slightly worse than the Porter algorithm. It achieves slightly better MAP values than the UeaLite implementation in combination with all ranking models except DPH. In general, all rule-based stemming algorithms outperform the character n-gram implementations. The analysis of the results for the latter techniques shows that the 4-gram stemming clearly outperforms the 5-gram stemming approach on this test collection. There is only little variance in the group of the rule-based stemming techniques in combination with each of the tested ranking models. This observation is similar to the findings for the two test collections on the TREC45-CR document corpus.

A thorough examination of the performance of the different ranking models shows that the ranking models BB2, IFB2, and In_expB2, achieve the best overall retrieval effectiveness in combination with the rule-based stemming algorithms. In combination with the n-gram stemming techniques, the Lucene ranking implementation outperforms the other ranking algorithms. In general, the variance across the different ranking algorithms on each of the tested stemming implementations is about 0.02 in absolute MAP values. This finding is in line with the observations on the other test collections. An observation worth pointing out is the variance in performance of the Lucene ranking approach across all tested stemming algorithms. The absolute difference between the best and the worst MAP values is only about 0.04. This indicates that Lucene might be more robust than other ranking algorithms on structured test collections like CLEF TEL-BL.

The retrieval effectiveness for the optimal PRF configurations on the CLEF TEl-BL collection is presented in the bottom part of Figure 7.14. The most obvious observation is the relative gain of the configurations including the Krovetz stemmer in relation to the other rule-based stemming implementations. On all ranking models except the ones from the traditional group, the Krovetz implementation performs better than the Porter stemmer.



	TF-IDF	BM25	Lucene	Hiemstra	BB2	IFB2	In_expB2	DPH	DLH13	LGD
🟦 4gram	0.3130	0.3044	0.3529	0.2937	0.3289	0.3258	0.3300	0.2915	0.2993	0.3120
🟥 5gram	0.2813	0.2742	0.3542	0.2702	0.3027	0.2971	0.3023	0.2635	0.2688	0.2842
🟩 Porter	0.3839	0.3727	0.4031	0.3737	0.3991	0.3862	0.3945	0.3696	0.3782	0.3867
🟪 Krovetz	0.3773	0.3666	0.4003	0.3665	0.3899	0.3784	0.3859	0.3644	0.3742	0.3797
🟨 UeaLite	0.3575	0.3481	0.3805	0.3450	0.3692	0.3602	0.3653	0.3435	0.3509	0.3567

	TF-IDF	BM25	Lucene	Hiemstra	BB2	IFB2	In_expB2	DPH	DLH13	LGD
🟦 4gram	0.3538	0.3474	0.3994	0.3225	0.3717	0.3666	0.3714	0.3279	0.3366	0.3553
🟥 5gram	0.3209	0.3180	0.4024	0.2946	0.3427	0.3387	0.3431	0.2968	0.3011	0.3195
🟩 Porter	0.4449	0.4320	0.4663	0.4291	0.4535	0.4397	0.4505	0.4316	0.4419	0.4516
🟪 Krovetz	0.4333	0.4253	0.4672	0.4217	0.4420	0.4302	0.4377	0.4223	0.4323	0.4317
🟨 UeaLite	0.4202	0.4138	0.4421	0.4076	0.4277	0.4209	0.4249	0.4075	0.4174	0.4178

Figure 7.15: Illustration of the best system configurations for each of the components under investigation, tested on the CLEF GIRT-4 collection. Baseline configurations (a) without PRF are compared to (b) the respective configurations by selecting the best PRF configuration for each of the 50 combinations for stemming and ranking. Note that the actual PRF configuration for each of the bars (or each value in the matrix) might be different for this reason (see Appendix C.4 for details).

This is surprising because of the lower performance on the corresponding baseline without PRF. This indicates an interaction effect between the stemming and the PRF components on this test collection. Other features, like the variance across ranking models for each stemming algorithm, or the relationship between rule-based and n-gram stemming approaches, remain the same as for the configurations without PRF. The best configuration for this test collection incorporates the Krovetz stemmer, the In_expB2 ranking algorithm, and the Bo2 PRF model with 30 terms extracted from the top-9 documents. The complete experiment configurations including the optimal PRF parameter set-up for the CLEF TEL-BL test collection is presented in Appendix C.3.

The corresponding results for the last test collection CLEF GIRT-4 are illustrated in Figure 7.15. The Porter stemming algorithm achieved the best results in combination with each of the ranking models for the baseline configurations without PRF (see Figure 7.15, top). The results for the Krovetz stemmer are very close to the performance of the Porter algorithm. From the rule-based stemming approaches UeaLite returned the worst results in terms of MAP. For some ranking models, namely BM25, Hiemstra, DPH, and DLH13, the performance of the UeaLite stemmer is even worse than the best result for the character n-gram stemming algorithms. Considering the empirical results on the other test collections, this is surprising. The variance of the different implementations in the group of the rule-based stemmers in combination with each ranking model is larger than on all other test collection. However, the absolute differences in terms of MAP are not higher than 0.03. From the tested n-gram stemming techniques, the 4-gram implementation consistently outperforms the 5-gram implementation across the different ranking models, except for the ranking implementation of Lucene. Here, the performance of both n-gram stemming component instances is similar.

The performance of the different ranking models on the CLEF GIRT-4 collection is our next topic. The ranking model of Lucene achieves the best retrieval effectiveness in terms of MAP. Some of the models from the DFR group, namely BB2 and In_expB2, produced results of similar quality. The variance across the ranking models in combination with each of the stemmers is small, just as for the other test collections considered for evaluation. Using the Porter stemmer as reference, the absolute MAP values vary between 0.3696 and 0.4031. Both illustrations in Figure 7.15 demonstrate the ability of the Lucene ranking implementation to achieve strong retrieval performance regardless of the used stemmer. The results suggest that this is a unique feature of the ranking implementation in Lucene on structured text collections, like CLEF GIRT-4, or CLEF TEL-BL.

Comparing the results of the baseline experiments on the CLEF GIRT-4 collection to the corresponding experiments with the optimal PRF configuration (see Figure 7.15, bottom) provides only few additional findings. First of all, the best ranking model is Lucene. What can also be seen is that the Krovetz stemmer performs slightly better than the Porter stemmer in combination with Lucene as the ranking model. Here, the reversal between the best stemming implementations when comparing baseline experiments and configurations with optimal PRF, appears only in combination with Lucene. The absolute difference in terms of MAP is very small. In general the gain from using the optimal PRF configuration is about 0.05 in terms of absolute MAP. This difference is larger than on the other test collections. It can be attributed to the higher retrieval effectiveness of the baseline experiments. The optimal configuration from the tested set of experiment configurations on the CLEF GIRT-4 test collection includes the Krovetz stemmer, the Lucene ranking model, and the KLCorrect PRF model, using 30 terms from the top-20 documents for automatic query expansion. The complete list of all 50 experiment configurations with optimal PRF set-up (see Figure 7.15, bottom) is presented in Appendix C.4.

7.4 Further Applications of the Experiment Set

It has already been pointed out in Section 6.3 that the clarity of IR evaluation can be further enhanced. The experimental results generated for the evaluation in this work enable a thorough analysis of system components and their interactions in terms of retrieval effectiveness on standard test collections. In Section 7.2, an exploratory data analysis and a selection strategy covering all component instances were applied to study their effect on several test collections. Many other research questions can be addressed with the created experiment set. For example, one could analyse which of the components and configurations affect retrieval performance most. An analysis of variance would be one approach to address this question. Further studies might investigate whether specific component instances are favoured by certain test collections. Another detailed analysis may clarify whether the topics can be classified in different groups that should be processed with different system configurations. The size of the data set also enables the research community to conduct in-depth statistical analyses across system components and test collections. Filling this list of potential topics for research is beyond the scope of this work.

For that reason the focus is on two specific questions that have already been indicated throughout the result analysis in Section 7.2. First, systematic failures of a particular component implementation are analysed and then it is demonstrated how such problems can be identified and corrected. Based on a comparison across the selected test collections, the functionality of the implemented correction is verified. Second, the optimal experiment sets from Section 7.3 are used to study the variance of the individual topics from the test collections used for evaluation.

7.4.1 Detection and Analysis of Systematic Failures

The exploratory data analysis demonstrated that particular components fail systematically across test collections, or at least on particular test collections. This section is intended to illustrate the course of action that is needed to detect, analyse and correct such problems. For that purpose, the systematic failure of Hiemstra's ranking model on all four selected test collections is presented and discussed in detail. The effect of the implementation of Hiemstra's ranking model (Hiemstra*), in Terrier is documented in Appendix B.1.2. It can be seen that the best system configuration incorporating

Hiemstra* as the ranking model returns worse results in terms of retrieval effectiveness than an average system configuration on most of the other ranking models. This observation is consistent across the used test collections.

As a result the hypothesis that the implementation of the Hiemstra ranking model in Terrier is incorrect has been formulated. Since Hiemstra's ranking model is parametric, the root of the problem could also be due to a misconfiguration of the parameter α. For this reason, the original publication on the model [83] was analysed in order to study the issue in detail. Hiemstra suggested four ranking functions which are based on Equation 3.22 from Section 3.2.4.4. In fact the first of the ranking functions proposed in [83, p. 85] is equivalent to Equation 3.22.

A second ranking function replaced the background model estimation, which was based on c_{q_i}, the number of times the query word q_i that appeared in the collection C, and $|C|$, the total number of words in the collection C, as presented in Definition 7.1. It can be seen that c_{q_i} is substituted by f_{D,q_i}, the number of documents D, that contain the query word q_i (also known as document frequency), and that $|C|$ was replaced with $|f_{D,t}|$, which denotes the sum over all document frequencies for each term t in the collection C. Since $|f_{D,t}|$ is independent of the query, it can be approximated with collection-specific figures like the total number of tokens in the collection.

$$log(P(Q|D)) = \sum_{i=1}^{n} log(\lambda \cdot \frac{f_{q_i,D}}{|D|} + (1-\lambda) \cdot \frac{f_{D,q_i}}{|f_{D,t}|}) \tag{7.1}$$

The two remaining ranking functions proposed in [83] incorporate a document length correction in the weight of the query word q_i. In order to keep matters clear, the corresponding functions for Definition 3.22 and Definition 7.1 are not reproduced here. According to [83, p. 85], the sum over all query words q_i can be implemented by multiplying the weight of q_i by its number of occurrences in the query. The current implementation in Terrier is based on Definition 7.2. It can be seen that it differs from the proposed ranking function in two points. Firstly, the implementation does not contain the query term frequency factor that was suggested in [83, p. 85] to compensate the re-computation of weights for terms that appear more than once. Secondly, the background model is estimated by the fraction of c_{q_i}, the number of times the query word q_i appeared in the collection C, divided by $|f_{D,t}|$, which is approximated by the total number of tokens in the collection.

$$log(P(Q|D)) = \sum_{i=1}^{n} log(\lambda \cdot \frac{f_{q_i,D}}{|D|} + (1-\lambda) \cdot \frac{c_{q_i}}{|f_{D,t}|}) \tag{7.2}$$

Due to the bad results for the original implementation of Hiemstra's ranking model in Terrier, the ranking algorithm was re-implemented based on Definition 7.1. The two implementations of the ranking model were compared using the experimental set-up in order to verify the hypothesis of a systematic failure in the original implementation. Table 7.6 lists the MAP values for the baseline experiment configurations without PRF and using the Porter stemming algorithm. The original implementation of the ranking model is denoted with Hiemstra*, and the corrected implementation as presented in Equation 7.1 is designated as Hiemstra_Corrected.

The comparison of the two implementations of the ranking model demonstrates that our correction consistently outperforms the current implementation contained in Terrier. The relative gain in MAP varies between 9.3 and 47.9 percent depending on the test collection. It can be seen that the difference between the two ranking functions is larger for structured document collections like CLEF TEL-BL and CLEF GIRT-4. Based on this empirical evaluation, it is concluded that the initial hypothesis is valid. As a result, the original implementation of the Hiemstra ranking model in Terrier has to be considered as being incorrect. The suggested correction as shown in Equation 7.1

Collection	Hiemstra*	Hiemstra_Corrected
TREC45-CR		
TREC7-AH	0.1791	0.2137 (+19.32%)
TREC8-AH	0.2190	0.2539 (+15.94%)
TREC2003-Robust	0.3269	0.3573 (+09.30%)
TREC2004-Robust	0.2580	0.2967 (+15.00%)
CLEF TEL-BL		
CLEF2008-AH	0.2455	0.3545 (+44.40%)
CLEF2009-AH	0.2243	0.3318 (+47.93%)
CLEF GIRT-4		
CLEF2003-DS	0.3198	0.4451 (+39.18%)
CLEF2004-DS	0.2802	0.3383 (+20.74%)
CLEF2005-DS	0.2932	0.3791 (+29.30%)
CLEF2006-DS	0.2467	0.3172 (+28.58%)

Table 7.6: Comparison of two implementations of Hiemstra's ranking model.

was submitted for integration into the Terrier software core[2]. Currently, the status of this integration is still pending.

7.4.2 Studying Interactions between Topics and Configurations

Topic sets used for empirical IR evaluation need to be designed carefully in order to achieve maximum utility for the comparison of systems. Bearing in mind that the present experimental design was evaluated against four different test collections with similar features of the topic sets and two major types of document collections, the resulting data can be used to study the variance of topic sets across identical and known system configurations. Based on that reasoning the variance of the four selected topic sets was analysed by using the optimal configurations that were presented in Section 7.3.

This analysis provides implications for two major applications. Firstly, evaluating the variance of a topic set on a fixed set of system configurations allows the assessment of

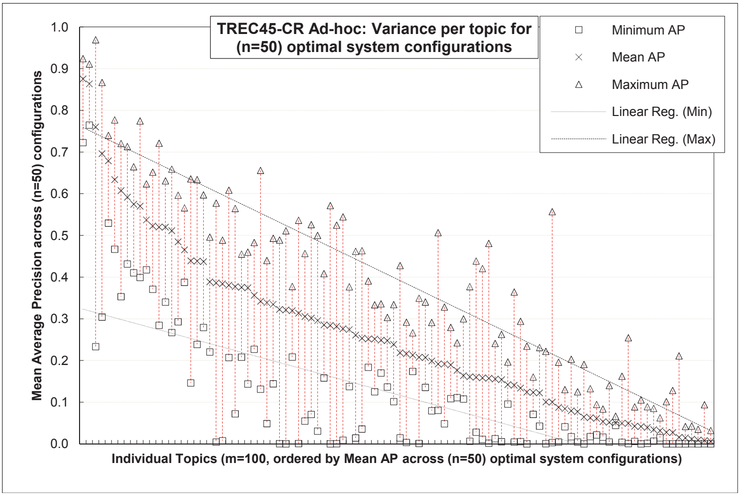
---

[2] `http://terrier.org/issues/browse/TR-183`, retrieved on March 1, 2012

the overall usefulness of the test collection. Secondly, the results can be used to define boundaries for topic-level optimisation. In theory, it is possible to select an optimal system configuration for each topic. The generated experiment set provides an upper limit for this approach. However, in order to put such a theory into practice, the topic sets have to be considerably larger than those that have been used here.

Figure 7.16 illustrates the variance for each of the topics in the TREC45-CR ad-hoc (see 7.16(a)), and robust (see 7.16(b)), test collections, which contain 100 query formulations each. The range between the best and the worst experiment configuration is presented by means of a red dashed line. Each topic set is organised in decreasing order in terms of average retrieval effectiveness across the 50 selected system configurations from Section 7.3. In order to visualise the lower and upper limit for the selected set of system configurations, each of the figures includes two linear regression curves. These regression models represent the worst case and the best case on the selected experiment set, i.e. it shows the maximum (resp. minimum) retrieval effectiveness if the best (resp. worst) system configuration is selected per topic. The linear curve above the average line shows the fitted linear regression model for the optimal retrieval effectiveness and the linear curve below the average illustrates the fitted linear regression model for the worst case scenario.

For the TREC45-CR ad-hoc collection the average retrieval effectiveness across the 50 selected experiments in terms of MAP is 0.2630. Assuming an algorithm would guess the system configuration for each topic, the worst case result on the experiment set would be 0.1185 and the best case would be 0.3955. Note that the best system configuration across all topics on this test collection resulted in a MAP values of 0.3001. As a result the retrieval effectiveness of the best configuration across all topics can be improved about 32 percent by selecting the best configuration for each topic. Figure 7.16(a) shows that the variance across the system configurations is low for topics with an average performance greater than 0.4, between 0.25 and 0.16, and below 0.12.



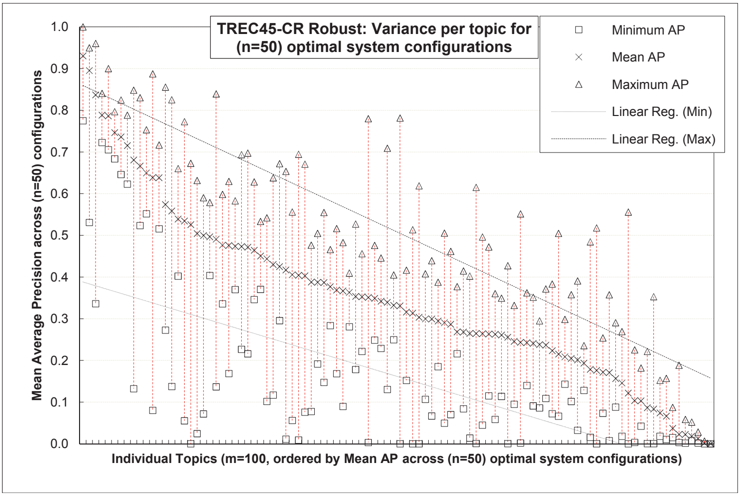


Figure 7.16: Variance in retrieval effectiveness across the selected optimal system configurations using PRF (see Section 7.3) on (a) the TREC45-CR ad-hoc and (b) robust test collections. The topics are sorted in decreasing order from left to right by using the average performance across the 50 system configurations as metric. Two linear regression curves illustrate the general trend for the theoretical maximum (resp. minimum) in retrieval effectiveness resulting from selecting the best (resp. worst) configuration per topic.

In contrast to that, topics with an average performance between 0.4 and 0.25 as well as between 0.16 and 0.12 display more variance across the system configuration under examination. This observation indicates that further studies could focus on investigating whether this effect is caused by specific features of the respective topics. Figure 7.16(a) also indicates that a considerable number of topics have a minimum performance close to zero. This implies that some of the selected system configurations fail on many topics even if an optimal system configuration achieved an average precision between 0.3 and 0.5. Further studies are needed in order to investigate whether these failures are due to specific components of the corresponding system configurations.

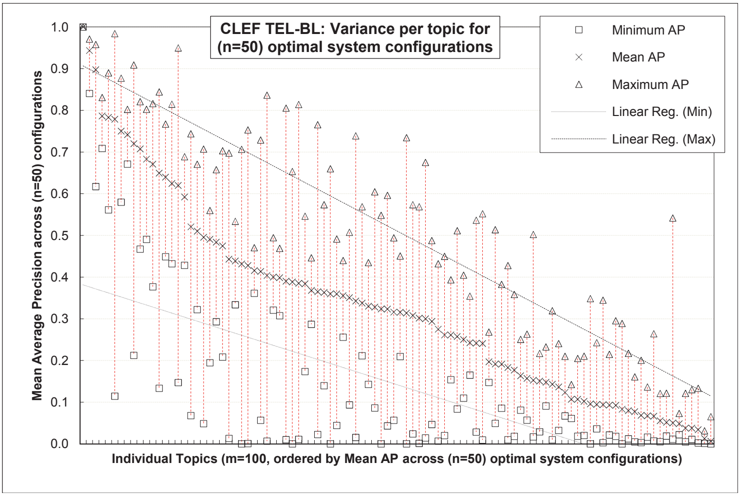
The average retrieval effectiveness across the same set of system configurations is 0.3526 for the TREC45-CR robust test collection. The best case scenario on the selected system configurations would result in a MAP value of 0.5085, where as the worst case scenario would return a MAP value of 0.1618. The optimal system configuration as reported in Appendix C.2 achieves a MAP of 0.4056. The potential gain in retrieval effectiveness from selecting one of the system configurations for each topic is about 25 percent. However, given the selected sample of system configurations, the risk for a significant decrease in retrieval performance using the selection strategy is very high. The relative loss could be up to 60 percent in comparison to the best system configuration on this test collection.

Figure 7.16(b) shows different levels of variance in retrieval performance across the topic set. But similar to the TREC45-CR ad-hoc test collection, the topics with high and low average performance have a lower range in retrieval performance, across the system configurations under examination, than topics with an average retrieval performance. There are less topics with a MAP value close to zero in the worst case scenario for this test collection. In general, the two linear regression models for the minimum and maximum performance across the tested system configurations cover a larger range than on the TREC45-CR ad-hoc test collection. This indicates that the TREC45-CR robust test collection might be more effective in discriminating the 50 selected system configurations than the TREC45-CR ad-hoc collection.

Figure 7.17(a) illustrates the topic variance for the selected set of system configurations on the CLEF TEL-BL test collection. It can be seen that the variance is much higher in general than on the other two test collections. Moreover, the number of topics with a retrieval effectiveness close to zero in the worst case scenario is significantly higher than on the other test collections. This indicates that particular system configurations fail on many topics.

This assumption is supported by two observations. First, for most of these topics the maximum performance of another system configuration is significantly higher than the average across all system configurations. Second, our analyses in Sections 7.2.2.2 and 7.3 demonstrated that the DPH ranking model seems to fail systematically on this collection. The average retrieval performance for the 50 selected system configuration is 0.3326. The optimal system configuration for the entire topic set achieves a MAP of 0.3934. The best case scenario results in a MAP value of 0.5107, whereas the worst case scenario achieves a MAP value of 0.1422. As a result, the potential for improvement in retrieval effectiveness is about 30 percent when the optimal system configuration is selected for each topic. The potential loss in performance is 65 percent, which is the highest value across the four test collections.

Figure 7.17(b) presents the variance for each topic in the CLEF GIRT-4 test collection. In general the range from minimum to maximum performance is lower than on the other test collections. Topics with high average performance ($AP > 0.5$) across the selected set of system configurations have smaller differences between minimum and maximum than topics with lower average performance. This observation is in line with the findings on the other test collections. The average performance across the 50 selected system configurations is 0.3963. This value could be increased to a MAP value of 0.5481 by selecting the optimal configuration for each topic. This corresponds to an increase of about 17 percent over the optimal system configuration with a MAP value of 0.4672 (see Appendix C.4).



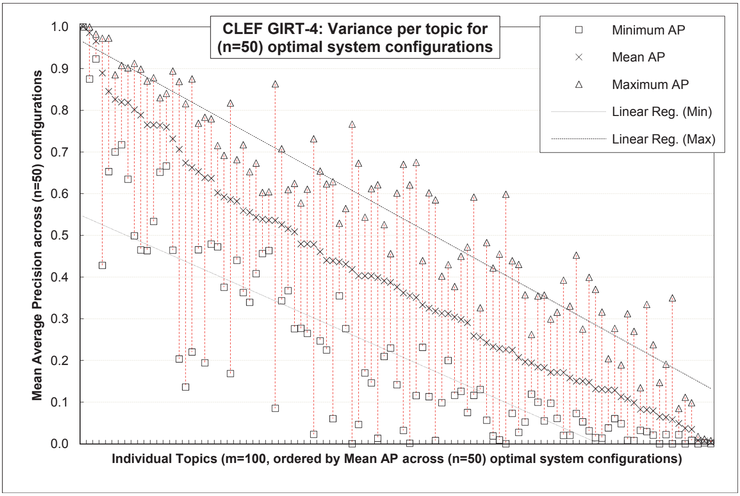


Figure 7.17: Variance in retrieval effectiveness across the selected optimal system configurations using PRF (see Section 7.3), on (a) the CLEF TEL-BL and (b) CLEF GIRT-4 test collections. The topics are sorted in decreasing order from left to right by using the average performance across the 50 system configurations as metric. Two linear regression curves illustrate the general trend for the theoretical maximum (resp. minimum) in retrieval effectiveness resulting from selecting the best (resp. worst) configuration per topic.

The potential loss for the topic-level configuration approach is about 54 percent, based on the MAP value of 0.2139 for the worst case scenario. Note that the curves that were fitted to the minimum and maximum performance using linear regression are almost parallel for the CLEF GIRT-4 test collection. This indicates that the distribution of topic difficulty measured in terms of the selected set of system configurations is almost uniform. However, it remains unclear whether such a uniform distribution of topic difficulty indicates higher utility of the test collection for the discrimination of system configurations.

7.5 Integration of Additional Components: Data Fusion

The experimental results that have been reported and analysed so far, contained three key components of IR systems. It was shown in Section 5.2 that the modular architecture of Xtrieval allows the integration of further system components. The empirical results on different kinds of ad-hoc evaluation tasks demonstrated that the retrieval performance of individual system configurations can be improved by the combination of different instances of components in a data fusion approach (see Section5.4). In order to answer the question of whether these observations are also valid for the optimal results from Section 7.3, a pairwise data fusion approach was integrated in the general experimental set-up.

This section illustrates the results of this integration into the automated component-level evaluation process. Data fusion (or merging) describes the process of combining multiple result sets from different systems, or system configurations, into a single list of documents. This combination is typically achieved with standard operators that incorporate two different types of information from the result lists: (1) the rank, i.e. the position of documents in the list, or (2) the retrieval status value (RSV), the value describing the confidence of the system that the document is relevant).

System1 Stemmer	Ranking	PRF (D, T)	System2 Stemmer	Ranking	PRF (D, T)	MAP
Krovetz	Lucene	KLCorr(9, 20)	UeaLite	Hiemstra	Bo2(6, 80)	0.3101
Porter	Hiemstra	Bo2(6, 100)	Krovetz	Lucene	KLCorr(9, 20)	0.3090
Porter	Lucene	KLCorr(3, 15)	UeaLite	Hiemstra	Bo2(6, 80)	0.3087
Porter	Lucene	KLCorr(3, 15)	Krovetz	Hiemstra	Bo2(6, 60)	0.3084
Krovetz	Lucene	KLCorr(9, 20)	Krovetz	Hiemstra	Bo2(6, 60)	0.3081
Porter	BM25	Bo2(3, 25)	Krovetz	Hiemstra	Bo2(6, 60)	0.3062
Porter	DLH13	Bo2(6, 30)	Krovetz	Hiemstra	Bo2(6, 60)	0.3056
Porter	TF-IDF	Bo2(3, 40)	Krovetz	Hiemstra	Bo2(6, 60)	0.3052
Porter	Hiemstra	Bo2(6, 100)	Krovetz	DLH13	Bo2(6, 40)	0.3049
Porter	Hiemstra	Bo2(6, 100)	Krovetz	Hiemstra	Bo2(6, 60)	0.3045

Table 7.7: Top-10 system configuration pairs that were combined with data fusion based on the Z-Score operator and tested on the TREC45 ad-hoc collection.

Most of these operators use the RSV in order to generate the merged result list. Such data fusion operators have been proposed [65] in line with early TREC evaluation experiments. Other empirical studies [102, 161] on test collections from the CLEF campaign demonstrated that the Z-Score operator is preferable in most situations. The Z-Score standardisation operator is based on the mean and the standard deviation of a population, i.e. the RSV's of documents in the result lists. In these experiments, a more complex variant of the Z-Score operator taken from [161, p. 240] was applied. It is illustrated in Definition 7.3, where α_i represents a weight that can be used to favour specific result lists, RSV_k is the retrieval status value for document k in result list i, and μ_i ρ_i, and min_i, represent the mean, the standard deviation, and the minimum RSV, for result list i.

$$\sum_{k=1}^{n} \frac{RSV_k - \mu_i}{\rho_i} + \frac{\mu_i - min_i}{\rho_i} \tag{7.3}$$

Two test collections and their corresponding optimal experiment sets were selected from Section 7.3 for the data fusion experiments. Since each of the experiment sets consists of only 50 different system configurations it was decided to test all possible pairs of combinations. This resulted in a total number of 1,225 data fusion experiments for each of the test collections: TREC45 ad-hoc and CLEF GIRT-4.

Table 7.7 lists the results for the 10 best data fusion results on the TREC45 ad-hoc test collection. The best system configuration returned a MAP value of 0.3001 on this collection (see Appendix C.1). It can be seen that the data fusion approach slightly improved retrieval effectiveness. However, only 41 out of 1,225 pairs of combinations achieved an improvement over the best individual system configuration. Table 7.7 demonstrates that the best pairs of configurations have some interesting features. The optimal system configuration for the TREC45 ad-hoc collection (see Section 7.3 and Appendix C.1) is part of 7 out of 10 pairs of system configurations. This observation indicates that the optimal configuration should be part of a pairwise data fusion approach. In 9 out of 10 data fusion experiments the Krovetz stemmer was part of at least one of the configurations. The Hiemstra ranking model appeared in one of the two system configurations for each of the top-performing data fusion experiments. Only 3 out of 5 stemmers, and 6 out of 10 ranking models, appear in any of the system configurations presented in Table 7.7. Bearing in mind the significant variance across the topics on this particular selection of experiments (see Figure 7.16 in Section 7.4.2), these observations are surprising. From the presented experimental results it was concluded that pairwise data fusion on the TREC45 ad-hoc collection using the Z-Score operator has only little impact on average retrieval effectiveness. In order to test whether this is due to the specific features of the TREC45 ad-hoc test collection, the same experiment was conducted on the CLEF GIRT-4 test collection.

Table 7.8 lists the results for the data fusion experiment on the CLEF GIRT-4 test collection. On this collection the best system configuration achieved a MAP value of 0.4672, as can be seen in Appendix C.4. This optimal configuration incorporates the Krovetz stemmer, the ranking model of Lucene, and a PRF configuration based on the KLCorrect PRF model using 30 terms from the top-20 documents. From the 1,225 data fusion experiments only 29 slightly improved retrieval effectiveness in comparison to the best system configuration. It can be seen that this system configuration

System1 Stemmer	Ranking	PRF (D, T)	System2 Stemmer	Ranking	PRF (D, T)	MAP
Porter	BB2	Bo2(20, 100)	Krovetz	Lucene	KLCorr(20, 30)	0.4816
Porter	In_expB2	Bo2(20, 100)	Krovetz	Lucene	KLCorr(20, 30)	0.4793
Porter	Lucene	KLCorr(20, 25)	Krovetz	Lucene	KLCorr(20, 30)	0.4786
Porter	TF-IDF	Bo2(20, 100)	Krovetz	Lucene	KLCorr(20, 30)	0.4772
Porter	DLH13	Bo2(15, 70)	Krovetz	Lucene	KLCorr(20, 30)	0.4763
Porter	IFB2	Bo2(20, 100)	Krovetz	Lucene	KLCorr(20, 30)	0.4759
Porter	Lucene	KLCorr(20, 25)	Krovetz	Lucene	KLCorr(20, 30)	0.4751
Porter	BM25	Bo2(20, 100)	Porter	BB2	Bo2(20, 100)	0.4748
Porter	LGD	KLCorr(15, 60)	Krovetz	Lucene	KLCorr(20, 30)	0.4746
Porter	Lucene	KLCorr(20, 25)	Porter	In_expB2	Bo2(20, 100)	0.4739

Table 7.8: Top-10 system configuration pairs that were combined with data fusion based on the Z-Score operator and tested on the CLEF GIRT-4 collection.

is part of 8 out of the top-10 data fusion experiments. The ranking model of Lucene appears in at least one of the two system configurations in 9 out of the 10 experiments listed in Table 7.8. From the 10 implementations of ranking models that were studied in our component-level evaluation experiments, 8 appear in the top-10 data fusion experiments. The results also show that only 2 out of the 5 implementations of stemming algorithms are part of any of the system configurations. In 8 out of the top-10 data fusion experiments the system configuration pairs contain the Porter and the Krovetz stemmer. These observations indicate that the retrieval effectiveness for the best system configuration can be improved marginally by: (1), combining the optimal system configuration with configurations that are based on other ranking models and a different stemmer and (2), using the Z-Score data fusion operator.

The comparison of the results for the data fusion approach on the test collections TREC45 ad-hoc and CLEF GIRT-4 demonstrated that the retrieval effectiveness can be slightly improved. The data fusion experiments were also repeated on the two remaining test collections used for our large-scale evaluation on component-level. The results were in line with those discussed in this section.

7.6 Archiving Experiment Configurations and Results

It has been pointed out in Sections 6.2.5 and 6.3 that the preservation of experiment configurations and results is necessary in order to ensure comparability in long-term evaluations. Such long-term evaluations are needed in order to track the progress of IR approaches and the evaluation methodology. Throughout this chapter, different subsets of the empirical data resulting from the large-scale experiment have been presented and different tools have been used to analyse these results. The discussed questions cover only a fraction of the full potential of the data set. For this reason the complete data set will be made publicly available for further studies. In addition to that, a selection of the optimal configurations on the original evaluation tasks will be published in the open platform *EvaluatIR.org*[3] for future reference.

Figure 7.18 illustrates a web-based tool [112], [196] that allows the archiving and exploration of component-level evaluation results in a flexible way. Other state-of-the-art platforms for the archival storage of evaluation results are maintained by the organisers of major campaigns like TREC[4], CLEF[5], or NTCIR[6]. All of these platforms provide access to most of the generated test collections and allow the downloading of the submitted experiments. They also supply rich information on the results of the archived evaluation experiments. But none of the systems allows access to or comparison of the actual system configurations. The reason is obvious: the system configurations are described in corresponding research articles that were published together with the evaluation results. This is a major drawback which is addressed by the web-based tool illustrated in Figure 7.18.

The key features of the developed tool for the component-level comparison of IR evaluation results are dicussed next. A more comfortable way to explore the key features

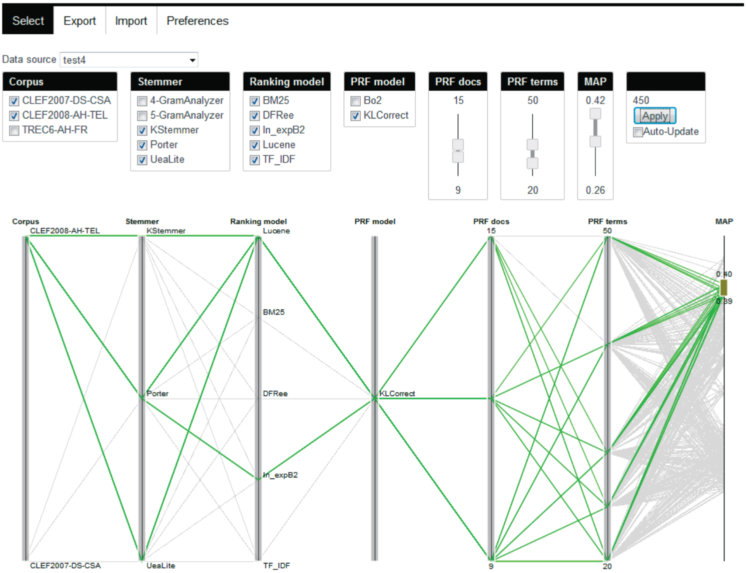
---

[3] `http://www.evaluatir.org/`, retrieved on March 1, 2012

[4] `http://trec.nist.gov/results/`, retrieved on March 1, 2012

[5] `http://direct.dei.unipd.it/`, retrieved on March 1, 2012

[6] `http://research.nii.ac.jp/ntcir/data/data-en.html`, retrieved on March 1, 2012



Data source test4

Corpus
☑ CLEF2007-DS-CSA
☑ CLEF2008-AH-TEL
☐ TREC6-AH-FR

Stemmer
☐ 4-GramAnalyzer
☐ 5-GramAnalyzer
☑ KStemmer
☑ Porter
☑ UeaLite

Ranking model
☑ BM25
☑ DFRee
☑ In_expB2
☑ Lucene
☑ TF_IDF

PRF model
☐ Bo2
☑ KLCorrect

PRF docs 15 9

PRF terms 50 20

MAP 0.42 0.26

450

Figure 7.18: Screen shot of the exploratory data archive for the presented empirical investigation of IR system component configurations evaluated on standard ad-hoc test collections.

of the tool is to watch explanatory tutorial videos[7]. The evaluation tool provides a flexible interface which allows the definition of the components that are part of a specific evaluation task. The data import interface allows the importing of CSV files, i.e. the component-level experiments have to be described in this plain text format. After the import of the data, the layout for the visual presentation can be defined. It is possible to re-arrange the columns of the input file and to assign new labels to each of the columns. The sequence of the columns determines the visual presentation of the parallel coordinates for the comparison of the system components.

Using this flexible model for the import of empirical evaluation results ensures that the system components under investigation are free from any restriction, i.e. the focus of each evaluation task can be configured individually by the definition of the com-

---

[7] `http://sachsmedia.tv/compeval/#tutorials`, retrieved on March 1, 2012

ponents under investigation. For example, a straightforward way to study the effect of the three key IR system components covered in this work would be to arrange these components as columns and show the different implementations as factors of the component. The tool also allows the definition of the data type for each of the columns. Alphanumerical factors are stored in a String data type, where as numerical factors, such as the number of documents for PRF, are stored as integers. Since some numerical factors and typical evaluation metrics are defined as floating point numbers, the system also supports this data type and allows the definition of the precision that is used for the visual presentation. Note that each data format has to be defined only once, directly after uploading the first CSV file. All subsequent files will be added to the previously defined data source if they have the same format.

The automated component-level evaluation of the key IR system components presented in this work resulted in 14,274 experiment configurations. Comparing all of these configurations at once is not only hard to realise technically, it also provides little additional insights beyond the findings that were presented in Sections 7.2.2 and 7.3. For that reason the web-based tool provides a mechanism to select specific subsets of experiments in order to restrict the total number of experiments that will be visualised. Once these experiments are presented in the form of parallel coordinates a researcher can start to explore the data set. Each of the coordinates can be restricted to an arbitrary range of values or factors. This allows the study of the effect of the components individually or in a desired combination.

The interactive nature of the visual presentation of the data facilitates ideas for the formulation of hypotheses about the data sets at hand. In order to test any formulated hypothesis, the selected data set has to be analysed using statistical tools that are appropriate for the corresponding hypothesis. Providing these statistical analyses is beyond the scope of the visualisation tool. Nevertheless, it supplies an interface to support this subsequent step. Once a selected part of the data set is visualised, this selection of experiments can be exported for further scientific analysis. Other IR researchers might be interested in the complete data set that was discussed in this work.

In order to simplify access to the entire collection of experiments, it will be provided independently from the visualisation tool.

7.7 Summary and Implications

Individual conclusions on the results of the presented and discussed experiments were provided in the corresponding subsections of this chapter. The purpose of this final section is to provide insights into the general impact of the presented experiments. The approach to address this is twofold. First, the scope of the experiments conducted in this work is emphasised in order to assess the results that were achieved. The basis of this evaluation is the comparison of the results obtained here with the results that were submitted to the original evaluation tasks at the TREC and CLEF campaigns. So it is possible to critically assess both the applied methodology and the results in terms of a standard retrieval effectiveness metric. Second, the most important conclusions of the results of the evaluation experiments in Sections 7.2.2, 7.3, and 7.4.2 are condensed and their implications on IR evaluation in general are outlined.

Before the results are presented and assessed with respect to the original evaluation tasks, the general scope of the experimental set-up needs to be summarised. The focus of our empirical investigation was on three major components of modern IR systems: stemming algorithms, ranking functions, and automatic pseudo-relevance feedback. Most of the tested algorithms are implemented in open-source frameworks that are accessible with Xtrieval. It was argued that the choice of the included algorithms reflects the state-of-the-art for ad-hoc search tasks. However, there are more complex implementations for each of the components under investigation in order to handle the more specific search tasks that have emerged recently. The developed approach to component-level evaluation is generic, i.e. it can be adapted for all possible search tasks once the major system components are determined for a particular problem under investigation. For this reason the present experiments do not cover all the latest and most sophisticated techniques to address problems like selective automatic relevance feedback. Instead, the emphasis was put on ad-hoc retrieval in order to demonstrate the potential of the combination of the most widely-used IR frameworks for automated IR evaluation at the component-level in general. All the presented experiments have been conducted in a closed laboratory environment, i.e. in order to improve retrieval effectiveness, no external sources from the web or elsewhere were used.

The results of the discussed experiments on the 10 original test collections from TREC and CLEF that formed the basis of the empirical study in this chapter are reported and analysed next. The following discussion is organised in three separate sections which focus on three different document collections. Each of these sections is based on four different types of experimental results which are abbreviated as follows:

- *@TREC / @CLEF* represents the retrieval effectiveness for the best experiment submitted to the corresponding evaluation task. In brackets we report the total number of submissions to each of the tasks in order to illustrate the scale of the competition for the best experiment.
- *@System* is the result for the optimal system configuration from Section 7.3 on the corresponding test collection. Note that each reported figure represents the best values out of 10,980 different IR system configurations.
- *@Data-Fusion* corresponds to the best result from the data fusion experiments in Section 7.5. Based on the pairwise combination of the 50 optimal experiments from Section 7.3, each of the values describes the outcome of the best merging experiment from a total number of 1,225.
- *@Topic-Select* indicates the upper limit of retrieval effectiveness, given the optimal set of system configurations from Section 7.3 for each test collection and assuming that an algorithm is able to identify the optimal system configuration for each of the topics individually.

Collection	Best Experiment	MAP	Collection	Best Experiment	MAP
TREC7-AH	@TREC (1/71)[8]	0.2961	TREC2003-Robust	@TREC (1/69)[8]	0.3922
	@System	0.3016		@System	0.4232
	@Data-Fusion	**0.3080**		@Data-Fusion	**0.4400**
	@Topic-Select	*0.3975*		@Topic-Select	*0.5233*
TREC8-AH	@TREC (1/107)[8]	**0.3207**	TREC2004-Robust	@TREC (1/58)[8]	**0.4227**
	@System	0.3029		@System	0.3943
	@Data-Fusion	0.3158		@Data-Fusion	0.4071
	@Topic-Select	*0.3935*		@Topic-Select	*0.4936*

Table 7.9: Comparison of the best experiments from TREC7-AH, TREC8-AH, TREC2003-Robust, and TREC2004-Robust with the best system configurations discussed in this work.

Table 7.9 summarises the corresponding figures for the four different topic sets of the TREC45-CR document collection. For each of these test collections the best of the first three figures is marked in bold. To separate the numbers for @Topic-Select from the rest of the values, these numbers are shown in italics. The respective effectiveness values for the best experiments from TREC were extracted from the web-based platform EvaluatIR.org [8].

In Table 7.9 it can be seen that from the three experimental set-ups generated in this work (@System, @Data-Fusion, @Topic-Select), the data fusion experiments resulted in the highest retrieval effectiveness. For the TREC7-AH and the TREC2003-Robust test collections the best individual system configurations from Section 7.3 (denoted @System) outperformed the best experiment submitted to the corresponding evaluation task at TREC.

The best individual experiment configurations submitted to the remaining test collections, TREC8-AH and TREC2004-Robust, achieved better retrieval effectiveness in terms of MAP than any of our experiments. However, the absolute difference between these submissions and our best experiments (@Data-Fusion) was only about 0.01 in terms of MAP.

---

[8] The presented number of submissions includes all experiments submitted to TREC that used the *title* and *description* parts of the topics and that were classified as *automatic* experiments. The listed MAP value belongs to the best experiment of this subset of submissions. Note that the absolute best experiment submitted to the corresponding TREC task may not be included in this subset.

Collection	Best Experiment	MAP	Collection	Best Experiment	MAP
CLEF2008-AH	@CLEF (1/38)[9]	0.3754	CLEF2009-AH	@CLEF (1/46)[9]	**0.4084**
	@System	0.4055		@System	0.3872
	@Data-Fusion	**0.4124**		@Data-Fusion	0.3976
	@Topic-Select	*0.5238*		@Topic-Select	*0.4976*

Table 7.10: Comparison of the best experiments from CLEF2008-AH & CLEF2009-AH with the best system configurations discussed in this work.

Given the repeated set-up for the evaluation task TREC8-AH and TREC2004-Robust, these observations indicate that participating research groups tuned their systems based on the outcome of the results from the previous year. The smaller difference between those results and the corresponding upper limits (@Topic-Select) also supports this interpretation. The relative differences between the top configurations generated in this work are either slightly better, or slightly worse, than the best experiments submitted to the corresponding TREC evaluation tasks. These empirical results suggest that the selection of IR system configurations reflects the state-of-the-art in ad-hoc retrieval evaluation on full-text document collections like TREC45-CR. A final point is the difference in the MAP values for the best system configuration @System and each corresponding upper limit @Topic-Select. The absolute difference between these two values shows very little variance across the test sets and lies between 0.09 and 0.1.

Table 7.10 shows the overall results on the CLEF TEL-BL test collection. The MAP values for the best experiments submitted to these two evaluation tasks were extracted from [53, p. 15] for CLEF2008-AH and from [54, p. 17] for CLEF2009-AH. It can be seen that the best data fusion experiment returned the best results of all experiments conducted within this work. On the CLEF2008-AH test collection this experiment outperformed every experiment submitted to the original evaluation task. The best overall submission to the CLEF2009-AH task was part of the original evaluation task. Neither the best individual experiment, nor the best data fusion experiment, was able to beat this baseline. However, there was only little absolute difference between these results

---

[9] The presented number of submissions includes all experiments submitted to the respective CLEF tasks. The total number of submissions to the tasks were extracted from [53, p. 16] and [54, p. 18].

Collection	Best Experiment	MAP	Collection	Best Experiment	MAP
CLEF2003-DS	@CLEF (1/6)[10]	0.5192	CLEF2004-DS	@CLEF (1/17)[10]	0.4053
	@System	0.5239		@System	0.4243
	@Data-Fusion	**0.5413**		@Data-Fusion	**0.4526**
	@Topic-Select	*0.6098*		@Topic-Select	*0.5122*
CLEF2005-DS	@CLEF (1/15)[10]	0.5065	CLEF2006-DS	@CLEF (1/8)[10]	0.4303
	@System	0.4995		@System	0.4347
	@Data-Fusion	**0.5098**		@Data-Fusion	**0.4400**
	@Topic-Select	*0.5777*		@Topic-Select	*0.4913*

Table 7.11: Comparison of the best experiments from CLEF2003-DS to CLEF2006-DS with the best system configurations discussed in this work.

and the best experiment from CLEF2009-AH. Again, better retrieval effectiveness of the experiments submitted to the original evaluation was found on the repeated evaluation task. It can also be seen from Table 7.10 that the absolute differences between the @System and the @Topic-Select values are about 0.11 for both of the topic sets.

The corresponding results on the CLEF GIRT-4 document collection are listed in Table 7.11. The MAP values for the best experiments submitted to these two evaluation tasks were extracted from the run statistics provided at the CLEF website[10] . The presented results show that the best data fusion experiment generated within this work returned the best results in terms of MAP on each of the test sets. It can be seen that the best system configuration from the experiments in Section 7.3 outperformed the best experiments submitted of all original evaluation tasks except CLEF2005-DS. The absolute difference in terms of MAP is small for all possible comparisons of the presented results on each of the test collections, except for the @Topic-Select values which are only reported to illustrate the upper limit of retrieval effectiveness.

Table 7.11 also demonstrates that the absolute number of experiments that were submitted to each task is much smaller than for the other test collections presented in Table 7.9 and Table 7.10. Nevertheless, the differences in terms of retrieval effectiveness between these experiments and the optimal system configurations are on the same

---

[10] The presented number of submissions includes all experiments submitted to the respective CLEF tasks. The total number of submissions to the tasks were extracted from [23, p. 1ff] for CLEF2003-DS, from [24, p. 1ff] for CLEF2004-DS, from [50, p. 9ff] for CLEF2005-DS, and from [51, p. 9] for CLEF2006-DS.

level as the differences on the other test collections. Since the system configurations created in this work were identical for all test collections, this observation indicates that, despite the little participation in the evaluation tasks on the CLEF GIRT-4 collection, the best experiments achieved strong results in terms of retrieval effectiveness. The absolute differences between the values for the @System and the @Topic-Select retrieval effectiveness vary between 0.06 and 0.09. These slightly smaller values indicate that the improvement of retrieval effectiveness is harder when the corresponding baseline experiments achieved better results.

In order to conclude the comparison of retrieval results of the original evaluation tasks with the experimental results generated in this work, the most important findings are summarised. On 7 out of 10 ad-hoc test collections the optimal configurations outperformed the best experiments of the respective evaluation task. The gap in terms of absolute difference in MAP to the best experiments on the remaining test collections was small, with a maximum difference of 0.0156. These results demonstrate that the IR system components included in this large-scale empirical analysis can serve as robust baselines for IR evaluation.

The essential findings of the exploratory data analysis of the large-scale experimental evaluation have been presented in Section 7.2.2.4. Here, the focus is on two important aspects. First, the exploratory data analysis allowed the visual identification of interaction effects between different components of an IR system and their configurations. For example, it has been observed that different types of probability distributions for some of the tested ranking models. A particular model which produced a bimodal distribution is likely to be dependent on another factor, i.e. the configuration of one of the other system components. Corresponding examples were discussed in Section 7.2.2. Second, the visual comparison of software implementations of ranking models empirically showed their theoretical relationship. To the knowledge of the present author, this is the first empirical evidence that closely related ranking models like TF-IDF and BM25 return almost identical probability distributions for a series of experiments conducted on different ad-hoc test collections.

The most essential observations on the optimal experiment configurations are the discussed next. Here, the focus was on the question of which individual system component configurations perform best on the different test collections. The experiments demonstrated that, given the optimal PRF set-up for a particular system configuration, there was only little variance across different ranking models. The same observation was made for each of the two tested groups of stemmers (n-gram and rule-based). Given a PRF set-up that works reasonably well, the differences between the implementations of system components are smoothed. However, the experiments also demonstrated that an optimal PRF configuration is hard to find, because it depends on the topics in the test collection rather than on other factors like IR system components.

The presented analyses of the created set of experiment configuration are only a first step in gaining a better understanding of the causal connections between different types of IR system components and retrieval effectiveness. More statistical analyses of the data are needed to exploit the full potential of the created experiment set. In order to stimulate further research in this direction, a web-based tool for archiving component-level evaluation experiments was developed (see Section 7.6). The presented tool complements existing platforms, like those maintained by the organisers of the major evaluation campaigns, or EvaluatIR.org. It provides straightforward interfaces for the creation of other component-level evaluation tasks, for the management of the resulting data, and allows visual analyses of the stored experiments. It is suggested that this tool or a similar application be implemented at major evaluation campaigns. It will further increase the transparency in IR evaluation and so support the advancement of the field in general.

8 Conclusion

This dissertation has proposed a generic approach to component-level evaluation of IR systems. The implemented strategy is based on the integration of different IR toolkits into the extensible retrieval and evaluation framework Xtrieval. A discussion of the results for various evaluation tasks has illustrated the potential benefits of a general framework like Xtrieval. Its meta-level design for accessing and combining different state-of-the-art retrieval toolkits allows fine-grained empirical studies at the component-level. By incorporating the findings of previous evaluation tasks that addressed particular components of IR systems, Xtrieval has been deployed to run a large series of grid experiments in order to provide a better understanding of the orchestration of components in modern IR systems.

In order to illustrate the problems with the current IR evaluation methodology, key elements of empirical evaluation, test collections and metrics for the assessment of the quality of search results have been discussed in detail. The theories and assumptions behind relevant IR system components have been carefully selected by an in-depth review of the state-of-the-art in IR theory.

A large-scale empirical experiment was designed and discussed to study the effect of state-of-the-art implementations of three key IR system components on retrieval performance. In contrast to typical empirical IR experiments which assessed the effect of one particular system component implementation with respect to some baseline model, this thesis covered a total of 13,176 unique IR system configurations. Each of these configurations was tested on four different test collections which contained 100 test queries each. The proposed test set-up allowed the determination of appropriate baselines for particular instances of system components which can be used for testing and verifying future components on the deployed test collections.

The experimental results have been analysed by means of an exploratory data analysis. This method allowed the identification of systematic failures of particular system components as well as the detailed comparison of the effects of, and interactions between, the selected IR system components. The obtained results have been compared to the optimal results for the corresponding evaluation tasks from TREC and CLEF. In order to stimulate further studies with respect to the impact of IR system components on retrieval effectiveness, the generated experimental data will be made publicly available to other researchers. This approach is intended to assist interested researchers in formulating hypotheses on particular effects that can be observed.

8.1 Results

Since this thesis has been devoted to IR evaluation at the component-level, the continuous development of the Xtrieval framework is one of the most important results. The foundation of all presented empirical results that are concerned with the comparison or combination of ranking models, stemming algorithms, or pseudo-relevance feedback mechanisms, is the integration of Lemur, Lucene, and Terrier into Xtrieval. Using Xtrieval for a variety of different kinds of evaluation tasks, like ad-hoc, domain-specific, and library record retrieval, image retrieval on photographs, question answering on speech transcripts, and video subject classification, helped to advance the framework.

The impact of the strategy of integrating different IR toolkits into a common framework has been substantiated with the presentation of strong results in empirical evaluation experiments. These include the different evaluation tasks above, as well as the results on the component-level experiment set that has been created in the course of this work. In fact it has been shown that the combination of different core retrieval models and the integration of different stemming algorithms is a good approach to improve the retrieval performance over systems that only used one particular implementation for ranking and stemming.

An analysis of recent approaches to component-level evaluation has demonstrated both conceptual problems and organisational issues. This thesis illustrated that the conceptual problems can be addressed with the integration strategy employed in Xtrieval. As a result it allowed the creation of an empirical experiment that covered several IR system components simultaneously. To the knowledge of the present author, this work is the first that analysed the effect of the three IR system components: text pre-processing, matching and ranking models, and pseudo-relevance feedback, on different types of document collections. Typical IR evaluation campaigns restrict the number of experiments that can be submitted for evaluation. With the the possibility of re-using existing test collections a total amount of 13,176 unique system configurations could be evaluated.

The results of this large experiment set have been studied by means of an exploratory data analysis. This approach allowed the visual interpretation of probability density functions for each of the tested implementations of the selected IR system components. As a result it is possible to identify which of the instances of a component are affected by other components. This is an extension of previous approaches to the analysis of experimental results which are typically focused on summary measures, like minimum, mean, and maximum retrieval performance. The presented results demonstrated that rule-based stemming performs significantly better than character n-gram stemming in general. In particular the Krovetz stemmer outperformed the other implementations on full-text collections like TREC45-CR. But the experiments also showed that the Porter stemmer works well across different types of document collections. The comparison of the tested ranking models revealed that recent ranking models do not outperform implementations which are based on classic models like the Vector Space model or the initial Probabilistic ranking model. The visual comparison of the probability distributions across the ranking models confirmed the relationships between implementations that are based on similar theoretical assumptions. In particular the tested instances that are based on the DFR model resulted in very similar result distributions. This empirical verification of the theoretical relatedness of ranking models is a unique feature of the employed method for the data analysis. Pseudo-relevance feedback was the third component of the study. The results suggest that the number of documents depend on the composition of the test topics. The optimal number of terms appears to be dependent on features of the document collection and the ranking model. Further statistical analysis of the generated data set are needed in order to verify this hypothesis. The analysis of the experiments has demonstrated that none of the component instances performed consistently better than other instances on the test collections.

The proposed method for the detailed comparison of IR system components has led to an unexpected observations. The analysis showed that some instances of ranking models consistently underperformed on a single or even on all test collections in comparison to other models or to the average of all models. This indication of systematic failure has been detailed further for one of the relevant ranking models. A detailed study of the foundations of the Hiemstra language model suggested that the corresponding implementation in Terrier was incorrect. This hypothesis was confirmed by implementing a corrected version of the ranking model and the empirical comparison with the previous implementation. The results showed a substantial improvement across all test collections used in the empirical experiment.

In order to preserve the results of the component-level evaluation experiment in particular, and to encourage further studies on the data set in general, a web-based tool has been developed. It is intended to illustrate that current impediments of component-level IR evaluation can be addressed with collaborative platforms. The tool allows the visual comparison of manually selected configurations and it features a flexible interface for the import and export of component-level evaluation results. It can be used to define new component-level evaluation tasks by importing the general set-up of the task in a simple format that can be adjusted to the specific components. This flexible approach ensures that a wide range of possible tasks can be managed within the tool. Similar platforms are already being maintained by the organisers of major evaluation campaigns, but none of them facilitates the combination of experiment configurations and corresponding results in this way.

8.2 Future Directions

In conclusion the impact and relevance of the discussed approaches and the obtained results need to be pointed out. This thesis provided two distinct contributions. First, the further development of the Xtrieval framework. And second, the creation of a large-scale empirical data set for the comparison of state-of-the-art implementations of IR system components.

The current version of the Xtrieval framework supports the access to widely-used IR toolkits. This integration is also particularly appealing from the perspective of teaching IR. The abstract architecture of Xtrieval enables the rapid design of IR evaluation experiments, but it is not limited to this use. It can also be used to implement a prototype search engine for any particular search task and collection. In combination with the accessibility of different ranking models that represent instances of general retrieval models, Xtrieval can be used for teaching IR courses in the academic field. In 2011, three groups of undergraduate students used the framework to design their own IR experiments in order to address specific search tasks. The students submitted results and working notes to the *Rich Speech Retrieval Task* [165] and the *Placing Task* [100] of the *Multimedia Benchmark Workshop*[1] 2011. This suggests that the Xtrieval framework can be integrated into teaching classes on IR methods.

---

[1] `http://www.multimediaeval.org/mediaeval2011/`, retrieved on March 1, 2012

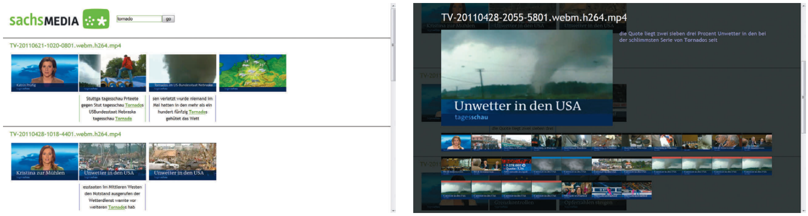

sachsMEDIA
tornado
go
TV-20110621-1020-0801.webm.h264.mp4
TV-20110428-1018-4401.webm.h264.mp4
Kristina zur Mühlen
Unwetter in den USA
TV-20110428-2055-5801.webm.h264.mp4
Unwetter in den USA
tagesschau

Figure 8.1: Prototype search application using Xtrieval as back end, redrawn from [99, p. 35].

Xtrieval is also being used for other applications, for instance, to provide access to audiovisual content, which is a major goal of the sachsMedia[2] research initiative. A prototype search application (see Figure 8.1) has been implemented to illustrate the potential support for editorial work in media companies. This application retrieves publicly available clips of a popular German news broadcast (*Tagesschau*) and triggers processes for automatic content extraction. The resulting meta-data is indexed using Xtrieval and so the clips are ready to be found.

The created component-level experiment set can be used for further studies concerning the interactions between IR system configurations. The following list outlines a few of these possibilities:

- *Component Variance*
  The analyses of the results have shown that the variance across IR system components in terms of retrieval performance is substantial. This suggests that an analysis of variance on the data set may allow the observation of which of the components affect retrieval performance more than others. Several levels of detail are possible. For instance, the emphasis could be put on entire test collections in order to draw general conclusions. Another possibility is to focus on individual topics which would address the question of whether specific groups

---

[2] `http://sachsmedia.tv/`, retrieved on March 1, 2012

of topics cause the variance across the instances of particular system components.

- *Component Interactions*
  Studying interaction effects between the IR system components and their tested instances is another subject for further studies. The obtained results indicated that particular component instances seem to favour particular configurations of the remaining components. For instance, ranking models like the Hiemstra language model achieved the best results when using many terms for PRF. In contrast to that, the Lucene ranking function worked well regardless of the used stemming approach on sparse library collections. A detailed analysis of these interactions between the tested components could result in general recommendations for the optimal configuration of an IR system when using a particular instances as point of origin.

- *Test Collection Variance*
  The present empirical evaluation was conducted on four ad-hoc test collections. But these test collections are based on three different document collections, i.e. two test collections have one document collection in common. This constellation allows to address the question of whether the topic sets or the document collection affects the ranking of IR systems more. Given the identical system configurations used here, a comparison of the rankings of these configurations on the different test collections will provide insights into this question.

- *Topic-Level Optimisation*
  In this thesis, the upper limit for optimal retrieval performance was defined as the (manual) selection of the optimal system configuration for each topic in a test collection. It was suggested that this optimisation at the topic-level could be addressed with machine learning techniques. This would require the identification of interactions between individual or particular groups of topics that share specific features that can be observed before or during the retrieval

process. Whether such groups of topics and the corresponding common features exist is the first question to be addressed.

- *Test Set Assessment*
  A specific feature of the generated data set is the investigation of identical system configurations on different test collections. This allows the reversal of one of the initial questions for the experiment: instead of finding the best system configuration for each of the test collections, one could also try to answer the question which test collection (or which part of each test collection) discriminates a selection of state-of-the-art IR system configurations best?

The generated experimental data set will be made publicly available in order to enable other researchers to address these and possibly further questions.

A final comment on how the results concerning component-level evaluation might be further developed. Currently, the Xtrieval framework allows the rapid configuration and comparison of key IR system components. One particular direction for future research is to establish automatic component-level baselines in IR evaluation. This could be realised by providing the Xtrieval framework as a web service or a web-based application. For every new evaluation task at TREC or other major evaluation campaigns, the framework could be used to generate baseline results before the actual tasks is opened. As a result, it would be possible to assess the submitted experiments with respect to these baselines. Based on the assumption that these baselines are improved by the submissions to the evaluation task, the new approaches have to be integrated into Xtrieval in order to track the overall progress over time.

Bibliography

[1] AGIRRE, E., DI NUNZIO, G. M., FERRO, N., MANDL, T., AND PETERS, C. CLEF 2008: Ad Hoc Track Overview. In *Proceedings of the 9th Cross-language evaluation forum conference on Evaluating systems for multilingual and multimodal information access* (Berlin, Heidelberg, 2009), Springer-Verlag, pp. 15–37.

[2] ALLAN, J., CARTERETTE, B., ASLAM, J. A., PAVLU, V., DACHEV, B., AND KANOULAS, E. Million Query Track 2007 Overview. In *Proceedings of the 16th Text Retrieval Conference (TREC 2007), NIST Special Publication 500-274* (2007).

[3] ALONSO, O., AND BAEZA-YATES, R. Design and Implementation of Relevance Assessments Using Crowdsourcing. In *Advances in Information Retrieval* (2011), P. Clough, C. Foley, C. Gurrin, G. J. Jones, W. Kraaij, H. Lee, and V. Mudoch, Eds., vol. 6611 of *Lecture Notes in Computer Science*, Springer, pp. 153–164.

[4] ALONSO, O., AND MIZZARO, S. Can We Get Rid of TREC Assessors? Using Mechanical Turk for Relevance Assessment. In *ACM SIGIR Workshop on The Future of IR Evaluation* (2009).

[5] ALONSO, O., ROSE, D. E., AND STEWART, B. Crowdsourcing for Relevance Evaluation. *SIGIR Forum 42*, 2 (2008), pp. 9–15.

[6] Amati, G., and Van Rijsbergen, C. J. Probabilistic Models of Information Retrieval Based on Measuring the Divergence from Randomness. *ACM Transactions on Information Systems 20*, 4 (2002), pp. 357–389.

[7] Arampatzis, A., Zagoris, K., and Chatzichristofis, S. A. Visual-Concept Search Solved? *IEEE Computer 43*, 6 (2010), pp. 76–78.

[8] Armstrong, T. G., Moffat, A., Webber, W., and Zobel, J. EvaluatIR: An Online Tool for Evaluating and Comparing IR Systems. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval* (New York, NY, USA, 2009), ACM, pp. 833–833.

[9] Armstrong, T. G., Moffat, A., Webber, W., and Zobel, J. Improvements That Don't Add Up: Ad-hoc Retrieval Results Since 1998. In *Proceedings of the 18th ACM conference on Information and knowledge management* (New York, NY, USA, 2009), ACM, pp. 601–610.

[10] Arni, T., Clough, P., Sanderson, M., and Grubinger, M. Overview of the ImageCLEFphoto 2008 Photographic Retrieval Task. In *Proceedings of the 9th Cross-language evaluation forum conference on Evaluating systems for multilingual and multimodal information access* (Berlin, Heidelberg, 2009), Springer-Verlag, pp. 500–511.

[11] Aslam, J. A., and Pavlu, V. A practical sampling strategy for efficient retrieval evaluation. Tech. rep., Northeastern University, 2008.

[12] Aslam, J. A., and Savell, R. On the Effectiveness of Evaluating Retrieval Systems in the Absence of Relevance Judgments. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval* (New York, NY, USA, 2003), ACM, pp. 361–362.

[13] Bailey, P., Craswell, N., and Hawking, D. Engineering a Multipurpose Test Collection for Web Retrieval Experiments. *Information Processing & Management 39*, 6 (November 2003), pp. 853–871.

[14] BAILEY, P., CRASWELL, N., SOBOROFF, I., THOMAS, P., DE VRIES, A. P., AND YILMAZ, E. Relevance Assessment: Are Judges Exchangeable and Does it Matter. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval* (New York, NY, USA, 2008), ACM, pp. 667–674.

[15] BANKS, D., OVER, P., AND ZHANG, N.-F. Blind Men and Elephants: Six Approaches to TREC data. *Information Retrieval 1*, 1-2 (1999), pp. 7–34.

[16] BEITZEL, S. M., JENSEN, E. C., CHOWDHURY, A., GROSSMAN, D., AND FRIEDER, O. Using Manually-built Web Directories for Automatic Evaluation of Known-item Retrieval. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval* (New York, NY, USA, 2003), ACM, pp. 373–374.

[17] BELKIN, N. J., ODDY, R. N., AND BROOKS, H. M. ASK for Information Retrieval: part I.: Background and Theory. In *Readings in information retrieval*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997, pp. 299–304.

[18] BJONER, S., AND ARDITO, S. C. Online Before the Internet, Part 1: Early Pioneers Tell Their Stories. *Searcher: The Magazine for Database Professionals 11*, 6 (2003), pp. 36–47.

[19] BODOFF, D., AND LI, P. Test Theory for Assessing IR Test Collections. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval* (New York, NY, USA, 2007), ACM, pp. 367–374.

[20] BOLDI, P., AND VIGNA, S. MG4J at TREC 2005. In *Proceedings of the 14th Text Retrieval Conference (TREC 2005), NIST Special Publication 500-266* (2005).

[21] BOOKSTEIN, A., AND SWANSON, D. R. Probabilistic Models for Automatic Indexing. *Journal of the American Society for Information Science 25*, 5 (1974), pp. 312–316.

[22] BORLUND, P. User-Centred Evaluation of Information Retrieval Systems. In *Information Retrieval: Searching in the 21st Century*. John Wiley & Sons, Ltd, 2009, ch. 2, pp. 21–37.

[23] BRASCHLER, M. Appendix B: Results for Core Tracks. In *CLEF 2003 Working Notes* (2003).

[24] BRASCHLER, M. Appendix A: Results of the Monolingual. Bilingual, Multilingual and Domain-Specific Tracks: Run Statistics. In *CLEF 2004 Working Notes* (2004).

[25] BRIN, S., AND PAGE, L. The Anatomy of a Large-scale Hypertextual Web Search Engine. In *Proceedings of the seventh international conference on World Wide Web* (Amsterdam, The Netherlands, The Netherlands, 1998), pp. 107–117.

[26] BÜTTCHER, S., CLARKE, C. L. A., AND SOBOROFF, I. The TREC 2006 Terabyte Track. In *Proceedings of the 15th Text Retrieval Conference (TREC 2006), NIST Special Publication 500-272* (2006).

[27] BUCKLEY, C., DIMMICK, D., SOBOROFF, I., AND VOORHEES, E. Bias and the limits of pooling for large collections. *Information Retrieval 10*, 6 (December 2007), pp. 491–508.

[28] BUCKLEY, C., SALTON, G., ALLAN, J., AND SINGHAL, A. Automatic Query Expansion Using SMART: TREC 3. In *Proceedings of the 3rd Text Retrieval Conference (TREC-3), NIST Special Publication 500-226* (1995), pp. 69–80.

[29] BUCKLEY, C., AND VOORHEES, E. M. Evaluating Evaluation Measure Stability. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval* (New York, NY, USA, 2000), ACM, pp. 33–40.

[30] BUCKLEY, C., AND VOORHEES, E. M. Retrieval Evaluation with Incomplete Information. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval* (New York, NY, USA, 2004), ACM, pp. 25–32.

[31] BURGES, C., SHAKED, T., RENSHAW, E., LAZIER, A., DEEDS, M., HAMILTON, N., AND HULLENDER, G. Learning to Rank Using Gradient Descent. In *Proceedings of the 22nd international conference on Machine learning* (New York, NY, USA, 2005), ACM, pp. 89–96.

[32] BURKOWSKI, F. J. Retrieval Activities in a Database Consisting of Heterogeneous Collections of Structured Text. In *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval* (New York, NY, USA, 1992), ACM, pp. 112–125.

[33] CALZOLARI, N., SORIA, C., GRATTA, R. D., GOGGI, S., QUOCHI, V., RUSSO, I., CHOUKRI, K., MARIANI, J., AND PIPERIDIS, S. The LREC Map of Language Resources and Technologies. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)* (2010), European Language Resources Association (ELRA), pp. 949–956.

[34] CARTERETTE, B. Robust Test Collections for Retrieval Evaluation. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval* (New York, NY, USA, 2007), ACM, pp. 55–62.

[35] CARTERETTE, B., AND ALLAN, J. Incremental Test Collections. In *Proceedings of the 14th ACM international conference on Information and knowledge management* (New York, NY, USA, 2005), ACM, pp. 680–687.

[36] CARTERETTE, B., ALLAN, J., AND SITARAMAN, R. Minimal Test Collections for Retrieval Evaluation. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval* (New York, NY, USA, 2006), ACM, pp. 268–275.

[37] CARTERETTE, B., PAVLU, V., FANG, H., AND KANOULAS, E. Million Query Track 2009 Overview. In *Proceedings of the 18th Text Retrieval Conference (TREC 2007), NIST Special Publication 500-278* (2009).

[38] CLARKE, C., CRASWELL, N., AND SOBOROFF, I. Overview of the TREC 2004 Terabyte Track. In *Proceedings of the 13th Text Retrieval Conference (TREC 2004), NIST Special Publication 500-261* (2004).

[39] CLEVERDON, C. W. ASLIB Cranfield Research Project: report on the first stage of an investigation into the comparative efficiency of indexing systems. Tech. rep., College of Aeronautics, Cranfield, 1960.

[40] CLEVERDON, C. W. The Cranfield Tests on Index Language Devices. In *Readings in information retrieval*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997, pp. 47–59.

[41] CLINCHANT, S., AND GAUSSIER, E. Bridging Language Modeling and Divergence from Randomness Models: A Log-Logistic Model for IR. In *Proceedings of the 2nd International Conference on Theory of Information Retrieval: Advances in Information Retrieval Theory* (Berlin, Heidelberg, 2009), ICTIR '09, Springer-Verlag, pp. 54–65.

[42] CLOUGH, P., GRUBINGER, M., DESELAERS, T., HANBURY, A., AND MÜLLER, H. Overview of the ImageCLEFphoto 2006 Photographic Retrieval Task. In *CLEF 2006 Working Notes* (2006).

[43] CORMACK, G. V., PALMER, C. R., AND CLARKE, C. L. A. Efficient Construction of Large Test Collections. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval* (New York, NY, USA, 1998), ACM, pp. 282–289.

[44] CRASWELL, N., AND HAWKING, D. Overview of the TREC-2002 Web Track. In *Proceedings of the 11th Text Retrieval Conference (TREC 2002), NIST Special Publication 500-251* (2002).

[45] CROFT, B. W., METZLER, D., AND STROHMAN, T. *Search Engines: Information Retrieval in Practice*. Pearson Education, 2010.

[46] CUMMINS, R., LALMAS, M., AND O'RIORDAN, C. The Limits of Retrieval Effectiveness. In *Proceedings of the 33rd European conference on Advances in information retrieval* (Berlin, Heidelberg, 2011), Springer-Verlag, pp. 277–282.

[47] DANG, H. T., KELLY, D., AND LIN, J. Overview of the TREC 2007 Question Answering Track. In *Proceedings of the 16th Text Retrieval Conference (TREC 2007), NIST Special Publication 500-274* (2007).

[48] DAWSON, J. L. Suffix Removal and Word Conflation. *ALLC Bulletin 2*, 3 (1974), pp. 33–46.

[49] DI NUNZIO, G., FERRO, N., MANDL, T., AND PETERS, C. CLEF 2006: Ad Hoc Track Overview. In *Evaluation of Multilingual and Multi-modal Information Retrieval*, C. Peters, P. Clough, F. Gey, J. Karlgren, B. Magnini, D. Oard, M. de Rijke, and M. Stempfhuber, Eds., vol. 4730 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2007, pp. 21–34.

[50] DI NUNZIO, G. M., AND FERRO, N. Appendix A: Results of the Core Tracks and Domain-Specific Tracks. In *CLEF 2005 Working Notes* (2005).

[51] DI NUNZIO, G. M., AND FERRO, N. Appendix C: Results of the Domain Specific Track. In *CLEF 2006 Working Notes* (2006).

[52] DI NUNZIO, G. M., AND FERRO, N. Appendix C: Results of the Domain Specific Track. In *CLEF 2007 Working Notes* (2007).

[53] DI NUNZIO, G. M., AND FERRO, N. Appendix A: Results of the TEL@CLEF Track. In *CLEF 2008 Working Notes* (2008).

[54] Di Nunzio, G. M., and Ferro, N. Appendix A: Results of the TEL@CLEF Track. In *CLEF 2009 Working Notes* (2009).

[55] Dittenbach, M., Pflugfelder, B., Pesenhofer, A., Roda, G., and Berger, H. SOIRE: A Service-oriented IR Evaluation Architecture. In *Proceedings of the 18th ACM conference on Information and knowledge management* (New York, NY, USA, 2009), ACM, pp. 2101–2102.

[56] Downie, J. S. The Music Information Retrieval Evaluation Exchange (2005–2007): A Window into Music Information Retrieval Research. *Acoustical Science and Technology 29*, 4 (2008), pp. 247–255.

[57] Efron, B., and Tibshirani, R. Bootstrap Methods for Standard Errors, Confidence Intervals, and Other Measures of Statistical Accuracy. *Statistical Science 1*, 1 (1986), pp. 54–75.

[58] Efron, M. Using Multiple Query Aspects to Build Test Collections without Human Relevance Judgments. In *Proceedings of the 31th European Conference on IR Research on Advances in Information Retrieval* (Berlin, Heidelberg, 2009), Springer-Verlag, pp. 276–287.

[59] Eibl, M., and Kürsten, J. Putting it All Together: The Xtrieval Framework at Grid@CLEF 2009. In *Proceedings of the 10th cross-language evaluation forum conference on Multilingual information access evaluation: text retrieval experiments* (Berlin, Heidelberg, 2009), Springer-Verlag, pp. 570–577.

[60] Fautsch, C., Dolamic, L., Abdou, S., and Savoy, J. Domain-Specific IR for German, English and Russian Languages. In *CLEF 2007 Working Notes* (2007).

[61] Ferro, N., and Harman, D. Dealing with MultiLingual Information Access: Grid Experiments at TrebleCLEF. In *Post-proceedings of the Forth Italian Research Conference on Digital Library Systems* (2008), DELOS: an Association for Digital Libraries, pp. 29–32.

[62] FERRO, N., AND HARMAN, D. CLEF 2009: Grid@CLEF pilot track overview. In *Proceedings of the 10th cross-language evaluation forum conference on Multilingual information access evaluation: text retrieval experiments* (Berlin, Heidelberg, 2009), Springer-Verlag, pp. 552–565.

[63] FINKEL, J. R., GRENAGER, T., AND MANNING, C. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics* (Stroudsburg, PA, USA, 2005), Association for Computational Linguistics, pp. 363–370.

[64] FISHER, R. A. *The Design of Experiments*. Oliver & Boyd, 1935.

[65] FOX, E. A., AND SHAW, J. A. Combination of Multiple Searches. In *Proceedings of the 2nd Text Retrieval Conference (TREC-2), NIST Special Publication 500-215* (1994), pp. 243–252.

[66] GRIFFITHS, A., LUCKHURST, H. C., AND WILLETT, P. Using Interdocument Similarity Information in Document Retrieval Systems. In *Readings in information retrieval*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997, pp. 365–373.

[67] GRUBINGER, M., CLOUGH, P., HANBURY, A., AND MÜLLER, H. Overview of the ImageCLEFphoto 2007 Photographic Retrieval Task. In *CLEF 2007 Working Notes* (2007).

[68] GRUBINGER, M., CLOUGH, P., HANBURY, A., AND MÜLLER, H. Overview of the ImageCLEFphoto 2007 Photographic Retrieval Task. In *Advances in Multilingual and Multimodal Information Retrieval* (Berlin, Heidelberg, 2008), Springer-Verlag, pp. 433–444.

[69] GRUBINGER, M., CLOUGH, P. D., MÜLLER, H., AND DESELAERS, T. The IAPR Benchmark: A New Evaluation Resource for Visual Information Systems. In *International Conference on Language Resources and Evaluation* (2006).

[70] HANBURY, A., AND MÜLLER, H. Automated Component-level Evaluation: Present and Future. In *Proceedings of the 2010 international conference on Multilingual and multimodal information access evaluation: cross-language evaluation forum* (Berlin, Heidelberg, 2010), Springer-Verlag, pp. 124–135.

[71] HARMAN, D. Overview of the Second Text REtrieval Conference (TREC-2). In *Proceedings of the 4th Text Retrieval Conference (TREC-2), NIST Special Publication 500-215* (1993).

[72] HARMAN, D. Overview of the Fourth Text REtrieval Conference (TREC-4). In *Proceedings of the 4th Text Retrieval Conference (TREC-4), NIST Special Publication 500-236* (1995).

[73] HARMAN, D. Panel: Building and Using Test Collections. In *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval* (New York, NY, USA, 1996), ACM, pp. 335–337.

[74] HARMAN, D. The TREC Conferences. In *Readings in information retrieval*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997, pp. 247–256.

[75] HARMAN, D., AND BUCKLEY, C. Overview of the Reliable Information Access Workshop. *Information Retrieval 12*, 6 (December 2009), pp. 615–641.

[76] HAUFF, C., HIEMSTRA, D., AZZOPARDI, L., AND DE JONG, F. A Case for Automatic System Evaluation. In *Advances in Information Retrieval*, C. Gurrin, Y. He, G. Kazai, U. Kruschwitz, S. Little, T. Roelleke, S. Rüger, and K. van Rijsbergen, Eds., vol. 5993 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2010, pp. 153–165.

[77] HAWKING, D. Overview of the TREC-7 Very Large Collection Track. In *Proceedings of the 7th Text Retrieval Conference (TREC-7), NIST Special Publication 500-242* (1998).

[78] HAWKING, D., AND THISTLEWAITE, P. Overview of TREC-6 Very Large Collection Track. In *Proceedings of the 6th Text Retrieval Conference (TREC-6), NIST Special Publication 500-240* (1997).

[79] HAWKING, D., THISTLEWAITE, P., AND HARMAN, D. Scaling Up the TREC Collection. *Information Retrieval 1*, 1-2 (1999), pp. 115–137.

[80] HAWKING, D., VOORHEES, E., CRASWELL, N., AND BAILEY, P. Overview of the TREC-8 Web Track. In *Proceedings of the 8th Text Retrieval Conference (TREC-8), NIST Special Publication 500-246* (1999).

[81] HEAPS, H. S. *Information Retrieval: Computational and Theoretical Aspects*. Academic Press, Inc., Orlando, FL, USA, 1978.

[82] HERSH, W., BUCKLEY, C., LEONE, T. J., AND HICKAM, D. OHSUMED: an Interactive Retrieval Evaluation and New Large Test Collection for Research. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval* (New York, NY, USA, 1994), Springer-Verlag New York, Inc., pp. 192–201.

[83] HIEMSTRA, D. *Using Language Models for Information Retrieval*. PhD thesis, Centre for Telematics and Information Technology, Enschede, 2001.

[84] HIEMSTRA, D. Information Retrieval Models. In *Information Retrieval: Searching in the 21st Century*. John Wiley & Sons, Ltd, 2009, ch. 1, pp. 1–19.

[85] HINTZE, J. L., AND NELSON, R. D. Violin Plots: A Box Plot-Density Trace Synergism. *The American Statistician 52*, 2 (May 1998), pp. 181–184.

[86] HOAGLIN, D., MOSTELLER, F., AND TUKEY, J. *Understanding Robust and Exploratory Data Analysis*. Wiley series in probability and mathematical statistics. Probability and mathematical statistics. Wiley, 1983.

[87] HULL, D. Using Statistical Testing in the Evaluation of Retrieval Experiments. In *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval* (New York, NY, USA, 1993), ACM, pp. 329–338.

[88] JENKINS, M.-C., AND SMITH, D. Conservative Stemming for Search and Indexing. Online publication at the University of East Anglia: `http://www.uea.ac.uk/polopoly_fs/1.85493!stemmer25feb.pdf`, 2005.

[89] JONES, K. S., WALKER, S., AND ROBERTSON, S. E. A Probabilistic Model of Information Retrieval: Development and Comparative Experiments. *Information Processessing & Management 36*, 6 (2000), pp. 779–808.

[90] JÄRVELIN, K., AND KEKÄLÄINEN, J. IR Evaluation Methods for Retrieving Highly Relevant Documents. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval* (New York, NY, USA, 2000), ACM, pp. 41–48.

[91] JÄRVELIN, K., AND KEKÄLÄINEN, J. Cumulated Gain-based Evaluation of IR Techniques. *ACM Transactions on Information Systems 20*, 4 (2002), pp. 422–446.

[92] KAMPSTRA, P. Beanplot: A Boxplot Alternative for Visual Comparison of Distributions. *Journal of Statistical Software 28*, 1 (November 2008), pp. 1–9.

[93] KANOULAS, E. *Building Reliable Test and Training Collections in Information Retrieval*. PhD thesis, College of Computer and Information Science, Boston, Massachusetts, 2009.

[94] KANTOR, P. B., AND VOORHEES, E. M. Report on the TREC-5 Confusion Track. In *Proceedings of the 5th Text Retrieval Conference (TREC-5), NIST Special Publication 500-238* (1996).

[95] KAZAI, G. In Search of Quality in Crowdsourcing for Search Engine Evaluation. In *Advances in Information Retrieval* (2011), P. Clough, C. Foley, C. Gurrin, G. J. Jones, W. Kraaij, H. Lee, and V. Mudoch, Eds., vol. 6611 of *Lecture Notes in Computer Science*, Springer, pp. 165–176.

[96] KAZAI, G., AND LALMAS, M. eXtended Cumulated Gain Measures for the Evaluation of Content-oriented XML Retrieval. *ACM Transactions on Information Systems 24*, 4 (2006), pp. 503–542.

[97] KENDALL, M. G. A New Measure of Rank Correlation. *Biometrika 30*, 1-2 (1938), pp. 81–93.

[98] KENT, A., BERRY, M. M., LUEHRS, F. U., AND PERRY, J. W. Machine Literature Searching VIII. Operational Criteria for Designing Information Retrieval Systems. *American Documentation 6*, 2 (1955), pp. 93–101.

[99] KNAUF, R., KÜRSTEN, J., KURZE, A., RITTER, M., BERGER, A., HEINICH, S., AND EIBL, M. Produce. annotate. archive. repurpose - Accelerating the Composition and Metadata Accumulation of TV Content. In *Proceedings of the 2011 ACM international workshop on Automated media analysis and production for novel TV services* (New York, NY, USA, 2011), ACM, pp. 31–36.

[100] KRIPPNER, F., MEIER, G., HARTMANN, J., AND KNAUF, R. Placing Media Items Using the Xtrieval Framework. In *Working Notes Proceedings of the MediaEval 2011 Workshop* (2011).

[101] KROVETZ, R. Viewing Morphology as an Inference Process. In *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval* (New York, NY, USA, 1993), ACM, pp. 191–202.

[102] KÜRSTEN, J. Systematisierung und Evaluierung von Clustering-Verfahren im Information Retrieval. Diploma thesis, Department of Computer Science, Chemnitz, Germany, 2006.

[103] KÜRSTEN, J. Chemnitz at CLEF 2009 Ad-Hoc TEL Task: Combining Different Retrieval Models and Addressing the Multilinguality. In *CLEF 2009 Working Notes* (2009).

[104] KÜRSTEN, J., AND EIBL, M. Monolingual Retrieval Experiments with a Domain-Specific Document Corpus at the Chemnitz University of Technology. In *Evaluation of Multilingual and Multi-modal Information Retrieval* (Berlin, Heidelberg, 2007), Springer-Verlag, pp. 178–185.

[105] KÜRSTEN, J., AND EIBL, M. Video Classification as IR Task: Experiments and Observations. In *Proceedings of the 10th international conference on Cross-language evaluation forum: multimedia experiments* (Berlin, Heidelberg, 2010), Springer-Verlag, pp. 377–384.

[106] KÜRSTEN, J., KUNDISCH, H., AND EIBL, M. QA Extension for Xtrieval: Contribution to the QAst track. In *CLEF 2008 Working Notes* (2008).

[107] KÜRSTEN, J., WILHELM, T., AND EIBL, M. CLEF 2008 Ad-Hoc Track: On-line Processing Experiments with Xtrieval. In *CLEF 2008 Working Notes* (2008).

[108] KÜRSTEN, J., WILHELM, T., AND EIBL, M. Extensible Retrieval and Evaluation Framework: Xtrieval. In *LWA 2008 - Workshop-Woche: Lernen, Wissen & Adaptivität* (2008), J. Baumeister and M. Atzmüller, Eds., Department of Computer Science, University of Würzburg, Germany, pp. 107–110.

[109] KÜRSTEN, J., WILHELM, T., AND EIBL, M. The XTRIEVAL Framework at CLEF 2007: Domain-Specific Track. In *Advances in Multilingual and Multimodal Information Retrieval* (Berlin, Heidelberg, 2008), Springer-Verlag, pp. 174–181.

[110] Kürsten, J., Wilhelm, T., and Eibl, M. CLEF 2008 Ad-hoc Track: Comparing and Combining Different IR Approaches. In *Proceedings of the 9th Cross-language evaluation forum conference on Evaluating systems for multilingual and multimodal information access* (Berlin, Heidelberg, 2009), Springer-Verlag, pp. 75–82.

[111] Kürsten, J., Wilhelm, T., and Eibl, M. The Xtrieval Framework at CLEF 2008: Domain-specific Track. In *Proceedings of the 9th Cross-language evaluation forum conference on Evaluating systems for multilingual and multimodal information access* (Berlin, Heidelberg, 2009), Springer-Verlag, pp. 215–218.

[112] Kürsten, J., Wilhelm, T., and Eibl, M. Vergleich von IR Systemkonfigurationen auf Komponentenebene. In *Proceedings of the 12th international symposium of information science* (Konstanz, 2011), Verlag Werner Hülsbusch, pp. 47–59.

[113] Kundisch, H. Question Answering Erweiterung für das Xtrieval-Framework. Student research project, Department of Computer Science, Chemnitz, Germany, 2009.

[114] Kwok, K., Grunfeld, L., and Lewis, D. TREC-3 Ad-Hoc, Routing Retrieval and Thresholding Experiments Using PIRCS. In *Proceedings of the 3rd Text Retrieval Conference (TREC-3), NIST Special Publication 500-226* (1995), pp. 247–256.

[115] Larson, M., Newman, E., and Jones, G. J. F. Overview of VideoCLEF 2009: New Perspectives on Speech-based Multimedia Content Enrichment. In *Proceedings of the 10th international conference on Cross-language evaluation forum: multimedia experiments* (Berlin, Heidelberg, 2010), Springer-Verlag, pp. 354–368.

[116] LE, J., EDMONDS, A., HESTER, V., AND BIEWALD, L. Ensuring Quality in Crowdsourced Search Relevance Evaluation: The Effects of Training Question Distribution. In *ACM SIGIR Workshop on Crowdsourcing for Search Evaluation* (2010), pp. 17–20.

[117] LEDWITH, R. On the Difficulties of Applying the Results of Information Retrieval Research to Aid in the Searching of Large Scientific Databases. *Information Processing & Management 28*, 4 (1992), pp. 451–455.

[118] LIN, W.-H., AND HAUPTMANN, A. Revisiting the Effect of Topic Set Size on Retrieval Error. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval* (New York, NY, USA, 2005), ACM, pp. 637–638.

[119] LOVINS, J. B. Development of a Stemming Algorithm. *Mechanical Translation and Computational Linguistics 11*, 1-2 (1968), pp. 22–31.

[120] LUHN, H. P. The Automatic Creation of Literature Abstracts. *IBM Journal of Research and Development 2*, 2 (1958), pp. 159–165.

[121] MACDONALD, C., AND OUNIS, I. The TREC Blogs06 Collection : Creating and Analysing a Blog Test Collection. Tech. Rep. TR-2006-224, Department of Computing Science, University of Glasgow, 2006.

[122] MACDONALD, C., OUNIS, I., AND SOBOROFF, I. Overview of the TREC2009 Blog Track. In *Proceedings of the 18th Text Retrieval Conference (TREC 2009), NIST Special Publication 500-278* (2009).

[123] MAGDY, W., AND JONES, G. J. PRES: A Score Metric for Evaluating Recall-oriented Information Retrieval Applications. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval* (New York, NY, USA, 2010), ACM, pp. 611–618.

[124] MANDL, T. Recent Developments in the Evaluation of Information Retrieval Systems: Moving Towards Diversity and Practical Relevance. *Informatica (Slovenia) 32*, 1 (2008), pp. 27–38.

[125] MANMATHA, R., RATH, T., AND FENG, F. Modeling Score Distributions for Combining the Outputs of Search Engines. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval* (New York, NY, USA, 2001), ACM, pp. 267–275.

[126] MANNING, C. D., RAGHAVAN, P., AND SCHÜTZE, H. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[127] MCCANDLESS, M., HATCHER, E., AND GOSPODNETIĆ, O. *Lucene in Action*. Manning, 2010.

[128] MCNAMEE, P., NICHOLAS, C., AND MAYFIELD, J. Addressing Morphological Variation in Alphabetic Languages. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval* (New York, NY, USA, 2009), ACM, pp. 75–82.

[129] METZLER, D., AND CROFT, W. B. Combining the Language Model and Inference Network Approaches to Retrieval. *Information Processing & Management 40*, 5 (2004), pp. 735–750.

[130] MIDDLETON, C., AND BAEZA-YATES, R. A Comparison of Open Source Search Engines, 2007.

[131] MITAMURA, T., NYBERG, E., SHIMA, H., KATO, T., MORI, T., YEW LIN, C., SONG, R., JIE LIN, C., SAKAI, T., JI, D., AND KANDO, N. Overview of the NTCIR-7 ACLIA Tasks: Advanced Cross-Lingual Information Access. In *Proceedings of the 7th NTCIR Workshop Meeting on Evaluation of Information Access Technologies: Information Retrieval, Question Answering, and Cross-Lingual Information Access* (2008).

[132] MIZZARO, S., AND ROBERTSON, S. Hits hits TREC: Exploring IR Evaluation Results with Network Analysis. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval* (New York, NY, USA, 2007), ACM, pp. 479–486.

[133] MOFFAT, A., AND ZOBEL, J. Rank-biased Precision for Measurement of Retrieval Effectiveness. *ACM Transactions Information Systems 27*, 1 (2008), pp. 2:1–2:27.

[134] MOOERS, C. N. Information Retrieval Viewed as Temporal Signaling. In *Proceedings of the International Congress of Mathematicians* (1950), vol. 1, pp. 572–573.

[135] OUNIS, I., AMATI, G., PLACHOURAS, V., HE, H., MACDONALD, C., AND LIOMA, C. Terrier: A High Performance and Scalable Information Retrieval Platform. *Proceedings of OSIR Workshop '06* (2006).

[136] OUNIS, I., DE RIJKE, M., MACDONALD, C., MISHNE, G., AND SOBOROFF, I. Overview of the TREC-2006 Blog Track. In *Proceedings of the 15th Text Retrieval Conference (TREC 2006), NIST Special Publication 500-272* (2006).

[137] PAICE, C. D. Another Stemmer. *SIGIR Forum 24*, 3 (1990), pp. 56–61.

[138] PARAMITA, M. L., SANDERSON, M., AND CLOUGH, P. Diversity in Photo Retrieval: Overview of the ImageCLEFPhoto Task 2009. In *Proceedings of the 10th international conference on Cross-language evaluation forum: multimedia experiments* (Berlin, Heidelberg, 2010), Springer-Verlag, pp. 45–59.

[139] PETRAS, V., AND BAERISCH, S. The Domain-specific Track at CLEF 2008. In *Proceedings of the 9th Cross-language evaluation forum conference on Evaluating systems for multilingual and multimodal information access* (Berlin, Heidelberg, 2009), Springer-Verlag, pp. 186–198.

[140] PORTER, M. F. An Algorithm for Suffix Stripping. In *Readings in information retrieval*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997, pp. 313–316.

[141] RAGHAVAN, V., BOLLMANN, P., AND JUNG, G. S. A Critical Investigation of Recall and Precision as Measures of Retrieval System Performance. *ACM Transactions on Information Systems 7*, 3 (1989), pp. 205–229.

[142] ROBERT MCGILL, J. W. T., AND LARSEN, W. A. Variations of Box Plots. *The American Statistician 32*, 1 (February 1978), pp. 12–16.

[143] ROBERTSON, S. On GMAP: And Other Transformations. In *Proceedings of the 15th ACM international conference on Information and knowledge management* (New York, NY, USA, 2006), ACM, pp. 78–83.

[144] ROBERTSON, S. On the History of Evaluation in IR. *Jorunal of Information Science 34*, 4 (2008), pp. 439–456.

[145] ROBERTSON, S. Richer Theories, Richer Experiments. In *Proceedings of the SIGIR 2009 Workshop on the Future of IR Evaluation* (2009), p. 4.

[146] ROBERTSON, S. On the Contributions of Topics to System Evaluation. In *Advances in Information Retrieval*, P. Clough, C. Foley, C. Gurrin, G. Jones, W. Kraaij, H. Lee, and V. Mudoch, Eds., vol. 6611 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2011, pp. 129–140.

[147] ROBERTSON, S. E. The Probability Ranking Principle in IR. In *Readings in information retrieval*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997, pp. 281–286.

[148] ROBERTSON, S. E., AND JONES, K. S. Relevance Weighting of Search Terms. *Journal of the American Society for Information Science 27*, 3 (1976), pp. 129–146.

[149] ROBERTSON, S. E., AND WALKER, S. Some Simple Effective Approximations to the 2-Poisson Model for Probabilistic Weighted Retrieval. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval* (New York, NY, USA, 1994), SIGIR '94, Springer-Verlag New York, Inc., pp. 232–241.

[150] ROCCHIO, J. J. Relevance Feedback in Information Retrieval. In *The SMART Retrieval System - Experiments in Automatic Document Processing*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1971, pp. 313–323.

[151] RODA, G., TAIT, J., PIROI, F., AND ZENZ, V. CLEF-IP 2009: Retrieval Experiments in the Intellectual Property Domain. In *Proceedings of the 10th cross-language evaluation forum conference on Multilingual information access evaluation: text retrieval experiments* (Berlin, Heidelberg, 2009), Springer-Verlag, pp. 385–409.

[152] SAKAI, T. New Performance Metrics based on Multigrade Relevance: Their Application to Question Answering. In *Proceedings of NTCIR-4* (2004).

[153] SAKAI, T. Evaluating Evaluation Metrics Based on the Bootstrap. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval* (New York, NY, USA, 2006), ACM, pp. 525–532.

[154] SAKAI, T. Alternatives to Bpref. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval* (New York, NY, USA, 2007), ACM, pp. 71–78.

[155] SAKAI, T. Comparing Metrics across TREC and NTCIR:: the Robustness to Pool Depth Bias. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval* (New York, NY, USA, 2008), ACM, pp. 691–692.

[156] SALTON, G., AND MCGILL, M. J. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA, 1986.

[157] SALTON, G., AND YANG, C. S. On the Specification of Term Values in Automatic Indexing. *Journal of Documentation 29*, 4 (1973), pp. 351–372.

[158] SANDERSON, M. Test Collection Based Evaluation of Information Retrieval Systems. In *Foundations and Trends® in Information Retrieval*, vol. 4. Now Publishers, 2010, pp. 247–375.

[159] SANDERSON, M., AND ZOBEL, J. Information Retrieval System Evaluation: Effort, Sensitivity, and Reliability. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval* (New York, NY, USA, 2005), ACM, pp. 162–169.

[160] SAVOY, J. Statistical Inference in Retrieval Effectiveness Evaluation. *Information Processing and Management 33*, 4 (1997), pp. 495–512.

[161] SAVOY, J. Data Fusion for Effective European Monolingual Information Retrieval. In *Multilingual Information Access for Text, Speech and Images*, C. Peters, P. Clough, J. Gonzalo, G. Jones, M. Kluck, and B. Magnini, Eds., vol. 3491 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2005, pp. 921–921.

[162] SAVOY, J. Light Stemming Approaches for the French, Portuguese, German and Hungarian Languages. In *Proceedings of the 2006 ACM symposium on Applied computing* (New York, NY, USA, 2006), ACM, pp. 1031–1035.

[163] SAVOY, J., AND ABDOU, S. UniNE at CLEF 2006: Experiments with Monolingual, Bilingual, Domain- Specific and Robust Retrieval. In *CLEF 2006 Working Notes* (2006).

[164] SCHAMBER, L., EISENBERG, M., AND NILAN, M. S. A Re-examination of Relevance: Toward a Dynamic, Situational Definition. *Information Processessing and Management 26*, 6 (1990), pp. 755–776.

[165] SCHMIDT, K., KÖRNER, T., HEINICH, S., AND WILHELM, T. A Two-step Approach to Video Retrieval Based on ASR Transcriptions. In *Working Notes Proceedings of the MediaEval 2011 Workshop* (2011).

[166] SHAW, W. M., BURGIN, R., AND HOWELL, P. Performance Standards and Evaluations in IR Test Collections: Vector-space and Other Retrieval Models. *Information Processing & Management 33*, 1 (1997), pp. 15–36.

[167] SMUCKER, M. D., ALLAN, J., AND CARTERETTE, B. A Comparison of Statistical Significance Tests for Information Retrieval Evaluation. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management* (New York, NY, USA, 2007), ACM, pp. 623–632.

[168] SNOEK, C. G. M., WORRING, M., VAN GEMERT, J. C., GEUSEBROEK, J.-M., AND SMEULDERS, A. W. M. The Challenge Problem for Automated Detection of 101 Semantic Concepts in Multimedia. In *Proceedings of the 14th annual ACM international conference on Multimedia* (New York, NY, USA, 2006), ACM, pp. 421–430.

[169] SOBOROFF, I., NICHOLAS, C., AND CAHAN, P. Ranking Retrieval Systems without Relevance Judgments. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval* (New York, NY, USA, 2001), ACM, pp. 66–73.

[170] SPÄRCK-JONES, K. Automatic Indexing. *Journal of Documentation 30*, 4 (1974), pp. 393–432.

[171] SPÄRCK-JONES, K., AND VAN RIJSBERGEN, C. J. Report on the Need for and Provision of an 'Ideal' Information Retrieval Test Collection. Tech. Rep. 5266, University of Cambridge, 1975.

[172] SPÄRCK-JONES, K., AND VAN RIJSBERGEN, C. J. Information Retrieval Test Collections. *Journal of Documentation 32*, 1 (1976), pp. 59–75.

[173] SWETS, J. A. Information Retrieval Systems. *Science 141* (1963), pp. 245–250.

[174] SWETS, J. A. Effectiveness of Information Retrieval Methods. *American Documentation 20*, 1 (1969), pp. 72–89.

[175] THOMPSON, P., TURTLE, H., YANG, B., AND FLOOD, J. TREC-3 Ad-hoc Retrieval and Routing Experiments Using the WIN System. In *Proceedings of the 3rd Text Retrieval Conference (TREC-3), NIST Special Publication 500-226* (1995), pp. 211–218.

[176] TSIKRIKA, T., AND KLUDAS, J. Overview of the WikipediaMM Task at ImageCLEF 2008. In *Proceedings of the 9th Cross-language evaluation forum conference on Evaluating systems for multilingual and multimodal information access* (Berlin, Heidelberg, 2009), Springer-Verlag, pp. 539–550.

[177] TURMO, J., COMAS, P. R., ROSSET, S., LAMEL, L., MOREAU, N., AND MOSTEFA, D. Overview of QAST 2008. In *Proceedings of the 9th Cross-language evaluation forum conference on Evaluating systems for multilingual and multimodal information access* (Berlin, Heidelberg, 2009), Springer-Verlag, pp. 314–324.

[178] TURTLE, H., AND CROFT, W. B. Evaluation of an Inference Network-based Retrieval Model. *ACM Transactions on Information Systems 9*, 3 (1991), pp. 187–222.

[179] VAN RIJSBERGEN, C. J. Foundation of Evaluation. *Journal of Documentation 30*, 4 (1974), pp. 365–373.

[180] VAN RIJSBERGEN, C. J. *Information Retrieval*. Butterworths, London, UK, 1979.

[181] VOORHEES, E. M. Variations in Relevance Judgments and the Measurement of Retrieval Effectiveness. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval* (New York, NY, USA, 1998), ACM, pp. 315–323.

[182] VOORHEES, E. M. Overview of TREC 2002. In *Proceedings of the 11th Text Retrieval Conference (TREC 2002), NIST Special Publication 500-251* (2001), pp. 1–15.

[183] VOORHEES, E. M. The Philosophy of Information Retrieval Evaluation. In *Revised Papers from the Second Workshop of the Cross-Language Evaluation Forum on Evaluation of Cross-Language Information Retrieval Systems* (London, UK, 2002), CLEF '01, Springer-Verlag, pp. 355–370.

[184] VOORHEES, E. M. Overview of the TREC 2004 Robust Retrieval Track. In *Proceedings of the 13th Text Retrieval Conference (TREC-13), NIST Special Publication 500-261* (2005).

[185] VOORHEES, E. M. Overview of TREC 2007. In *Proceedings of the 16th Text Retrieval Conference (TREC-16), NIST Special Publication 500-274* (2007).

[186] VOORHEES, E. M. Topic Set Size Redux. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval* (New York, NY, USA, 2009), SIGIR '09, ACM, pp. 806–807.

[187] VOORHEES, E. M., AND HARMAN, D. Overview of the Sixth Text REtrieval Conference (TREC-6). *Information Processing & Management 36*, 1 (January 2000), pp. 3–35.

[188] WEBBER, W., MOFFAT, A., AND ZOBEL, J. Score Standardization for Inter-collection Comparison of Retrieval Systems. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval* (New York, NY, USA, 2008), ACM, pp. 51–58.

[189] WEBBER, W., AND PARK, L. A. F. Score Adjustment for Correction of Pooling Bias. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval* (New York, NY, USA, 2009), ACM, pp. 444–451.

[190] WILBUR, W. J. Non-parametric Significance Tests of Retrieval Performance Comparisons. *Journal of Information Science 20*, 4 (1994), pp. 270–284.

[191] WILCOXON, F. Individual Comparisons by Ranking Methods. *Biometrics Bulletin 1*, 6 (1945), pp. 80–83.

[192] WILHELM, T. Entwurf und Implementierung eines Frameworks zur Analyse und Evaluation von Verfahren im Information Retrieval. Diploma thesis, Department of Computer Science, Chemnitz, Germany, 2008.

[193] WILHELM, T., AND EIBL, M. ImageCLEF 2006 Experiments at the Chemnitz Technical University. In *CLEF 2006 Working Notes* (2006).

[194] WILHELM, T., KÜRSTEN, J., AND EIBL, M. Experiments for the ImageCLEF 2007 Photographic Retrieval Task. In *CLEF 2007 Working Notes* (2007).

[195] WILHELM, T., KÜRSTEN, J., AND EIBL, M. The Xtrieval Framework at CLEF 2008: ImageCLEF Photographic Retrieval Task. In *CLEF 2008 Working Notes* (2008).

[196] WILHELM, T., KÜRSTEN, J., AND EIBL, M. A Tool for Comparative IR Evaluation on Component Level. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval* (New York, NY, USA, 2011), ACM, pp. 1291–1292.

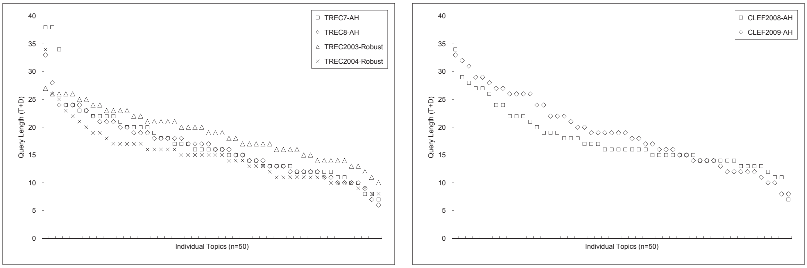
[197] YILMAZ, E., AND ASLAM, J. A. Estimating Average Precision with Incomplete and Imperfect Judgments. In *Proceedings of the 15th ACM international conference on Information and knowledge management* (New York, NY, USA, 2006), ACM, pp. 102–111.

[198] ZHAI, C., AND LAFFERTY, J. A Study of Smoothing Methods for Language Models Applied to Information Retrieval. *ACM Transactions on Information Systems 22*, 2 (2004), pp. 179–214.

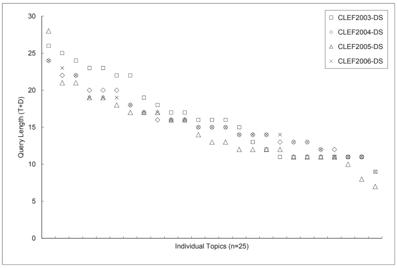
[199] ZOBEL, J. How Reliable Are the Results of Large-scale Information Retrieval Experiments? In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval* (New York, NY, USA, 1998), ACM, pp. 307–314.

A Topic Set Statistics

A.1 Query Length



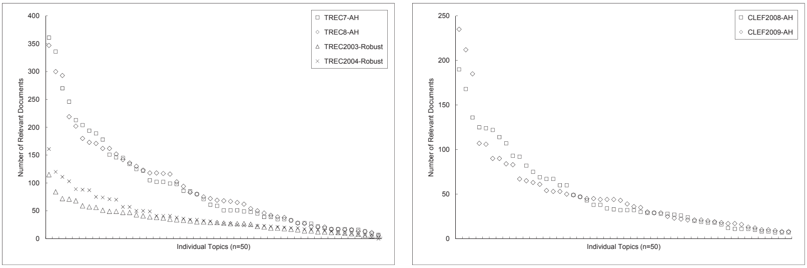
(a) TREC45-CR (b) CLEF TEL-BL



(c) CLEF GIRT-4

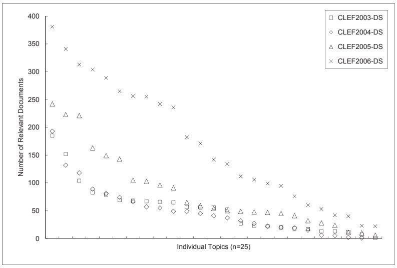
Figure A.1: Query lengths (title + description) for individual topics (order by decreasing query length) for original topic sets from TREC and CLEF: (a) TREC45-CR, (b) CLEF TEL-BL, and (c) CLEF GIRT-4.

A.2 Number of Relevant Documents



(a) TREC45-CR

(b) CLEF TEL-BL



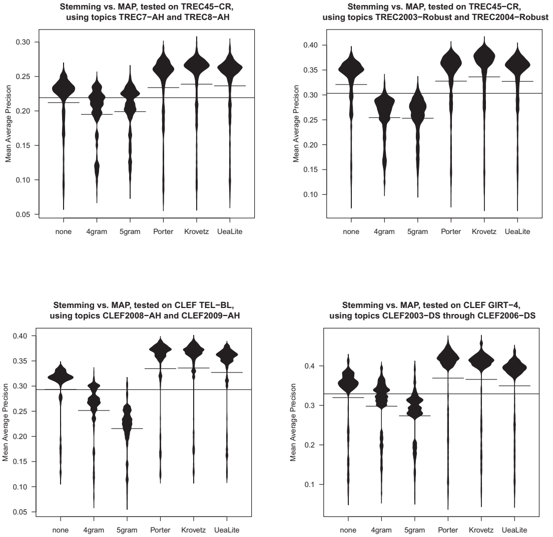
(c) CLEF GIRT-4

Figure A.2: Number of relevant documents for individual topics (order by decreasing number of relevant documents) for original topic sets from TREC and CLEF: (a) TREC45-CR, (b) CLEF TEL-BL, and (c) CLEF GIRT-4.

B Exploratory Result Analysis

B.1 Complete Experiment Set

B.1.1 Comparison of Stemmers



Stemming vs. MAP, tested on TREC45−CR, using topics TREC7−AH and TREC8−AH

Stemming vs. MAP, tested on TREC45−CR, using topics TREC2003−Robust and TREC2004−Robust

Stemming vs. MAP, tested on CLEF TEL−BL, using topics CLEF2008−AH and CLEF2009−AH

Stemming vs. MAP, tested on CLEF GIRT−4, using topics CLEF2003−DS through CLEF2006−DS

Figure B.1: Beanplot [92] visualisation of the distributions of MAP values for the *entire experiment set* (n=14,274) using *stemming implementations* as factors. Each figure illustrates the data for one of the merged topic sets created for evaluation.

B.1.2 Comparison of Ranking Models

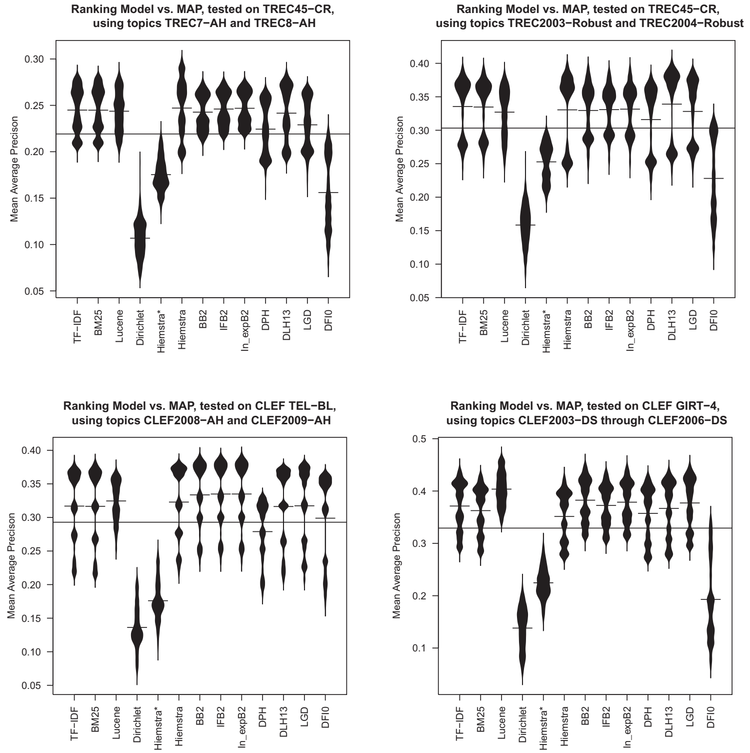


Figure B.2: Beanplot [92] visualisation of the distributions of MAP values for the *entire experiment set* (n=14,274) using *ranking models* as factors. Each figure illustrates the data for one of the merged topic sets created for evaluation.

B.1.3 Comparison of PRF Models

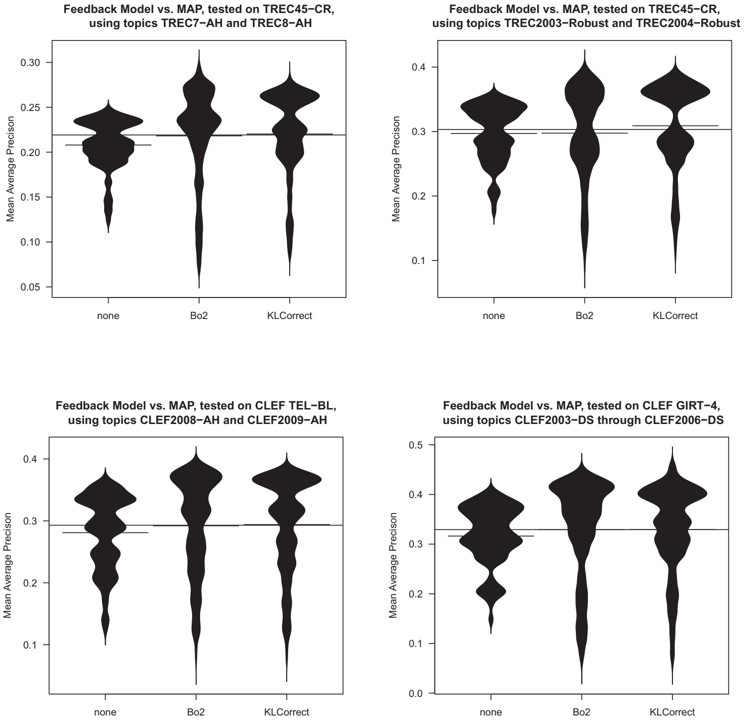


Figure B.3: Beanplot [92] visualisation of the distributions of MAP values for the *entire experiment set* (n=14,274) using *feedback models* as factors. Each figure illustrates the data for one of the merged topic sets created for evaluation.

B.1.4 Comparison of PRF Documents

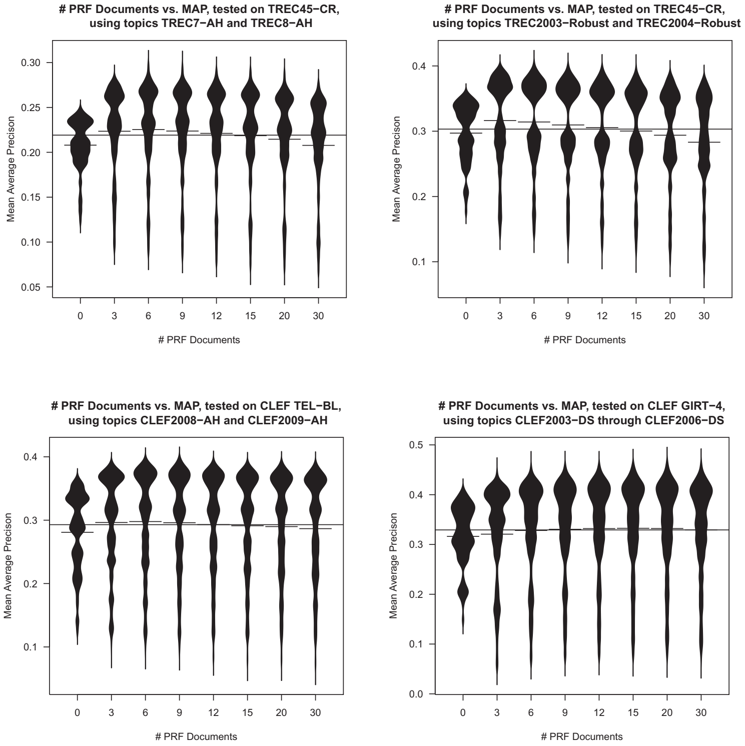


Figure B.4: Beanplot [92] visualisation of the distributions of MAP values for the *entire experiment set* (n=14,274) using *the number of PRF documents* as factors. Each figure illustrates the data for one of the merged topic sets created for evaluation.

B.1.5 Comparison of PRF Terms

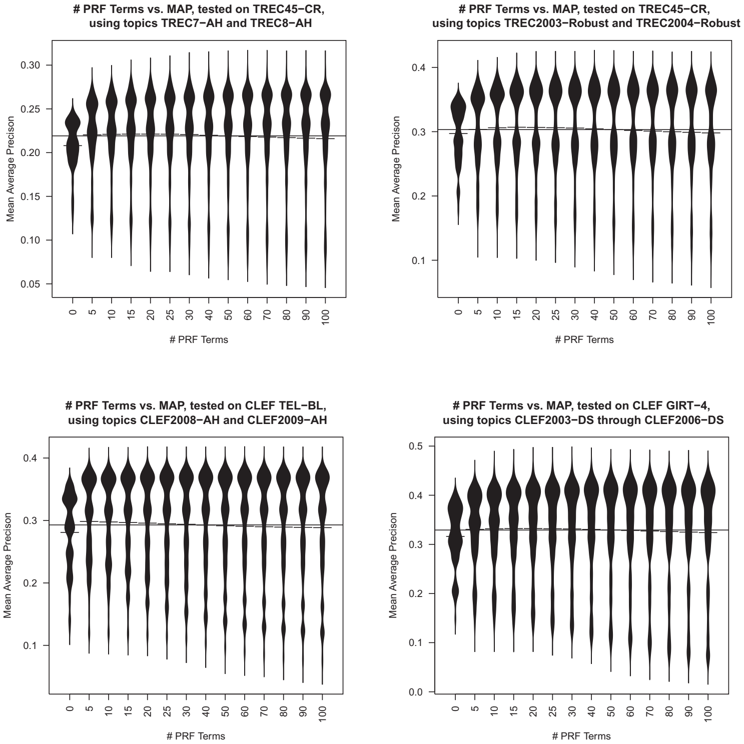
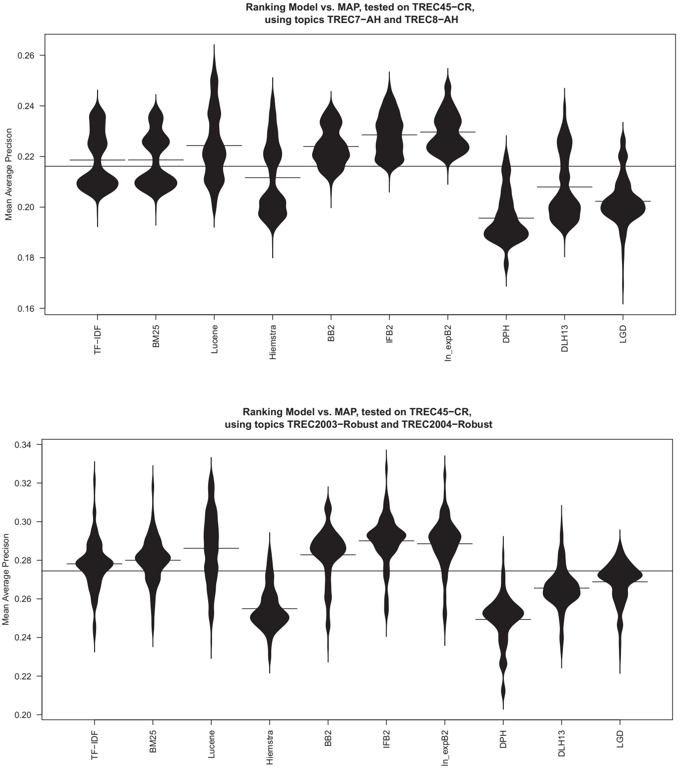


Figure B.5: Beanplot [92] visualisation of the distributions of MAP values for the *entire experiment set* (n=14,274) using *the number of PRF terms* as factors. Each figure illustrates the data for one of the merged topic sets created for evaluation.

B.2 N-gram Stemming Subset

B.2.1 Comparison of Ranking Models (TREC collections)

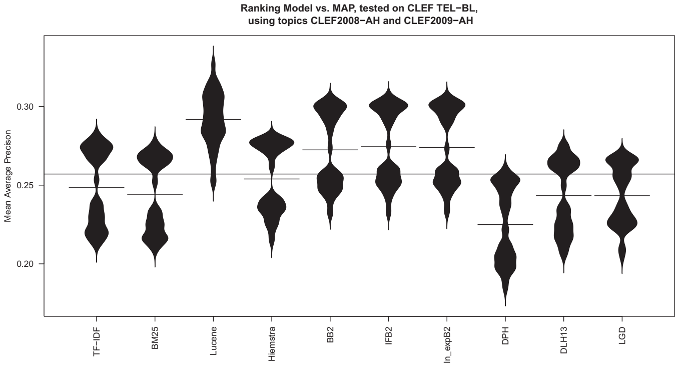


Ranking Model vs. MAP, tested on TREC45−CR, using topics TREC7−AH and TREC8−AH

Ranking Model vs. MAP, tested on TREC45−CR, using topics TREC2003−Robust and TREC2004−Robust

Figure B.6: Beanplot [92] visualisation of the distributions of MAP values for the *n-gram experiment subset* (n=3,660) using *ranking models* as factors.

B.2.2 Comparison of Ranking Models (CLEF collections)



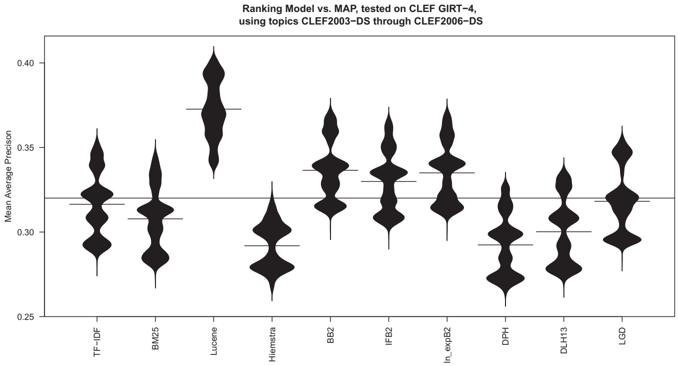


Figure B.7: Beanplot [92] visualisation of the distributions of MAP values for the *n-gram experiment subset* (n=3,660) using *ranking models* as factors.

B.2.3 Comparison of PRF Models

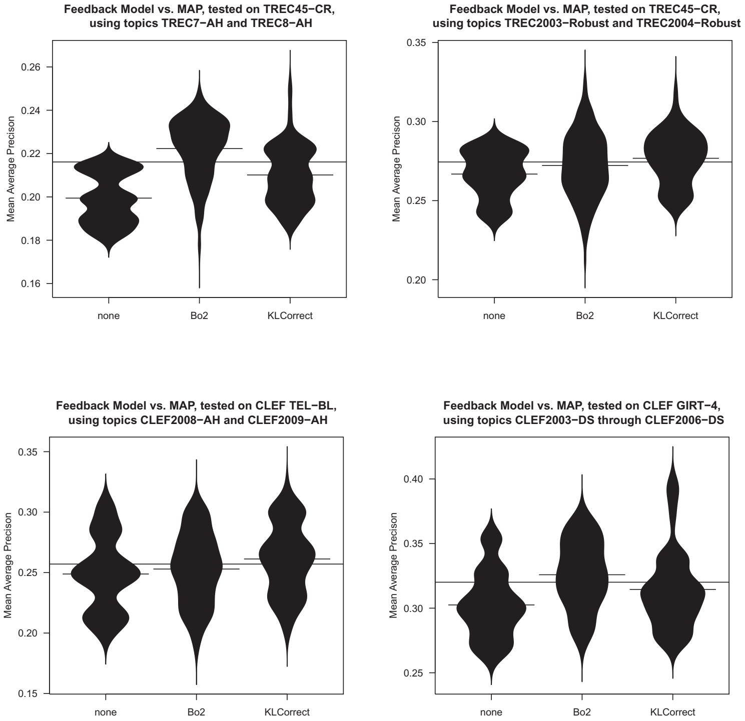
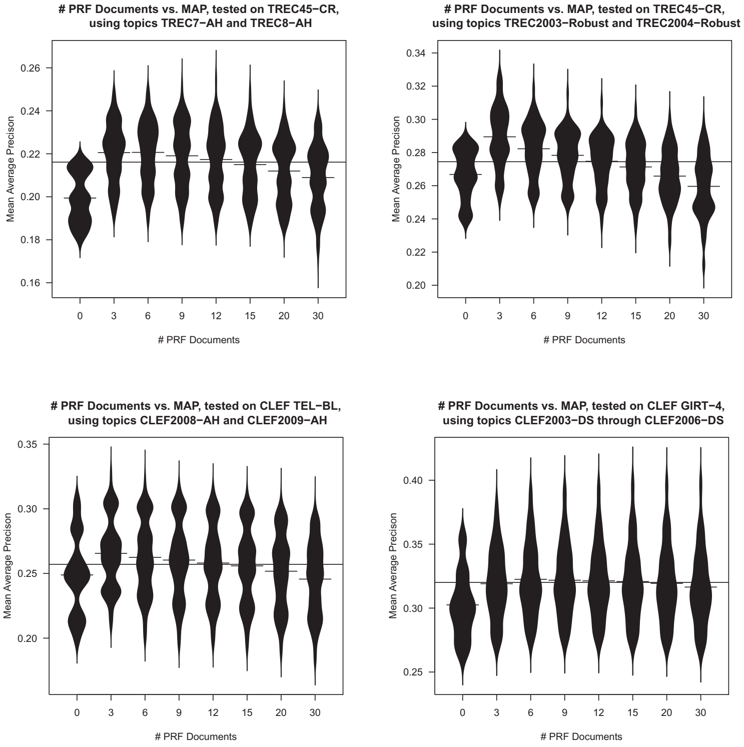


Figure B.8: Beanplot [92] visualisation of the distributions of MAP values for the *n-gram experiment subset* (n=3,660) using *PRF models* as factors.

B.2.4 Comparison of PRF Documents


# PRF Documents vs. MAP, tested on TREC45−CR, using topics TREC7−AH and TREC8−AH

# PRF Documents vs. MAP, tested on TREC45−CR, using topics TREC2003−Robust and TREC2004−Robust

# PRF Documents vs. MAP, tested on CLEF TEL−BL, using topics CLEF2008−AH and CLEF2009−AH

# PRF Documents vs. MAP, tested on CLEF GIRT−4, using topics CLEF2003−DS through CLEF2006−DS

Figure B.9: Beanplot [92] visualisation of the distributions of MAP values for the *n-gram experiment subset* (n=3,660) using *the number of PRF documents* as factors.

B.2.5 Comparison of PRF Terms

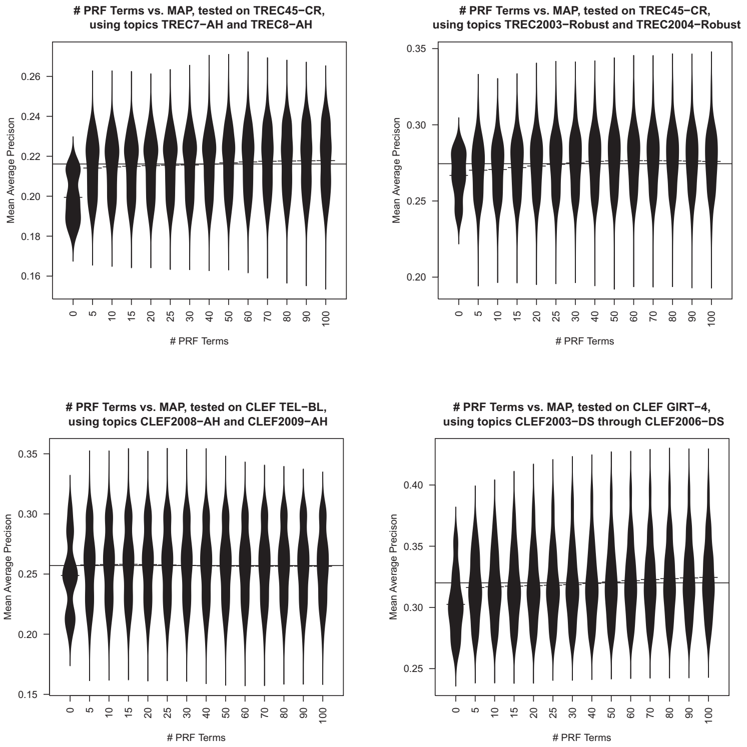


Figure B.10: Beanplot [92] visualisation of the distributions of MAP values for the *n-gram experiment subset* (n=3,660) using *the number of PRF terms* as factors.

C Optimal System Configurations per Test Collection

C.1 TREC45-CR Ad-hoc

Stemmer	Ranking	PRF (d, t)	MAP	Stemmer	Ranking	PRF (d, t)	MAP
4gram	TF-IDF	Bo2(12, 100)	0.2406	5gram	TF-IDF	Bo2(6, 100)	0.2393
	BM25	Bo2(9, 90)	0.2386		BM25	Bo2(6, 100)	0.2388
	Lucene	KLCorr(3, 40)	0.2304		**Lucene**	**KLCorr(12, 60)**	**0.2585**
	Hiemstra	Bo2(6, 100)	0.2425		Hiemstra	Bo2(15, 100)	0.2454
	BB2	Bo2(3, 15)	0.2400		BB2	Bo2(3, 50)	0.2389
	IFB2	Bo2(9, 60)	0.2435		IFB2	Bo2(6, 80)	0.2478
	In_expB2	**Bo2(3, 10)**	**0.2489**		In_expB2	Bo2(6, 100)	0.2491
	DPH	Bo2(3, 100)	0.2204		DPH	Bo2(6, 100)	0.2226
	DLH13	Bo2(6, 100)	0.2405		DLH13	Bo2(6, 100)	0.2413
	LGD	Bo2(3, 60)	0.2278		LGD	Bo2(3, 40)	0.2208
Porter	TF-IDF	Bo2(3, 40)	0.2813	Krovetz	TF-IDF	Bo2(6, 25)	0.2839
	BM25	Bo2(3, 25)	0.2831		BM25	Bo2(6, 30)	0.2818
	Lucene	KLCorr(3, 15)	0.2835		Lucene	KLCorr(9, 20)	0.2872
	Hiemstra	**Bo2(6, 100)**	**0.2939**		***Hiemstra***	***Bo2(6, 60)***	***0.3001***
	BB2	Bo2(3, 25)	0.2760		BB2	Bo2(3, 30)	0.2769
	IFB2	Bo2(3, 25)	0.2767		IFB2	Bo2(6, 50)	0.2765
	In_expB2	Bo2(3, 25)	0.2770		In_expB2	Bo2(3, 30)	0.2762
	DPH	Bo2(3, 25)	0.2635		DPH	Bo2(3, 30)	0.2707
	DLH13	Bo2(6, 30)	0.2857		DLH13	Bo2(6, 40)	0.2869
	LGD	KLCorr(6, 100)	0.2676		LGD	KLCorr(6, 100)	0.2715
UeaLite	TF-IDF	Bo2(9, 40)	0.2812				
	BM25	Bo2(12, 30)	0.2812				
	Lucene	KLCorr(15, 25)	0.2765				
	Hiemstra	**Bo2(6, 80)**	**0.2973**				
	BB2	KLCorr(12, 100)	0.2678				
	IFB2	Bo2(9, 40)	0.2746				
	In_expB2	Bo2(9, 40)	0.2732				
	DPH	Bo2(3, 25)	0.2648				
	DLH13	Bo2(6, 30)	0.2847				
	LGD	KLCorr(6, 100)	0.2714				

Table C.1: Complete system configurations for each system component resulting in optimal retrieval performance on the TREC45-CR Ad-hoc test collection.

C.2 TREC45-CR Robust

Stemmer	Ranking	PRF (d, t)	MAP	Stemmer	Ranking	PRF (d, t)	MAP
4gram	TF-IDF	Bo2(3, 100)	0.3239	5gram	TF-IDF	Bo2(3, 100)	0.2974
	BM25	Bo2(3, 100)	0.3217		BM25	Bo2(6, 90)	0.2970
	Lucene	KLCorr(3, 50)	0.3095		**Lucene**	**KLCorr(3, 50)**	**0.3259**
	Hiemstra	Bo2(3, 100)	0.2802		Hiemstra	Bo2(6, 100)	0.2870
	BB2	Bo2(3, 40)	0.3108		BB2	Bo2(3, 5)	0.3103
	IFB2	**Bo2(3, 100)**	**0.3299**		IFB2	Bo2(3, 5)	0.3151
	In_expB2	Bo2(3, 100)	0.3268		In_expB2	Bo2(3, 5)	0.3078
	DPH	Bo2(3, 100)	0.2851		DPH	Bo2(3, 100)	0.2748
	DLH13	Bo2(3, 100)	0.3012		DLH13	Bo2(3, 70)	0.2852
	LGD	Bo2(3, 40)	0.2885		LGD	Bo2(3, 40)	0.2847
Porter	TF-IDF	Bo2(6, 25)	0.3866	Krovetz	TF-IDF	Bo2(3, 40)	0.3951
	BM25	Bo2(6, 20)	0.3822		BM25	Bo2(6, 15)	0.3939
	Lucene	KLCorr(6, 20)	0.3786		Lucene	KLCorr(9, 40)	0.3873
	Hiemstra	Bo2(12, 100)	0.3888		Hiemstra	Bo2(12, 100)	0.3992
	BB2	Bo2(3, 30)	0.3818		BB2	Bo2(3, 30)	0.3954
	IFB2	Bo2(3, 50)	0.3819		IFB2	Bo2(3, 40)	0.3901
	In_expB2	Bo2(3, 40)	0.3852		In_expB2	Bo2(3, 40)	0.3944
	DPH	KLCorr(3, 100)	0.3742		DPH	Bo2(3, 40)	0.3836
	DLH13	**Bo2(6, 20)**	**0.3960**		***DLH13***	***Bo2(6, 40)***	***0.4056***
	LGD	KLCorr(3, 90)	0.3851		LGD	KLCorr(3, 60)	0.3924
UeaLite	TF-IDF	Bo2(6, 20)	0.3786				
	BM25	Bo2(6, 20)	0.3801				
	Lucene	KLCorr(6, 40)	0.3776				
	Hiemstra	**Bo2(15, 70)**	**0.3906**				
	BB2	KLCorr(3, 50)	0.3765				
	IFB2	Bo2(3, 40)	0.3677				
	In_expB2	Bo2(3, 90)	0.3678				
	DPH	Bo2(3, 30)	0.3711				
	DLH13	Bo2(6, 40)	0.3934				
	LGD	KLCorr(3, 100)	0.3880				

Table C.2: Complete system configurations for each system component resulting in optimal retrieval performance on the TREC45-CR Robust test collection.

C.3 CLEF TEL-BL

Stemmer	Ranking	PRF (d, t)	MAP	Stemmer	Ranking	PRF (d, t)	MAP
4gram	TF-IDF	Bo2(6, 20)	0.2827	5gram	TF-IDF	Bo2(3, 20)	0.2404
	BM25	Bo2(6, 20)	0.2779		BM25	Bo2(3, 25)	0.2354
	Lucene	KLCorr(3, 25)	0.3291		**Lucene**	**KLCorr(3, 30)**	**0.3099**
	Hiemstra	Bo2(12, 100)	0.2814		Hiemstra	KLCorr(6, 20)	0.2425
	BB2	Bo2(3, 10)	0.3056		BB2	KLCorr(9, 50)	0.2593
	IFB2	**Bo2(3, 10)**	**0.3065**		IFB2	KLCorr(9, 90)	0.2618
	In_expB2	Bo2(3, 10)	0.3064		In_expB2	KLCorr(3, 15)	0.2606
	DPH	KLCorr(3, 20)	0.2602		DPH	KLCorr(3, 25)	0.2213
	DLH13	Bo2(3, 10)	0.2759		DLH13	KLCorr(3, 25)	0.2380
	LGD	KLCorr(3, 25)	0.2704		LGD	KLCorr(3, 5)	0.2393
Porter	TF-IDF	Bo2(6, 90)	0.3811	Krovetz	TF-IDF	Bo2(6, 10)	0.3747
	BM25	Bo2(6, 25)	0.3828		BM25	Bo2(6, 20)	0.3755
	Lucene	KLCorr(9, 25)	0.3745		Lucene	KLCorr(9, 10)	0.3652
	Hiemstra	Bo2(6, 5)	0.3819		Hiemstra	Bo2(9, 25)	0.3835
	BB2	**Bo2(6, 5)**	**0.3911**		BB2	Bo2(9, 25)	0.3926
	IFB2	Bo2(6, 5)	0.3903		IFB2	Bo2(9, 30)	0.3917
	In_expB2	Bo2(6, 5)	0.3904		***In_expB2***	***Bo2(9, 30)***	***0.3934***
	DPH	KLCorr(6, 70)	0.3264		DPH	KLCorr(9, 60)	0.3322
	DLH13	KLCorr(9, 100)	0.3752		DLH13	Bo2(6, 5)	0.3757
	LGD	KLCorr(6, 25)	0.3811		LGD	KLCorr(20, 15)	0.3806
UeaLite	TF-IDF	Bo2(3, 15)	0.3697				
	BM25	Bo2(3, 15)	0.3691				
	Lucene	KLCorr(9, 30)	0.3696				
	Hiemstra	Bo2(6, 15)	0.3735				
	BB2	Bo2(3, 5)	0.3813				
	IFB2	Bo2(6, 25)	0.3813				
	In_expB2	**Bo2(6, 25)**	**0.3815**				
	DPH	KLCorr(3, 50)	0.3202				
	DLH13	Bo2(3, 15)	0.3644				
	LGD	KLCorr(3, 20)	0.3732				

Table C.3: Complete system configurations for each system component resulting in optimal retrieval performance on the CLEF TEL-BL test collection.

C.4 CLEF GIRT-4

Stemmer	Ranking	PRF (d, t)	MAP	Stemmer	Ranking	PRF (d, t)	MAP
4gram	TF-IDF	Bo2(6, 90)	0.3538	5gram	TF-IDF	Bo2(9, 100)	0.3209
	BM25	Bo2(9, 100)	0.3474		BM25	Bo2(6, 100)	0.3180
	Lucene	**KLCorr(20, 50)**	**0.3994**		**Lucene**	**KLCorr(15, 80)**	**0.4024**
	Hiemstra	Bo2(6, 100)	0.3225		Hiemstra	Bo2(6, 100)	0.2946
	BB2	Bo2(6, 100)	0.3717		BB2	Bo2(6, 100)	0.3427
	IFB2	Bo2(20, 100)	0.3666		IFB2	Bo2(6, 100)	0.3387
	In_expB2	Bo2(15, 100)	0.3714		In_expB2	Bo2(6, 100)	0.3431
	DPH	Bo2(12, 100)	0.3279		DPH	Bo2(12, 90)	0.2968
	DLH13	Bo2(15, 100)	0.3366		DLH13	Bo2(6, 100)	0.3011
	LGD	Bo2(9, 90)	0.3553		LGD	Bo2(9, 100)	0.3195
Porter	TF-IDF	Bo2(20, 100)	0.4449	Krovetz	TF-IDF	Bo2(20, 90)	0.4333
	BM25	Bo2(20, 100)	0.4320		BM25	Bo2(20, 100)	0.4253
	Lucene	**KLCorr(20, 25)**	**0.4663**		***Lucene***	***KLCorr(20, 30)***	***0.4672***
	Hiemstra	Bo2(20, 100)	0.4291		Hiemstra	Bo2(30, 100)	0.4217
	BB2	Bo2(20, 100)	0.4535		BB2	Bo2(20, 100)	0.4420
	IFB2	Bo2(20, 100)	0.4397		IFB2	Bo2(20, 100)	0.4302
	In_expB2	Bo2(20, 100)	0.4505		In_expB2	Bo2(20, 100)	0.4377
	DPH	Bo2(9, 50)	0.4316		DPH	Bo2(15, 50)	0.4223
	DLH13	Bo2(15, 70)	0.4419		DLH13	Bo2(20, 100)	0.4323
	LGD	Bo2(15, 60)	0.4516		LGD	Bo2(12, 50)	0.4317
UeaLite	TF-IDF	Bo2(20, 100)	0.4202				
	BM25	Bo2(20, 100)	0.4138				
	Lucene	**KLCorr(20, 50)**	**0.4421**				
	Hiemstra	Bo2(20, 100)	0.4076				
	BB2	Bo2(15, 90)	0.4277				
	IFB2	Bo2(20, 100)	0.4209				
	In_expB2	Bo2(20, 100)	0.4249				
	DPH	Bo2(15, 40)	0.4075				
	DLH13	Bo2(20, 80)	0.4174				
	LGD	Bo2(20, 60)	0.4178				

Table C.4: Complete system configurations for each system component resulting in optimal retrieval performance on the CLEF GIRT-4 test collection.

A Generic Approach to Component-Level Evaluation in Information Retrieval

Research in information retrieval deals with the theories and models that constitute the foundations for any kind of service that provides access or pointers to particular elements of a collection of documents in response to a submitted information need. The specific field of information retrieval evaluation is concerned with the critical assessment of the quality of search systems. Empirical evaluation based on the Cranfield paradigm using a specific collection of test queries in combination with relevance assessments in a laboratory environment is the classic approach to compare the impact of retrieval systems and their underlying models on retrieval effectiveness.

In the past two decades international campaigns, like the Text Retrieval Conference, have led to huge advances in the design of experimental information retrieval evaluations. But in general the focus of this system-driven paradigm remained on the comparison of system results, i.e. retrieval systems are treated as black boxes. This approach to the evaluation of retrieval system has been criticised for treating systems as black boxes. Recent works on this subject have proposed the study of the system configurations and their individual components. This thesis proposes a generic approach to the evaluation of retrieval systems at the component-level.

The focus of the thesis at hand is on the key components that are needed to address typical ad-hoc search tasks, like finding books on a particular topic in a large set of library records. A central approach in this work is the further development of the Xtrieval framework by the integration of widely-used IR toolkits in order to eliminate the limitations of individual tools. Strong empirical results at international campaigns that provided various types of evaluation tasks confirm both the validity of this approach and the flexibility of the Xtrieval framework.

Modern information retrieval systems contain various components that are important for solving particular subtasks of the retrieval process. This thesis illustrates the detailed analysis of important system components needed to address ad-hoc retrieval tasks. Here, the design and implementation of the Xtrieval framework offers a variety of approaches for flexible system configurations. Xtrieval has been designed as an open system and allows the integration of further components and tools as well as addressing search tasks other than ad-hoc retrieval. This approach ensures that it is possible to conduct automated component-level evaluation of retrieval approaches.

Both the scale and impact of these possibilities for the evaluation of retrieval systems are demonstrated by the design of an empirical experiment that covers more than 13,000 individual system configurations. This experimental set-up is tested on four test collections for ad-hoc search. The results of this experiment are manifold. For instance, particular implementations of ranking models fail systematically on all tested collections. The exploratory analysis of the ranking models empirically confirms the relationships between different implementations of models that share theoretical foundations. The obtained results also suggest that the impact on retrieval effectiveness of most instances of IR system components depends on the test collections that are being used for evaluation. Due to the scale of the designed component-level evaluation experiment, not all possible interactions of the system component under examination could be analysed in this work. For this reason the resulting data set will be made publicly available to the entire research community.

Informatik

21,00 €

ISSN: 2195-2574 **ISBN:** 978-3-941003-68-2

Universitätsverlag Chemnitz