



TECHNISCHE UNIVERSITÄT  
CHEMNITZ

FAKULTÄT FÜR INFORMATIK  
PROFESSUR INFORMATIONSSYSTEME UND SOFTWARETECHNOLOGIE

Diplomarbeit

# Nachrichtenklassifikation als Komponente in WEBIS

Björn Krellner

<bjk@informatik.tu-chemnitz.de>

25.09.2006

**Prüfer:** Prof. Dr.-Ing. Petr Kroha

**Betreuer:** Prof. Dr.-Ing. Petr Kroha

# AUFGABENSTELLUNG FÜR DIE DIPLOMARBEIT

*Name, Vorname des Diplomanden:* Krellner, Björn

*Immatrikulations-Nr.:* 22306

***Thema:***  
**Nachrichtenklassifikation als Komponente in WEBIS**

***Zielstellung:***

In dieser Arbeit soll der in der Studienarbeit „Nachrichtenklassifikation unter Nutzung regulärer Ausdrücke“ entwickelte Prototyp weiterentwickelt und mit dem Web-orientierten Informationssystem (WEBIS) verschmolzen werden.

Das System soll in der Lage sein, nachdem eine Reihe von gleichsprachigen Eingabetexten (Börsenmeldungen) Klassen (z. B. Up oder Down) zugeordnet wurden, weitere Texte zu klassifizieren. Das zu entwickelnde WEBIS-Plugin soll im Gegensatz zum Prototyp durch eine intuitive Bedienführung entsprechend nutzerfreundlich sein.

Weiterhin soll es möglich sein, auch weitere Klassifikationsalgorithmen auszuwählen und deren Parameter bestimmen zu können.

Die Diplomarbeit wird enthalten:

1. Beschreibung Ist-Zustand des WEBIS-Systems
2. Erstellung des Textklassifikationsplugins für WEBIS, welches im Batchbetrieb englischsprachige und deutschsprachige Börsennachrichten verarbeiten und klassifizieren kann
3. Erstellung eines separaten Programms mit der genannten Funktionalität
4. Vergleiche mit den Klassifikationsergebnissen durch Grammatikalische Analyse von Thomas Reichel und mit den Ergebnissen, die mit dem Prototyp erreicht wurden

***Betreuender Hochschullehrer:*** Prof. Dr. P. Kroha  
***Fakultät:*** Fakultät für Informatik  
***Professur:*** Informationssysteme u. Softwaretechnik  
***Betreuer:*** Prof. Dr. P. Kroha  
***Beginn am:*** 01.04.2006  
***Einzureichen am:*** 30.09.2006

### *Selbständigkeitserklärung*

Hiermit erkläre ich, die vorliegende Arbeit selbstständig und nur unter Zuhilfenahme der angegebenen Quellen angefertigt habe.

Diese Diplomarbeit wurde noch keiner Prüfungsbehörde in dieser oder anderer Form vorgelegt.

Chemnitz, 25.09.2006

Björn Krellner

# Inhaltsverzeichnis

<b>Abbildungsverzeichnis</b>	<b>6</b>
<b>Tabellenverzeichnis</b>	<b>8</b>
<b>Listings</b>	<b>9</b>
<b>1 Einleitung</b>	<b>10</b>
1.1 Börse und Internet . . . . .	10
1.2 Aufbau dieser Arbeit . . . . .	10
<b>2 WEBIS</b>	<b>12</b>
2.1 Erste Version 2001 . . . . .	12
2.2 Ursprünglichste Nutzerschnittstelle . . . . .	13
2.3 Umwandlung in Client/Server-Anwendung . . . . .	14
2.4 Spezialisierung auf die Börse . . . . .	15
2.5 Plug-in-Architektur . . . . .	16
2.6 Weitere Entwicklungen . . . . .	18
<b>3 Softwareentwicklung</b>	<b>19</b>
3.1 Entwicklung vom Prototyp zur GUI-basierten Software . . . . .	19
3.1.1 Vor- und Nachteile von Prototypen . . . . .	19
3.1.2 Formen von Prototypen . . . . .	20
3.2 Softwareübersicht . . . . .	20
3.3 Feature Reduction . . . . .	22
3.4 Wählbare Classifier . . . . .	25
3.4.1 Naive Bayes . . . . .	25
3.4.2 Maximum Entropy . . . . .	26
3.4.3 Balanced Winnow . . . . .	27
3.4.4 Decision Tree . . . . .	28
3.5 Beispielnutzung des Systems . . . . .	29
3.5.1 Konfigurationserstellung / komplette Klassifikation . . . . .	29
3.5.2 Testen neuer Daten . . . . .	34
3.6 WEBIS-Server-Plugin . . . . .	38
3.7 WEBIS-Client-Plug-in . . . . .	39

3.7.1 RemoteFileDialog . . . . .	40
3.7.2 Look and Feel als WEBIS-Client-Plug-in . . . . .	41
<b>4 Klassifikationsergebnisse</b>	<b>43</b>
4.1 Nachrichtenklassen und Untersuchungsvorhaben . . . . .	43
<b>5 Vergleich mit bisherigen Ergebnissen</b>	<b>55</b>
5.1 Vergleich mit [Rei06] . . . . .	55
5.2 Vergleich mit [Kre06] . . . . .	58
<b>6 Rückblick und Ausblick</b>	<b>61</b>
<b>Abkürzungsverzeichnis</b>	<b>63</b>
<b>Literaturverzeichnis</b>	<b>64</b>

# Abbildungsverzeichnis

2.1	WEBIS-GUI im Jahr 2001 (siehe [Gem01]) . . . . .	13
2.2	WEBIS-GUI im Jahr 2002 (siehe [Löf02]) . . . . .	14
2.3	WEBIS-Client/Server-Architektur (siehe [Ahn05]) . . . . .	15
2.4	WEBIS-Plug-in-Architektur (siehe [Mei04]) . . . . .	16
2.5	WEBIS-GUI im Jahr 2005 (siehe [Mei05]) . . . . .	17
3.1	Prinzipieller Bearbeitungsablauf . . . . .	21
3.2	Softwareübersicht: WEBIS-Plug-ins und separates Programm . . . . .	22
3.3	Trend Forecast Engine: Auswählen der Eingabedatei . . . . .	30
3.4	Trend Forecast Engine: Treffen der Vorverarbeitungsoptionen . . . . .	30
3.5	Trend Forecast Engine: Vorverarbeitung läuft, das obere Panel ist inaktiv . . . . .	31
3.6	Trend Forecast Engine: Klassen einstellen, Testanteil bestimmen . . . . .	32
3.7	Trend Forecast Engine: Dialog zum Erstellen der Feature-Vektoren . . . . .	33
3.8	Trend Forecast Engine: Klassifikationseinstellfenster . . . . .	34
3.9	Trend Forecast Engine: Konfiguration abspeichern nach Klassifikation . . . . .	35
3.10	Trend Forecast Engine: Klassifikationsergebnis anschauen . . . . .	35
3.11	Trend Forecast Engine: Ansichtswechsel . . . . .	36
3.12	Trend Forecast Engine: Vorverarbeitung der neuen Testklassen aktiv . . . . .	37
3.13	Trend Forecast Engine: Auswahl der Kursdaten . . . . .	37
3.14	Trend Forecast Engine: Zoomansicht bringt Erkenntnis . . . . .	38
3.15	Datei öffnen / Verzeichnis öffnen . . . . .	40
3.16	Trend Forecast Engine im WEBIS: Logger für Statusmeldungen . . . . .	41
3.17	Trend Forecast Engine im WEBIS: Öffnen einer CSV-Datei . . . . .	42
3.18	Trend Forecast Engine im WEBIS: Grafische Auswertung . . . . .	42
4.1	Definierte Phasen und DAX-Kursverläufe für Testzeitraum . . . . .	45
4.2	Alle Naive-Bayes-Klassifikationen mit um 75 % reduzierten Feature Vectors . . . . .	48
4.3	Alle Naive-Bayes-Klassifikationen mit um 50 % reduzierten Feature Vectors . . . . .	49
4.4	Alle Naive-Bayes-Klassifikationen mit um 25 % reduzierten Feature Vectors . . . . .	50
4.5	Alle Naive-Bayes-Klassifikationen mit unreduzierten Testvektoren . . . . .	51
4.6	Alle Naive-Bayes-Klassifikationen mit gleicher Reduktion . . . . .	52
4.7	Alle Naive-Bayes-Klassifikationen mit um 25 % reduzierten . . . . .	53
4.8	Klassifikationsergebnisse von 4 verschiedenen Classifiern . . . . .	54

5.1	Manuell festgelegte Nachrichtenklassen . . . . .	56
5.2	Ungeglätteter und geglätteter Naive-Bayes-Klassifikationsverlauf . . . . .	57
5.3	Klassifikationsvergleich mit den Ergebnissen aus [Rei06] . . . . .	59
5.4	Klassifikationsvergleich mit den Ergebnissen aus [Kre06] . . . . .	60

# Tabellenverzeichnis

- 4.1 Auf bestimmte Kategorien reduzierte Texteingabedaten . . . . . 44
- 4.2 Definierte Up- und Downphasen für den Testzeitraum . . . . . 44

# Listings

3.1	Codefragment für Feature Reduction . . . . .	22
3.2	Manifestdatei <code>plugin_server.xml</code> . . . . .	38
3.3	Manifestdatei <code>plugin_client.xml</code> . . . . .	39
5.1	<code>2001.12.14-00.00.00-01.txt</code> . . . . .	55
5.2	<code>2001.12.14-00.00.00-11.txt</code> . . . . .	55
5.3	<code>2003.05.27-00.00.00-01.txt</code> . . . . .	55
5.4	<code>2003.05.27-00.00.00-07.txt</code> . . . . .	58
5.5	<code>2005.11.02-03.05.00-00.txt</code> . . . . .	58
5.6	<code>2005.11.02-03.05.00-08.txt</code> . . . . .	58

# 1 Einleitung

## 1.1 Börse und Internet

Um ihre Kauf- und Verkaufsempfehlungen auf dem Aktienparkett geben zu können, stützen sich Fondsmanager und Finanzexperten auf Meldungen und Nachrichten aus dem Umfeld von Wirtschaft und Börse. Diese Informationen sind jedoch auch für Kleinaktiönäre interessant, um zu entscheiden, ob sie ihre Aktien verkaufen sollen oder in welche Aktien es sich lohnt zu investieren. Bei den besagten Nachrichten handelt es sich zum Einen um Meldungen zu bestimmten Firmen oder speziellen Wirtschaftsbereichen (z. B. Maschinenbausektor, IT-Branche, ...) und zum Anderen um Neuigkeiten, die die gesamte Wirtschaft und die Finanzstabilität im Land (z. B. Zinsänderungsankündigungen, Wirtschaftsgutachten, ...) betreffen.

Durch die immer weiter steigende Popularität des Internets nimmt auch der Strom solcher Nachrichten über das Medium Internet zu. Diese Nachrichten können u. a. in Form von Mails, die bei bestimmten Dienstleistern abonniert werden können, oder in Form von Meldungen auf speziellen Webseiten zur Verfügung gestellt sein. Die Menge dieser digital verfügbaren Nachrichten bringt es natürlich mit sich, dass es nahezu unmöglich ist, all diese Nachrichten in kürzester Zeit zu analysieren und auf diese Meldungen zeitnah und angemessen reagieren zu können. Die digitale Verfügbarkeit dieser Nachrichten bietet jedoch die Chance, die Auswertung mit Hilfe von Computersystemen zu unterstützen und damit die Flut der Daten zu bewältigen.

## 1.2 Aufbau dieser Arbeit

In [Kre05] werden mögliche Herangehensweisen vorgestellt und diskutiert, wie man automatisiert Erkenntnisse zum allgemeinen Trend der Aktienkursentwicklung (speziell am Verlauf des Leitindex, in Deutschland also dem DAX), gewinnen kann. In [Kre06] wird ein Prototyp entwickelt und ausprobiert. Der Prototyp besteht aus nacheinander zu startenden Shellskripten, die wiederum fragmenthaft implementierte Java-Klassen aufrufen, und ist in der Lage, Nachrichtentexte zu klassifizieren (Vorhersage über die Wahrscheinlichkeit des Zugehörens zu Klasse *UP* für positiven Trend bzw. negiert zur Klasse *DOWN* für negativen Trend).

Im nächsten Kapitel wird die bisherige Entwicklung im Projekt WEBIS skizziert und die Integrationsmöglichkeiten von neuem Code genannt. Im 3. Kapitel wird auf die Entwicklungsdetails der separaten Software, die für diese Arbeit entwickelt wurde als auch auf die Integration dieser Software in WEBIS eingegangen. Im 4. Kapitel werden Klassifikationsergebnisse, die mit der neuen Software erhalten wurden, vorgestellt und diese im 5. Kapitel mit bisherigen Erkenntnissen verglichen. Im abschließenden Kapitel wird noch einmal kurz auf die Entwicklung der Software und der Methodik zurückgeblickt und ein Ausblick auf weitere Verbesserungs- und Erweiterungsmöglichkeiten gegeben.

## 2 WEBIS

In diesem Abschnitt soll das WEBIS-System und seine Entwicklung bis zum jetzigen Stand vorgestellt werden.

### 2.1 Erste Version 2001

Das Internet stellt mit seiner ständig wachsenden Anzahl an zumeist frei anschaulichen Informationen eine unüberschaubare Fülle an Daten zur Verfügung. Die größten Probleme, die sich bei der Nutzung dieser Menge an Daten stellen, sind zum einen die Verteilung der Daten über das gesamte Netz, die fehlende Möglichkeit, komplexe Anfragen zu stellen, und die Schwierigkeiten, die für Computersysteme bei der Verarbeitung dieser Daten auftreten. Der Grund für diese Schwierigkeiten liegt in der Tatsache, dass das zum Standard gewordene HTML für die Darstellung von WWW-Seiten für die Lesbarkeit durch Menschen optimiert wurde und daher nur sehr wenige computerlesbare Informationen über die semantische Struktur eines Dokumentes beinhaltet.

In [Gem01] wird das Informationssystem WEBIS „geboren“. Die fünf Buchstaben stehen für Web-orientiertes Informationssystem, und ein solches transformiert Daten in Informationen, im speziellen Fall zumeist Daten aus dem WWW. Die genaue Art der Daten und deren Herkunft sind ebenso wenig wie die Transformationsschritte und die Datenhaltung vorgegeben.

Technisch gesehen besteht die Aufgabe des WEBIS darin, sich an die Gegebenheiten des Webs (also an die Protokoll- und Datenformate) anzupassen und mittels der gewonnenen Strukturkenntnisse die benötigten Daten zu gewinnen. Dabei simuliert das System den menschlichen Nutzer, der durch das Web navigiert und dabei die für den Menschen konzipierten Web-Seiten interpretiert. Eine wichtige Aufgabe besteht also darin, sich auch an veränderte Strukturen anpassen zu können.

Die persistente Speicherung der Daten wurde im XML-Format vorgesehen und implementiert. Mit diesen Daten (und speziell mit Zeitreihen) können später Informationen transformiert werden. Unter Daten versteht man Angaben über verschiedenste Fakten oder Zusammenhänge, die objektiv wahrnehmbar und potentiell verwertbar sind. Daten

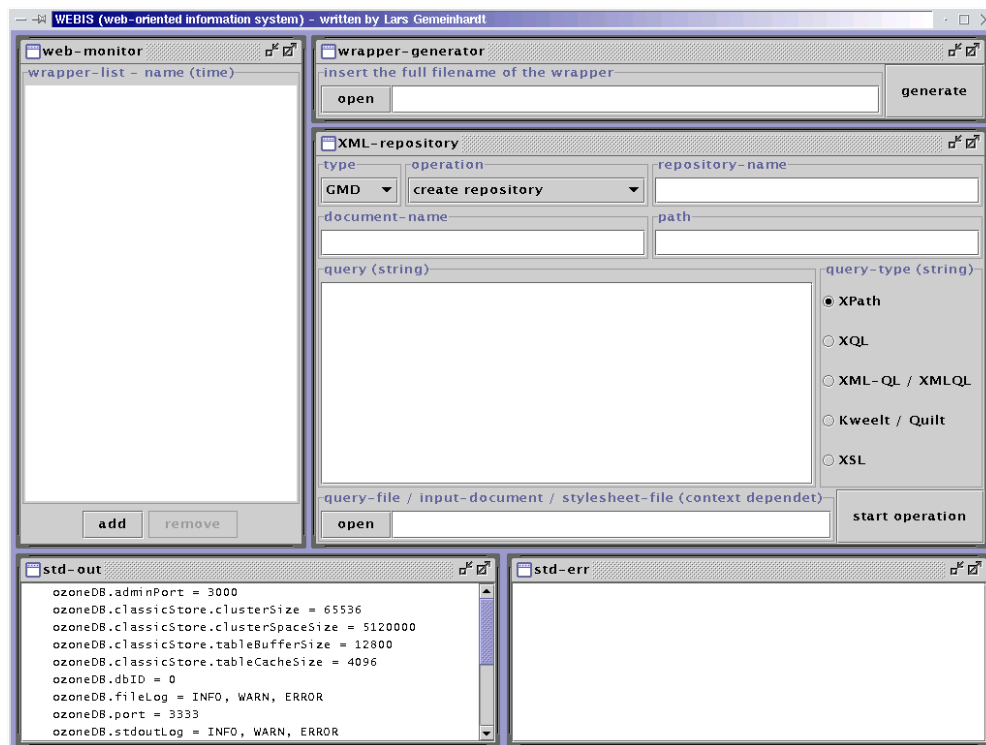


Abbildung 2.1: WEBIS-GUI im Jahr 2001 (siehe [Gem01])

fehlt jedoch im Vergleich zu Informationen die „Verwertbarkeit“, sie enthalten keine erkennbare Semantik. Ein einzelner Aktienkurswert ist demnach ein Datum, die Kenntnis des Kurses einer Firma zu bestimmten Zeitpunkten, könnte hingegen nützliche Informationen liefern.

## 2.2 Ursprünglichste Nutzerschnittstelle

Abbildung 2.1 zeigt die GUI des WEBIS in der ersten Version. Sie stellt die vom System gegebene Funktionalität dar. Folgende Komponenten (jeweils in einzelnen Fenstern) wurden verwendet:

- Web-Monitor
- Wrapper-Generator
- XML-Repository
- Standard-Ausgabe
- Standard-Fehlerausgabe

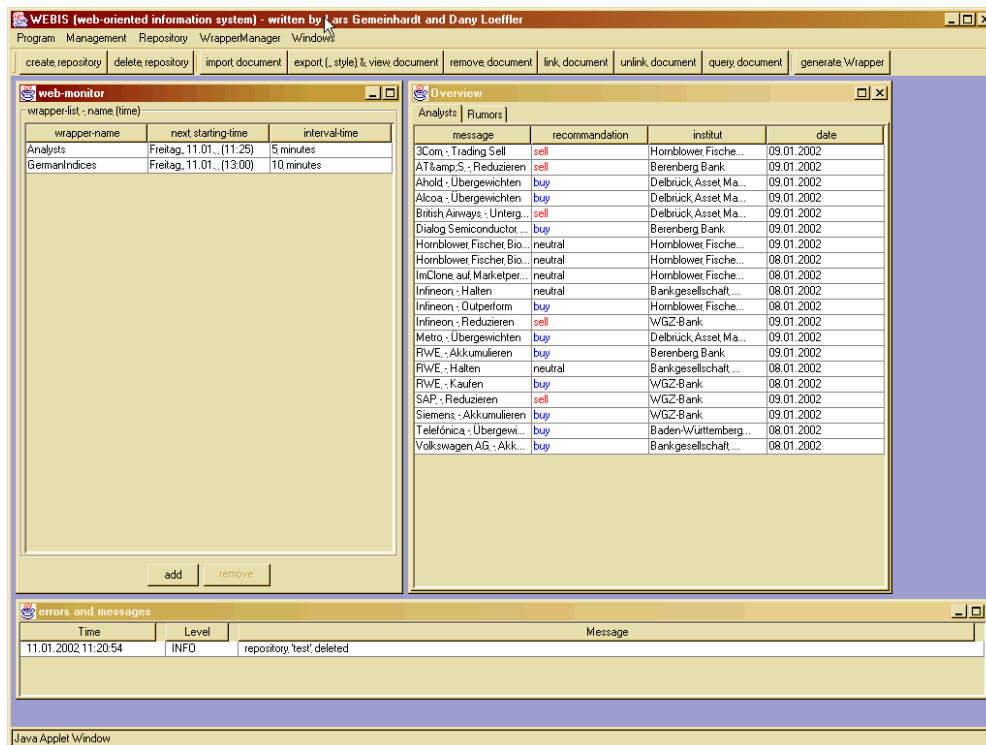


Abbildung 2.2: WEBIS-GUI im Jahr 2002 (siehe [Löf02])

## 2.3 Umwandlung in Client/Server-Anwendung

Das Client/Server-Schema, auf dem die Nutzung des WWW beruht, wird in [Löf02] auch für das Informationssystem aufgegriffen. Man konnte nun von verschiedenen Rechnern aus auf den WEBIS-Kern (Server) mit der XML-Datenhaltung zugreifen. Ebenso war nun eine Benutzerverwaltung integriert.

Abbildung 2.2 zeigt beispielhaft, wie auch die grafische Oberfläche angepasst worden ist.

Als Programmiersprache für die Entwicklung wurde von Anfang an Java verwendet. Die Client/Server-Architektur wurde mit dem RMI-Standard von Sun Microsystems implementiert. Der Client dient hauptsächlich noch als graphische Benutzerschnittstelle, was ja auch dem Client/Server-Prinzip entspricht. Der schematische Aufbau aus Benutzersicht ist in Abbildung 2.3 visualisiert.

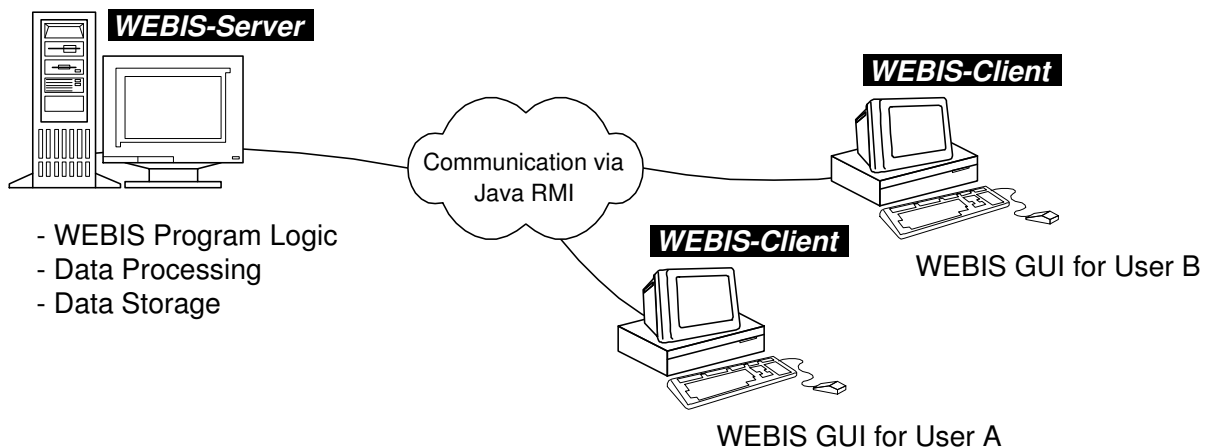


Abbildung 2.3: WEBIS-Client/Server-Architektur (siehe [Ahn05])

## 2.4 Spezialisierung auf die Börse

Das WEBIS-Projekt erforscht bis dato die typischen Aspekte eines im WWW-Umfeld angesiedelten Informationssystems. Das Spektrum dieser Untersuchungen deckt dabei die folgenden Kernbereiche ab:

- grundlegende Architektur eines solchen Informationssystems
- WWW als Datenquelle
- XML-basierte Speicherung der angesammelten Daten
- Transformation von Daten in Informationen
- und schließlich Präsentation der Informationen

Eine Spezialisierung auf ein bestimmtes Themengebiet, in dem man dann Data Mining und Datentransformationen durchführen kann, war aber sinnvoll: WEBIS wurde auf Börse getrimmt, ist aber prinzipiell auch weiterhin nicht auf diesen Anwendungsbereich eingeschränkt. Börsenorientierte Daten eignen sich besonders, weil sie im WWW nicht nur in großem Umfang verfügbar, sondern auch frei zugänglich sind und darüber hinaus eine leicht verständliche Semantik besitzen.

WEBIS war nun ein Informationssystem, was darauf spezialisiert wurde, Kaufempfehlungen für Aktien zu geben, die aus Aktienkursen und Nachrichten aus Börsentickern gewonnen wurden. Zum Systemaufbau gehörten ein Wrapper, ein Wrapper-Generator, ein Web-Monitor, ein Repository und eine Komponente für die Auswertung.

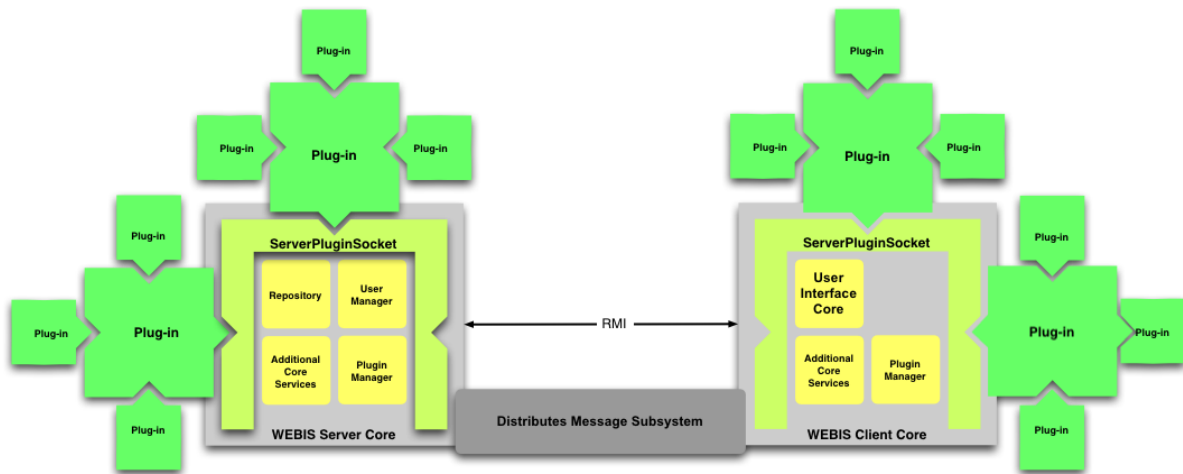


Abbildung 2.4: WEBIS-Plug-in-Architektur (siehe [Mei04])

Die Aufgabe des Wrappers ist es, vorgegebene Webseiten aus dem Internet zu holen, die gewünschten Daten zu extrahieren, in ein gegebenes Schema zu transformieren und im Repository zu speichern. Der Wrapper-Generator erzeugt die Wrapper, die für unterschiedliche Seiten benötigt werden. Um die Daten auf dem aktuellsten Stand zu halten überprüft der Web-Monitor regelmäßig, ob sich die Daten auf den Quell-Seiten verändert haben. Gegebenenfalls wird der Wrapper ausgelöst und die Daten werden aktualisiert. Der Auswertungsteil wandelt die im Repository gespeicherten Daten (Aktienkurse) in Informationen (Kaufempfehlungen) um.

## 2.5 Plug-in-Architektur

Es gab in der jüngeren Vergangenheit weitere große Veränderungen, die sowohl die Funktionalität als auch die Architektur betreffen. WEBIS ist in der aktuellen Variante auf Server- und auf Client-Seite Plug-in-basiert. Dadurch wurde die Funktionsvielfalt ausgelagert und kann nach Wunsch ausgewählt werden, was der Übersichtlichkeit spezielle der grafischen Oberfläche gut tat.

Das schematische Prinzip dieses neu integrierten Konzeptes ist in Abbildung 2.4 zu sehen, Abbildung 2.5 zeigt das veränderte Aussehen der Benutzeroberfläche – es ist im Hauptarbeitsbereich nur ein Fenster groß geöffnet, in dem man bequem agieren kann.

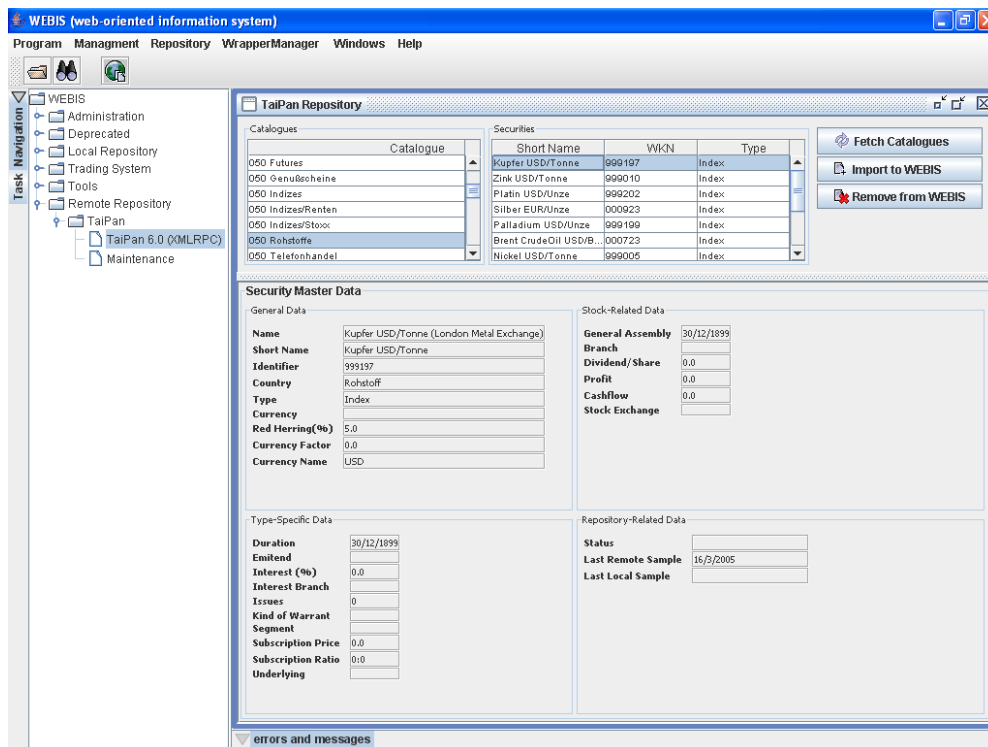


Abbildung 2.5: WEBIS-GUI im Jahr 2005 mit Plug-in-Auswahl im linken Fenster (siehe [Mei05])

## 2.6 Weitere Entwicklungen

Eine der Funktionen, die wie viele andere nachträglich zum Plug-in umgewandelt wurden, also schon vorher auf Server- und/oder Clientseite implementiert waren, ist die Webapplikationskomponente, durch die WEBIS auch von einem Webbrowser aus gesteuert werden kann.

Die Integration der Vorhersagesoftware, die für diese Arbeit implementiert wurde, geschieht mittels zwei Plug-ins: einem Client- und einem Server-Plug-in. Technologisch interagieren diese beiden Plug-ins via RMI miteinander, die direkte API ist jedoch versteckt und man verwendet die speziell entwickelten WEBIS-Schnittstellen, um die Kommunikation zu garantieren.

# 3 Softwareentwicklung

## 3.1 Entwicklung vom Prototyp zur GUI-basierten Software

[PB96] definieren: „Ein Software-Prototyp ist ein – mit wesentlich geringerem Aufwand als das geplante Produkt – hergestelltes und zu erweiterndes, ausführbares Modell des geplanten Software-Produkts, das nicht notwendigerweise alle Eigenschaften des Zielsystems besitzen muß, jedoch so geartet ist, daß vor der eigentlichen Systemimplementierung der Anwender die wesentlichen Systemeigenschaften erproben kann.“

Vor allem in der Automobil- und der Elektronikbranche sind Prototypen nicht wegzudenken aus dem Alltag der industriellen Fertigung, in der Softwaretechnik hingegen gehen die Meinungen auseinander. Einige Experten sehen in einem Quick-and-dirty-Prototyp einen schweren Verstoß gegen die Regeln der Softwaretechnik, für andere ist dieser der effiziente Weg zu einer tragfähigen und verständlichen Spezifikation.

### 3.1.1 Vor- und Nachteile von Prototypen

[Kro97] nennt als Vorteile unter anderem, dass

- wegen der frühen Anwendung die Anforderungen präzisiert werden können,
- die in den ursprünglichen Anforderungen enthaltene unnötige Funktionalität während der Entwicklung reduziert wird,

als Nachteile werden aufgeführt, dass

- die Gefahr besteht, ein chaotisches, nichthomogenes System zu entwickeln, wenn kein klares Gesamtkonzept existiert und
- wiederverwendbare Eigenschaften evtl. lange unentdeckt bleiben, da bestimmte Funktionen frühzeitig implementiert werden.

Einen Prototyp kann man

- ausführen

- erweitern
- als Basis zur Erprobung von Systemeigenschaften verwenden
- verwerfen, ohne bereits zu große Anstrengungen in eine als nicht sinnvoll betrachtete Entwicklung gesteckt zu haben

### 3.1.2 Formen von Prototypen

Hauptsächlich kann zwischen *Wegwerfprototypen* und *wiederverwendbaren Prototypen* unterschieden werden. Wie die Namen bereits aussagen wird die entstandene Software entweder wiederverwendet oder verworfen. Bei beiden Formen können entweder Teile aller Schichten der Systemarchitektur oder nur bestimmte Architekturschichten abgedeckt sein.

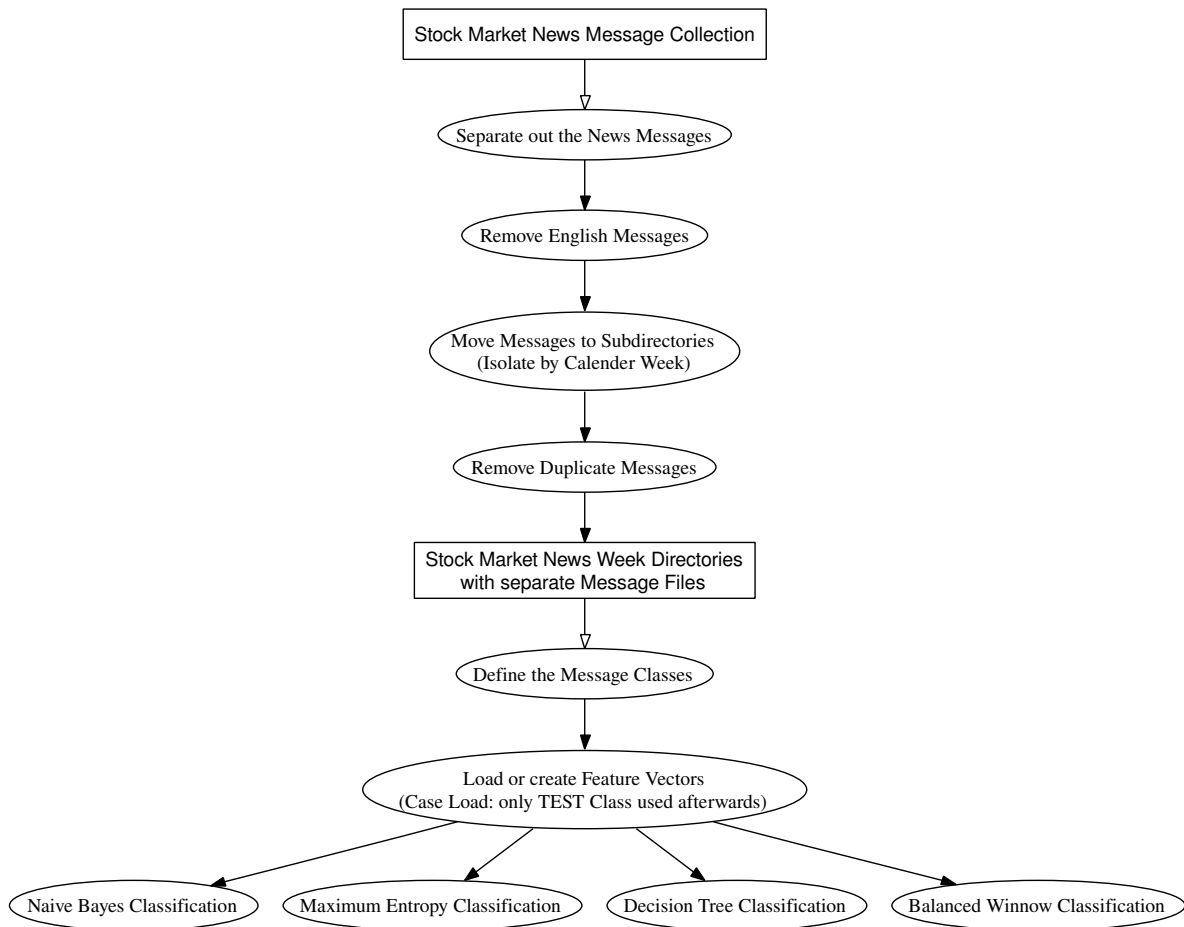
In [Kre06] wurde eine *Spezifikationsprototyp* entwickelt, mit dem ein Szenario „durchgespielt“ werden konnte, in dem Fall die Kernfunktionalität. Weit verbreitet sind im Gegensatz dazu jedoch *Oberflächenprototypen*, die vor allem die Benutzersicht simulieren, Programmausgaben jedoch meist fest deklariert sind und noch nicht von der Benutzereingabe abhängen.

## 3.2 Softwareübersicht

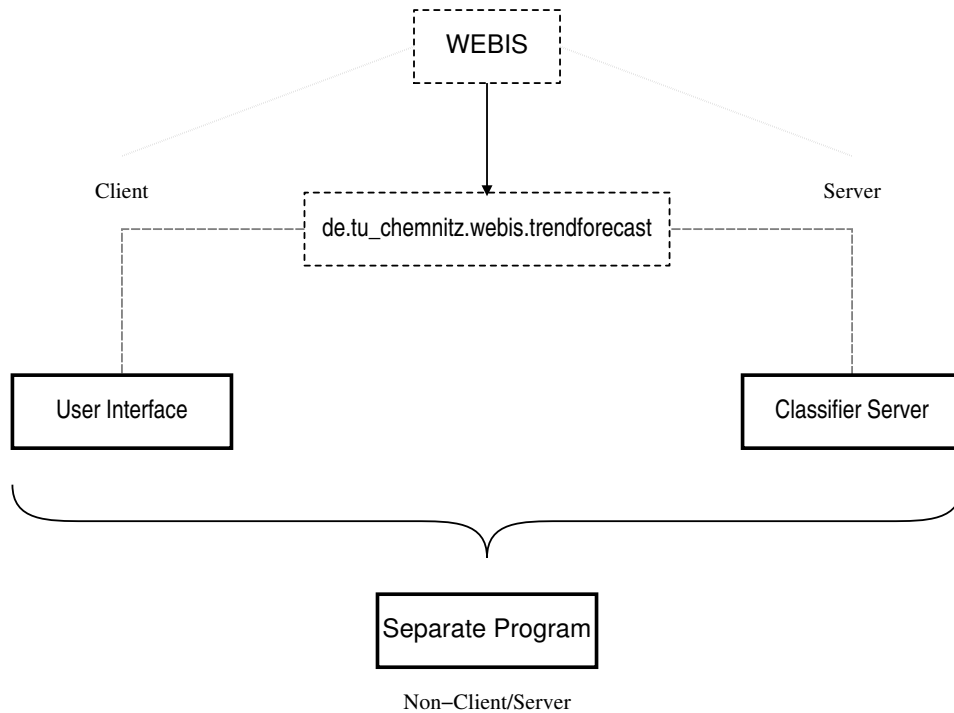
Die Abbildung 3.1 stellt das gedachte schematische Vorgehen der Vorhersagesoftware vor.

Bereits der Prototyp verwendete Mallet [McC02] als Klassifikationsbibliothek. Dies wird fortgesetzt, und es wurden nun weitere zur Verfügung stehende Classifier ausprobiert und in das neue System integriert. Bevor diese insgesamt vier Classifier kurz vorgestellt werden, wird aber auf einen weiteren interessanten Punkt eingegangen, der ebenfalls neue Erkenntnisse versprach, jedoch in Mallet nicht verfügbar war und somit selbst implementiert wurde: Feature Reduction - Reduzierung der Trainings-/Testvektoren.

Die „Komponentenzusammensetzung“ der entwickelten Java-Klassen ist in Abbildung 3.2 dargestellt. Nur die GUI-Implementierung mit den Entfernte-Methoden-Aufrufen (auf dem Server) und das RMI-Interface auf der Serverseite unterscheiden die Codestruktur der Client-/Server-WEBIS-Implementierung und des separaten Programms voneinander.



*Abbildung 3.1: Prinzipieller Bearbeitungsablauf*



**Abbildung 3.2:** Softwareübersicht: Funktionalität der beiden WEBIS-Plug-ins im separaten Programm vereint

### 3.3 Feature Reduction

In [Obe05a] und [Seb02] wird Feature Reduction als Mittel zur Verhinderung von Overfitting-Effekten bei bestimmten Klassifikatoren, aber auch als Verfahren zur Verbesserung der Effektivität der Vektoren und somit auch zur Klassifikationsverbesserung bei Verfahren, die große Trainingsmengen benötigen (z. B. Naive Bayes).

```

if ((keepFeaturesPercent < 100) && (ilist_full.size() > 1))
{
    /* <feature reduction> */
    java.util.Vector<Double> v = new java.util.Vector<Double>(
        ilist_full.size());

    for (int i = 0; i < ilist_full.size(); i++)
    {
        int numFeatures = ilist_full.getDataAlphabet().size();
        double[] featureCountSum = new double[numFeatures];
        Instance inst = ilist_full.getInstance(i);
        Labeling labeling = inst.getLabeling();
        FeatureVector fv = (FeatureVector) inst.getData();
    }
}
  
```

```

double labelWeightSum = 0;
for (int ll = 0; ll < labeling.numLocations(); ll++)
{
    // int li = labeling.indexAtLocation(ll);
    double labelWeight = labeling.valueAtLocation(ll);
    labelWeightSum += labelWeight;
    if (labelWeight == 0)
        continue;
    double cnt = labelWeight * ilist_full.getInstanceWeight(i);
    for (int fl = 0; fl < fv.numLocations(); fl++)
    {
        int fli = fv.indexAtLocation(fl);
        if (fv.valueAtLocation(fl) > 0)
            featureCountSum[fli] += cnt;
    }
}
assert (Math.abs(labelWeightSum - 1.0) < 0.0001);

double avg = 0.0;
for (int z = 0; z < featureCountSum.length; z++)
    avg += featureCountSum[z];
avg = (avg / featureCountSum.length);
v.add(avg);
}

double sort[] = new double[ilist_full.size()];
int ilmap[] = new int[ilist_full.size()];

final double correct = Math.log(2.3403e-3 / 672) / Math.log
    (1.5931e-2 / 4881);
/* average of the about 350 worst valued entries (mainly because
 * of the file size) straightened to the 50 best valued entries
 * 0.0015931    0.0023403
 * ----- = ----- (power function chosen because of
 *      x          x      its characteristics) x ~ 0.96662767
 * 4881        672
 */
for (int z = 0; z < sort.length; z++)
{
    int y = 0;
    int ei = z;
    double ed = v.get(z).doubleValue();
    try
    {

```

```

        File f = new File(new URI(ilist_full.getInstance(z).getName
            ().toString()));
        ed = ed / Math.pow((double)f.length(), correct);
    }
    catch (URISyntaxException use)
    {
        use.printStackTrace();
    }

    while ((y < z) && (ed < sort[y]))
        y++;

    while (y < (z + 1))
    {
        double td;
        int ti;
        if (y < z)
        {
            td = sort[y];
            ti = ilmap[y];
        }
        else
        {
            td = 0.0;
            ti = -1;
        }
        sort[y] = ed;
        ilmap[y] = ei;
        ed = td;
        ei = ti;
        y++;
    }
}
int keep = (sort.length *
    ((keepFeaturesPercent > 100) ? 100 :
    ((keepFeaturesPercent < 1) ? 1 :
    keepFeaturesPercent)) / 100;
if (keep < 1)
    keep = 1;
ilist = new InstanceList();
for (int z = 0; z < keep; z++)
    ilist.add(ilist_full.getInstance(ilmap[z]));
/* </feature reduction > */
} else

```

```
ilist = ilist_full;
```

**Listing 3.1:** Codefragment aus *de.tu\_chemnitz.webis.trendforecast.mallet.MalletText2Vectors*, welches die Feature Reduction durchführt

Das Verfahren verwendet also die Worthäufigkeiten, wendet dann jedoch einen dynamischen Faktor an, um nicht nur große Trainingsdateien zu bevorzugen. Aus dem Codelisting 3.1 ist das prinzipielle Vorgehen ersichtlich.

## 3.4 Wählbare Classifier

In der für diese Arbeit erstellten Software kann man zwischen 4 verschiedenen Classifiern, die von Mallet bereitgestellt werden, wählen, mit denen die Klassifikation durchgeführt werden soll: *Balanced Winnow*, *Decision Tree*, *Maximum Entropy* und *Naive Bayes*, die im Folgenden kurz prinzipiell vorgestellt werden:

### 3.4.1 Naive Bayes

Ein Bayes-Klassifikator leitet sich aus dem Satz von Bayes ab, der im Folgenden kurz erklärt wird.

Der Satz von Bayes ist ein Ergebnis der Wahrscheinlichkeitstheorie, benannt nach dem Mathematiker Thomas Bayes. Er gibt an, wie man mit bedingten Wahrscheinlichkeiten rechnet. Für zwei Ereignisse  $A$  und  $B$  sei  $P(A)$  die A-Priori-Wahrscheinlichkeit von  $A$ ,  $P(B|A)$  die bedingte Wahrscheinlichkeit von  $B$ , wenn  $A$  auftritt, und  $P(B)$  die A-Priori-Wahrscheinlichkeit von  $B$ . Dann gilt für die bedingte Wahrscheinlichkeit  $P(A|B)$ :

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

Dieser auch Bayessches Theorem genannte Zusammenhang gilt für endlich viele Ereignisse, wenn  $A_i$  für  $i = 1, \dots, N$  eine Zerlegung des Ereignisraumes in disjunkte Ereignisse ist, folgendermaßen:

$$P(A_i|B) = \frac{P(B|A_i) \cdot P(A_i)}{P(B)} = \frac{P(B|A_i) \cdot P(A_i)}{\sum_{j=1}^N P(B|A_j) \cdot P(A_j)}$$

Dabei ist  $P(A_i|B)$  die A-Posteriori-Wahrscheinlichkeit.

Ein Bayes-Klassifikator ordnet jedes zu klassifizierende Objekt der Klasse zu, zu der es mit der größten Wahrscheinlichkeit gehört. Die Grundannahme dabei ist, dass jedes Attribut nur vom Klassenattribut abhängt. Obwohl dies in der Realität selten zutrifft, erzielen naive Bayes-Klassifikatoren bei praktischen Anwendungen häufig gute Ergebnisse, solange die Attribute nicht zu stark korreliert sind.

Gegeben sei

$$b : \mathbb{R}^f \rightarrow C$$

als Funktion, die Vektoren aus dem  $f$ -dimensionalen reellwertigen Merkmalsraum auf eine Menge von Klassen  $C$  abbildet.  $b$  ist ein Bayes-Klassifikator.

In dem für die Trendvorhersage verwendeten Fall gilt  $C := \{0, 1\}$ , da genau zwei Klassen (*UP* und *DOWN*) betrachtet werden.

Die Wahrscheinlichkeitsdichte der Klasse  $i$ ,  $i \in C$ , sei durch  $p_i$  gegeben. Die Wahrscheinlichkeit, dass ein gegebener Merkmalsvektor in die Klasse  $i$  gehört, ist also durch  $p_i(x)$  gegeben. Der Bayes-Klassifikator definiert sich dann nach dem A-Posteriori-Kriterium:

$$b := \operatorname{argmax}_i p_i(x)$$

### 3.4.2 Maximum Entropy

Die Maximum-Entropie-Methode (MEM) wird zur optimalen Extraktion von Information aus verrauschten Signalen verwendet. Sie basiert auf dem Prinzip der maximalen Entropie (Principle of Maximum Entropy), das von Edwin Thompson Jaynes 1957 in die statistische Mechanik eingeführt wurde, und hängt von dem Signal-Rausch-Verhältnis ab. Die Methodik wurde von Burg 1967 und Jaynes 1968 weiter ausgearbeitet und findet vor allem Anwendung in der Spektralanalyse und der digitalen Bildverarbeitung. Die Wissenschaft hat aber bereits häufig gezeigt, dass es sehr interessant und erfahrungsreich sein kann, Verfahren aus anderen Forschungsdisziplinen auszuprobieren.

Die MEM ist also eine Regularisierungsmethode für die numerische Lösung von Problemen mit verrauschten Daten, bei der die Entropie als Regularisierungsterm eingesetzt wird. In der MEM ist die Regularisierung (oder Glättung), die die Grobheit der berechneten ungefähren Lösung  $\tilde{f}$  misst, gegeben durch den Negativbetrag der Shannon-Entropie von  $\tilde{f}$ :  $-H(\tilde{f})$

Bei stetigem Verlauf gilt ( $A$  beschreibt den Definitionsbereich von  $\tilde{f}$ ):

$$H(\tilde{f}) \equiv - \int_A \tilde{f}(x) \log \tilde{f}(x) dx$$

$\tilde{f}$  ist normalisiert, so dass  $\int_A \tilde{f}(x) dx = 1$  (und  $0 \log 0 = 0$ ).

Für den diskreten Fall gilt:

$$H(\tilde{f}) \equiv - \sum \tilde{f}(x_i) \log \tilde{f}(x_i)$$

$\tilde{f}$  ist normalisiert, so dass  $\sum \tilde{f}(x_i) = 1$ .

Die zu Grunde liegende Philosophie dieser Methode ist, dass man maximal unentschlossen bei der Beachtung der Unzulänglichkeit dieser Daten ist, wenn man die Entropie maximiert und somit den Informationsgehalt von  $\tilde{f}$  minimiert.

Für eindimensionale diskrete Faltungsgleichungen mit unverrauschten und periodischen Daten existiert eine geschlossene Lösung. Für andere Fälle werden iterative Methoden benutzt oder andere Formen der Regularisierungsmethode. Die Nichtnegativität der berechneten Lösung ist ein bei dieser Methode stets gegeben.

### 3.4.3 Balanced Winnow

Der Winnow-Algorithmus, der in [Lit87] eingeführt wurde, ist ein statistisches Verfahren, das eine Wertigkeit für jede Klasse kalkuliert statt die Wahrscheinlichkeiten direkt zu berechnen.

Berechnet wird ein  $n$ -dimensionaler Gewichtsvektor  $\omega^c = (\omega_1^c, \omega_2^c, \dots, \omega_n^c)$  für jede Klasse  $c$ .  $\omega_i^c$  ist das Gewicht des  $i$ -ten Features. Zurückgegeben wird 1, wenn für eine Klasse die summierten Gewichte aller aktiven Features einen vordefinierten Schwellenwert übertreffen, sonst 0. Für diese Wertigkeit  $\Omega$  gilt mit dem Schwellenwert  $\theta$  und der Anzahl aktiver Features  $n_a \leq n$ :

$$\Omega = \sum_{j=1}^{n_a} \omega_j^c > \theta$$

Das Ziel dieses Algorithmus ist das Erlernen eines linearen Trenners des Feature-Raumes. Der initiale Wert jedes Features ist 1,0. Die Gewichte jeder Klasse werden fortlaufend angepasst, wenn der Rückgabewert einer Klasse nicht der Erwartungswert (Trainingswert) ist. Wenn 0 statt 1 zurückgegeben wird, werden alle Gewichte der aktiven Features mit einem *Beförderungsfaktor*  $\alpha$  multipliziert,  $\alpha > 1 : \omega_j^c \leftarrow \alpha \times \omega_j^c$ . Wenn 1 statt 0 zurückgegeben wird, werden alle aktiven Gewichte mit einem *Degradierungsfaktor*  $\beta$  multipliziert,  $0 < \beta < 1 : \omega_j^c \leftarrow \beta \times \omega_j^c$

Da bei der Textklassifikation die Featureanzahl stark von der Textlänge abhängt, nimmt man statt eines festen Schwellenwertes die Anzahl  $n_a$  der aktiven Features, also  $\theta = n_a$ . Die anfänglichen Wertigkeiten sind demnach genau  $\theta$ , da anfänglich ja alle Features ein Gewicht von 1,0 haben.

Der so genannte Balanced-Winnow-Algorithmus verwendet zwei Gewichte für jedes Feature,  $\omega^+$  und  $\omega^-$ . (Betrachtet wird hier nur der binäre Fall.) Als Koeffizient eines Features wird dann die Differenz der beiden Gewichte genommen, und somit werden negative Koeffizienten zugelassen. Der Rückgabewert 1 ergibt sich nun für eine Dokumentinstanz  $s = (s_1, s_2, \dots, s_n)$ , wenn

$$\Omega = \sum_{j=1}^n (\omega_j^+ - \omega_j^-) s_j > \theta$$

Initial wird  $\omega^+$  auf  $\frac{2\theta}{d}$  und  $\omega^-$  auf  $\frac{\theta}{d}$  gesetzt ( $d$  ist die durchschnittliche Anzahl aktiver Features eines Dokuments), damit für ein „durchschnittliches“ Dokument der Wert  $\theta$  berechnet wird. Bei einer falsch berechnete positiven Trainingseinheit werden die Gewichte der aktiven Features gleichzeitig mit dem Faktor  $\alpha$  befördert ( $\omega^+$ ) und mit dem Faktor  $\beta$  degradiert ( $\omega^-$ ), bei einer falsch berechneten negativen Trainingseinheit entsprechend umgekehrt.

### 3.4.4 Decision Tree

Ein Entscheidungsbaum (Decision Tree) dient der Veranschaulichung aufeinanderfolgender, hierarchischer Entscheidungen. Beginnend mit einem Stamm führen Verzweigungen, die in mit Wahrscheinlichkeiten versehenen Äste führen, zu Endpunkte. Die Endpunkte sind durch einen eindeutigen Weg erreichbar. Im binären Fall ist jede Verzweigung eine Ja/Nein-Entscheidung.

In solch einem Graph ist jede Regel einfach dadurch ablesbar, indem man von der Wurzel her den Ästen des Baumes folgt, bis man an dem bestimmten Blatt angelangt ist, welches das Resultat der Entscheidungsfolge darstellt.

Generiert werden die Entscheidungsbäume üblicherweise im Top-Down-Prinzip. Bei jedem Schritt wird genau das Attribut gesucht, mit welchem man die Daten am besten klassifizieren kann. Um dieses zu ermitteln muss der beste Split gefunden werden, d.h die Aufteilung der Daten muss so gewählt werden, dass sie nach der Aufteilung möglichst rein sind. Ein Maß für dies Reinheit ist die Entropie. Aus der Entropie lässt sich dann berechnen welches Attribut für den Split den höchsten Informationsgewinn bietet. Ein weiteres Maß für die Bestimmung des besten Splits ist der Gini-Index (statistisches Maß für Verteilungsgleichheit).

Die Größe von Entscheidungsbäumen kann mittels Pruning-Verfahren (Beschneidung/-Verkleinerung des Baums für eine höhere Effizienz) – entweder wird die Baumtiefe beschränkt oder eine Mindestanzahl von zu klassifizierenden Objekten pro Knoten festgelegt – begrenzt sein, auch um die Übersichtlichkeit zu bewahren. In dem hier für die

Trendvorhersage verwendeten Klassifizierungsfall werden jedoch stattdessen die Eingabevektoren reduziert, um den Trainingsaufwand und die damit verbundene Baumgröße zu beeinflussen.

Durch die Verfahren zur Größenbeschränkung wird auch versucht das Phänomen des Overfitting zu verhindern. Overfitting ist die experimentelle Feststellung, dass bei zunehmender Baumkomplexität die Klassifikationsgüte auch wieder abnehmen kann.

## 3.5 Beispielnutzung des Systems

### 3.5.1 Erstellen einer Konfiguration und komplette Klassifikation am Beispiel

Die entwickelte Software lässt sich als WEBIS-Plug-in ohne Umgewöhnung genau so nutzen wie als separates Programm (wenn man davon absieht, dass natürlich der Programmstart – bei WEBIS: Serverstart → Clientstart → Trend-Forecast-Plug-in starten; im Falle des Einzelprogramms ist es der Aufruf eines JAR-Archives mit der JRE – sehr unterschiedlich ist). Laufen WEBIS-Server und -Client auf dem gleichen Rechner, merkt man bis auf die unterschiedliche Implementierung des Datei-öffnen- bzw. Verzeichnisauswahldialoges auch beim Zugriff auf Verzeichnisse/Dateien keinen Unterschied.

Das *Trend Forecast Engine* „getaufte“ System ist zum Einen zum Erstellen eines Klassifikators geeignet als auch simpel zum Klassifizieren von neuen Texten. Ein kompletter Ablauf bis zum Abspeichern einer Konfiguration sowie die Nutzung einer solchen danach wird nun beschrieben.

Im oberen Teil der Programmoberfläche befindet sich das Panel *Message Preparation*. Die Funktion dieses Panels bezieht sich auf Eingabetexte, die aus der Börsensoftware Tai-Pan [TP] exportiert werden können oder Eingabetexte, die dem Schema (siehe [Kre06]) entsprechen. Bevor die Nachrichtenvorverarbeitung ausgeführt werden kann, muss die Eingabedatei (Abbildung 3.3) sowie ein Verzeichnis, in dem die separierten und in Unterverzeichnisse nach Kalenderwochen aufgesplitteten Tickertexte abgelegt werden, ausgewählt werden. Zudem kann ausgewählt werden, ob englische Nachrichten durch simple statistische Untersuchung entfernt werden sollen, oder – jeweils pro Kalenderwoche – doppelte Nachrichten verhindert werden sollen. (Abbildung 3.4) Weiterhin ist es möglich, die Ausgabe auf bestimmte Kategorien, die in Tai-Pan definiert sind, zu beschränken.

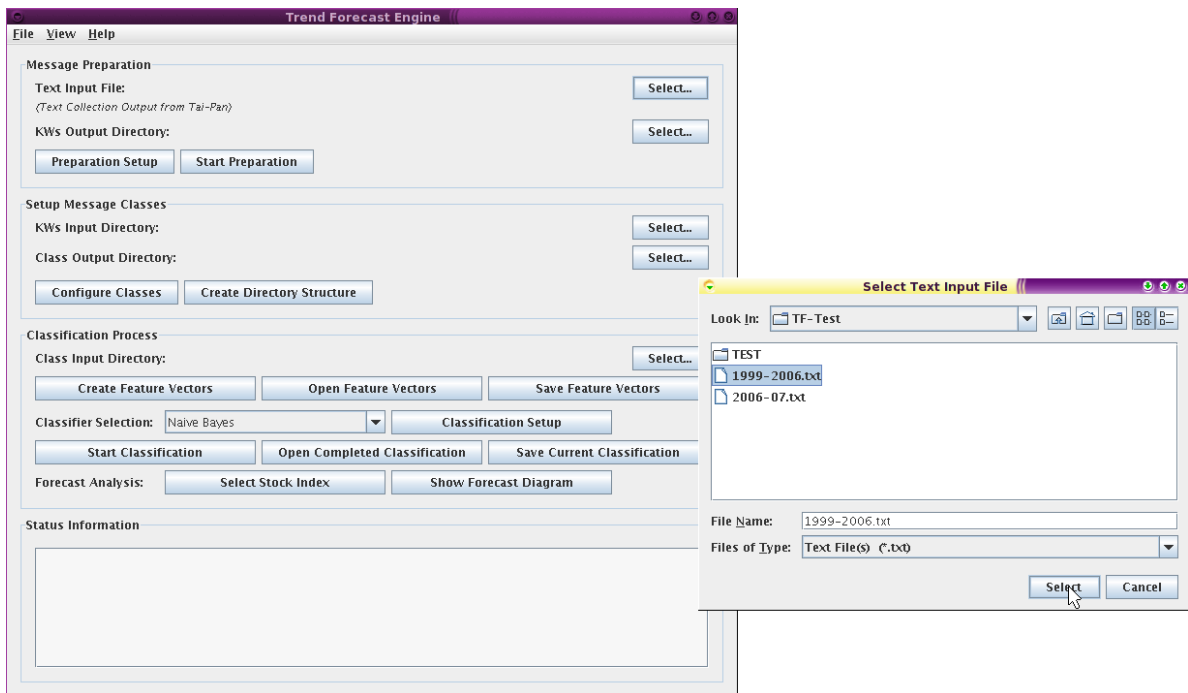


Abbildung 3.3: Trend Forecast Engine: Auswählen der Eingabedatei mit den Tickerdaten aus Tai-Pan [TP]

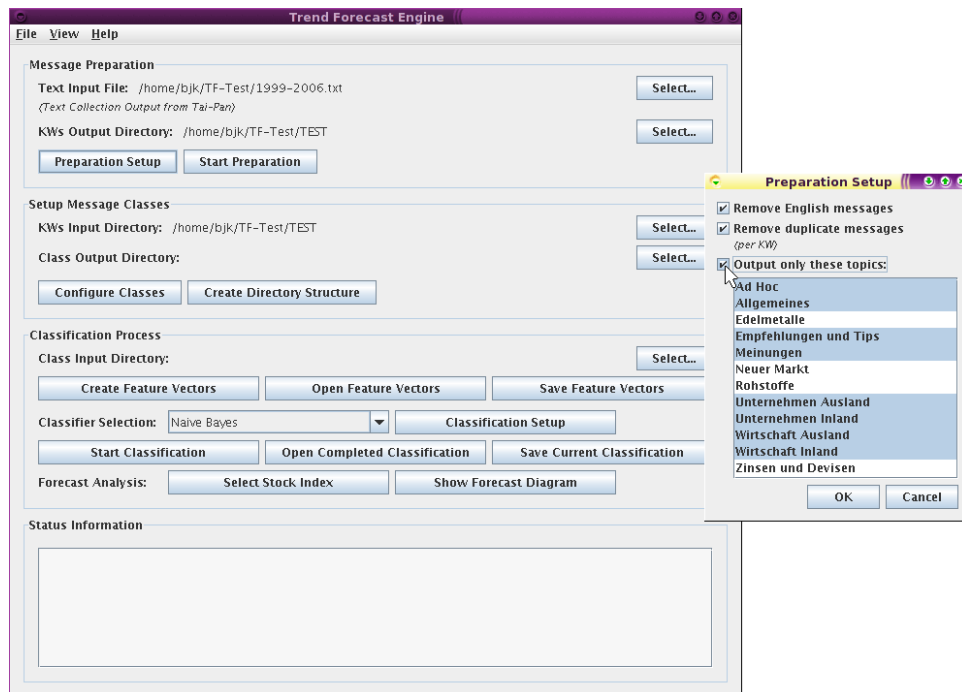
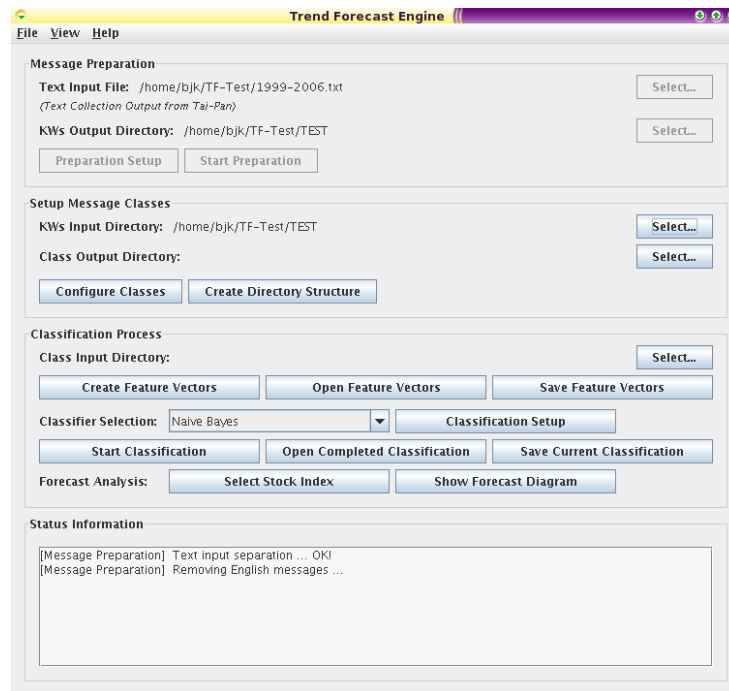


Abbildung 3.4: Trend Forecast Engine: Treffen der Vorverarbeitungsoptionen

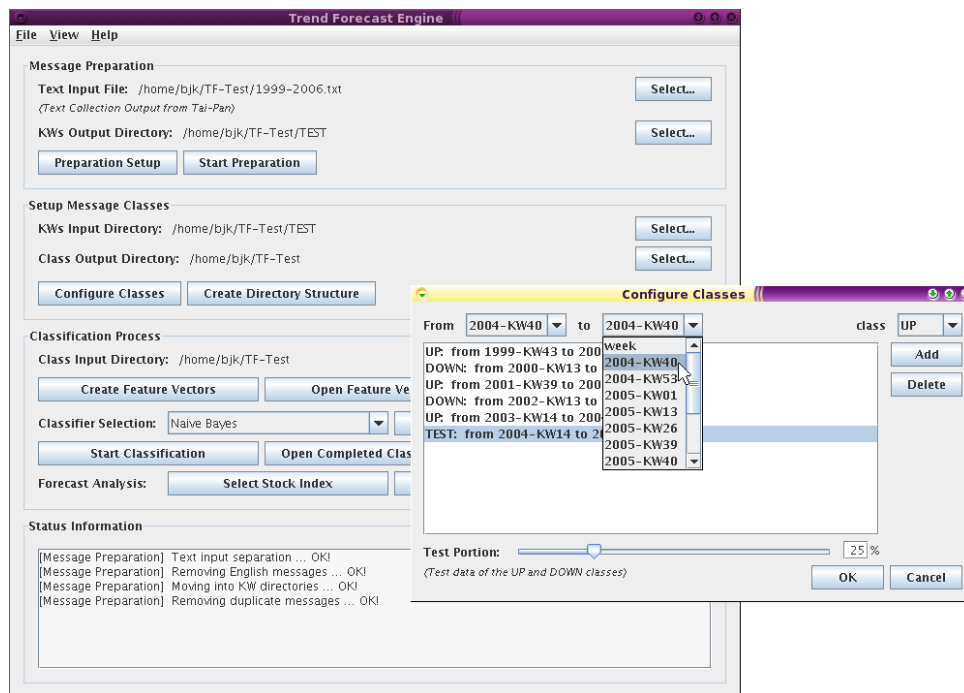


**Abbildung 3.5:** Trend Forecast Engine: Vorverarbeitung läuft, das obere Panel ist inaktiv

Die Vorverarbeitung (Abbildung 3.5) beinhaltet neben der Aufspaltung der Texteingabedatei das Umwandeln von teilweise vorhandenen DOS-Umlautzeichen<sup>1</sup>. (Das ist zumindest in älteren Texten aufgefallen.) Außerdem werden Zeichenketten wie Autorsignaturen oder Absenderkennungen, die erkennbar mit der Nachricht keinen Zusammenhang haben, herausgefiltert.

Das zweite Panel – *Setup Message Classes* – dient dem Anlegen der Nachrichtenklassen. Eingabedateien können hier aus jeder denkbaren Quelle sein, solange sich das Verzeichnis- und Dateinamenschema dem gewünschten Kalenderwochenprinzip anpasst: Als Eingabeverzeichnis muss ein Verzeichnis gewählt werden, das Kalenderwochenunterverzeichnisse enthält mit dem Verzeichnisnamenschema *YYYY-KWww*. (*YYYY* ist eine vierstellige Jahreszahlangabe, mit *ww* wird die stets zweistellige Angabe der Kalenderwoche codiert.) Die oben beschriebene Vorverarbeitung erzeugt darin Nachrichtendateien, die als Dateinamen Datum und Uhrzeit der jeweiligen Nachricht verwenden. (Dadurch geht diese Information nicht verloren, ist aber für einen Classifiertrainer erst gar kein Hindernis.) Wenn in Vorverarbeitungspanel das Ausgabeverzeichnis gewählt wird und das Eingabeverzeichnis her noch nicht gewählt wurde, wird es auf den gleichen Pfad gesetzt. Dies geschieht gegebenenfalls ebenso mit dem in diesem Panel zu setzenden

<sup>1</sup>DOS (*Disc Operating System*) verwendet gegenüber dem unter Microsoft Windows laufenden Tai-Pan-System eine andere Zeichensatzbelegung.



**Abbildung 3.6:** Trend Forecast Engine: Nachrichtenklassen einstellen und Testanteil bestimmen

Klassenausgabeverzeichnis und dem Eingabeverzeichnis für die Klassifikation im nächsten Panel.

Im Klassenkonfigurationsdialog (Abbildung 3.6) kann den einzelnen zur Verfügung stehenden Wochen nun eine der zwei Klassifikatorklassen (*UP* bzw. *DOWN*) oder *TEST* zugewiesen werden, um eine bestimmte Woche nur getestet und nicht als Trainingseinheit eingesetzt werden soll. Mit dem Prozentwert im unteren Fensterbereich wird gewählt, wie eine Klassifikatorklasse in Trainings- und Testmenge aufgeteilt wird. Es müssen nicht zwingend alle Wochen/Verzeichnisse „verbraucht“ werden. Nicht zugeordnete Wochen bleiben im gewählten Verzeichnis, die anderen werden in die Unterverzeichnisse *UP*, *DOWN* und *TEST* im gewählten Zielordner verschoben, die Klassifikatorklassen entsprechend in Trainings- und Testmenge aufgeteilt – das alles geschieht nach Anklicken des Buttons *Create Directory Structure*.

Das dritte Panel dient dem eigentlichen Klassifikationsprozess. Ist ein valides Eingabeverzeichnis gewählt (also eines mit Unterordnern *UP* und *DOWN* und jeweils mindestens einem Kalenderwochenverzeichnis darin), können die Feature Vectors für diese beiden Klassen erzeugt werden. Nach dem Klicken auf *Create Feature Vectors* kann dieser Vorgang noch genauer spezifiziert werden (Abbildung 3.7). Unter anderem ist es in dem dadurch geöffneten Dialog möglich die Eingabesprache (um intern die Liste der Stoppwörter korrekt zu wählen) festzulegen und zu wählen, wie eingeschränkt die Featurezahl

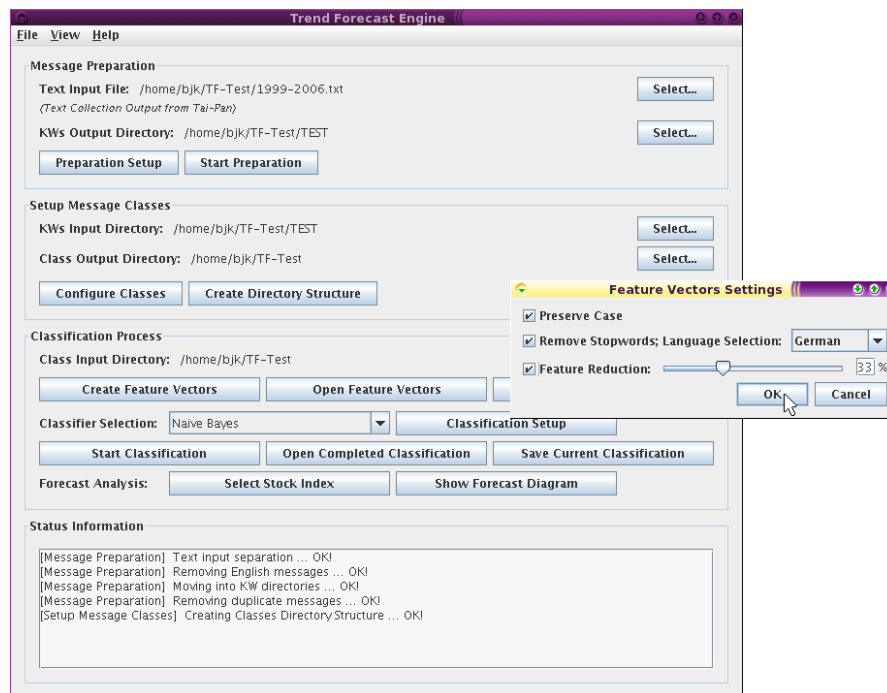


Abbildung 3.7: Trend Forecast Engine: Dialog zum Erstellen der Feature-Vektoren

sein soll.

Hat man bereits Vektoren erzeugt, kann man diese auch separat Abspeichern, um sie für einen späteren Klassifikationsprozess wieder Laden zu können. Diese Funktionalität ist jedoch im globalen Abspeichern der Konfiguration enthalten.

Es können nun einer der vier vorgestellten Klassifikatoren gewählt werden und die Einstellungen für den Klassifikationsprozess (Abbildung 3.8) getroffen werden. In diesem Dialog ist es auch wieder möglich eine Reduktion der Vektoren einzustellen, diesmal jedoch sind nur genau die Vektoren von jeder zu testenden Woche gemeint. (Das bedeutet speziell für den Fall, dass nur ein sehr geringer Testanteil für die beiden Klassifikatorklassen gewählt wurde, dass dann gegenüber einem „kompletten Wochenpfad,“ viel weniger Eingabedaten überhaupt zur Verfügung stehen, und diese dann auch noch reduziert werden.) Nachdem hier evtl. noch spezielle Einstellungen vorgenommen worden sind, kann die Klassifikation für die Testverzeichnisse im gewählten Eingabepfad gestartet werden. Aller zehn klassifizierter Wochen, wird im Statuspanel eine Fortschrittsinformation ausgegeben.

Nach Abschluss des Klassifikationsprozess können die Werte zur weiteren Verwendung im XML-Format abgespeichert werden oder direkt visualisiert werden. Dazu wird jedoch noch der Verlauf eines Aktienindex erwartet, zu dem man ja höchstwahrscheinlich auch die Klasseneinstellungen passend gemacht hat. Da Tai-Pan Kursinformationen nicht

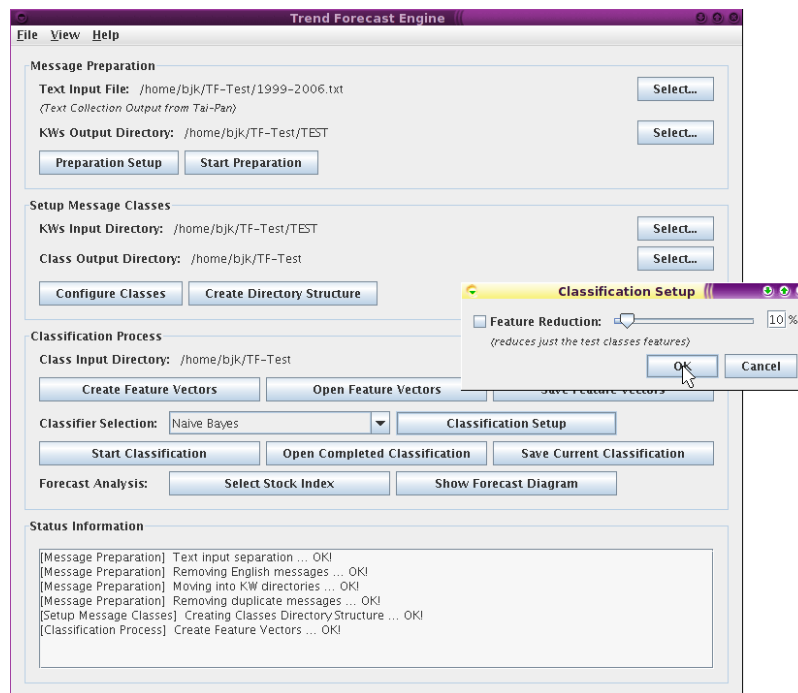


Abbildung 3.8: Trend Forecast Engine: Klassifikationseinstellfenster

in XML, jedoch im CSV-Format ausgeben kann, wurde dieses Eingabeformat gewählt und ein Parser implementiert. Die Klassifikationsergebnisse werden nach Drücken des Buttons *Show Forecast Diagram* in einem neuen Fenster gezeigt (Abbildung 3.10). Die dazu verwendete `jfreechart`-Bibliothek bietet vielfältige Möglichkeiten, u. a. Zoomen, Ausdrucken, Abspeichern als Grafik.

Speichert man nach einem Klassifikationsdurchlauf die Konfiguration ab (Abbildung 3.9), sind auch die Klassifikationsergebnisse mit gespeichert und werden beim nächsten Öffnen der Konfiguration als initiale Daten verwendet. Dadurch hat man z. B. beim Klassifizieren von Daten von nur einer neuen Woche auch noch die schon bekannten Klassifikationsergebnisse und kann Veränderungen erkennen. Wie man sinnvoll vorgeht, um Texte additiv zu klassifizieren, wird im nächsten Abschnitt beschrieben.

### 3.5.2 Testen neuer Daten mit vorhandener Konfiguration am Beispiel

Im *View*-Menü kann in die angepasste Ansicht *Add KW* (Abbildung 3.11) gewechselt werden, in der dann nur noch die sinnvollen Schritte zum Klassifizieren neuer Textdaten (speziell von Tai-Pan) aktiv sind und die wichtigsten Schritte durch Nummerierungsangaben speziell hervorgehoben sind. Durch die Festlegung des Verzeichnisnamens für die

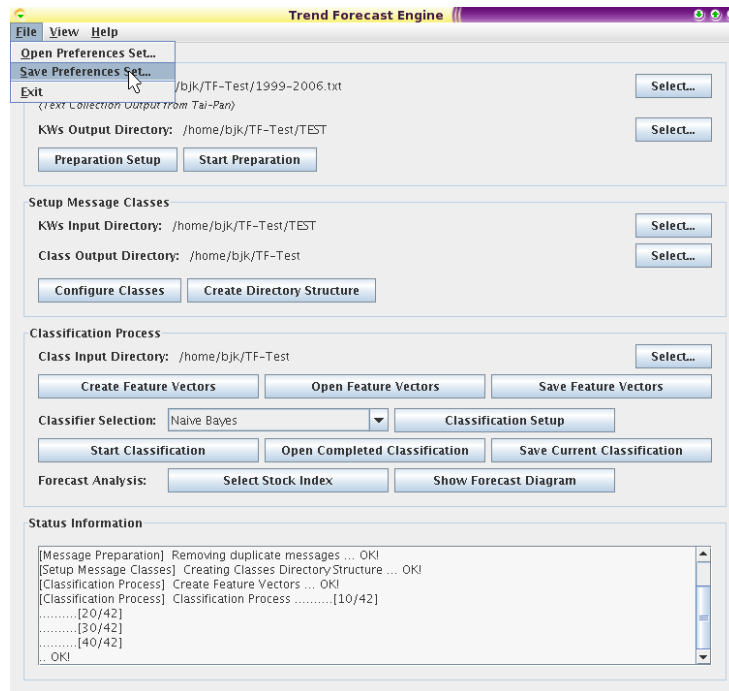


Abbildung 3.9: Trend Forecast Engine: Abspeichern der Konfiguration nach Klassifikationsdurchlauf

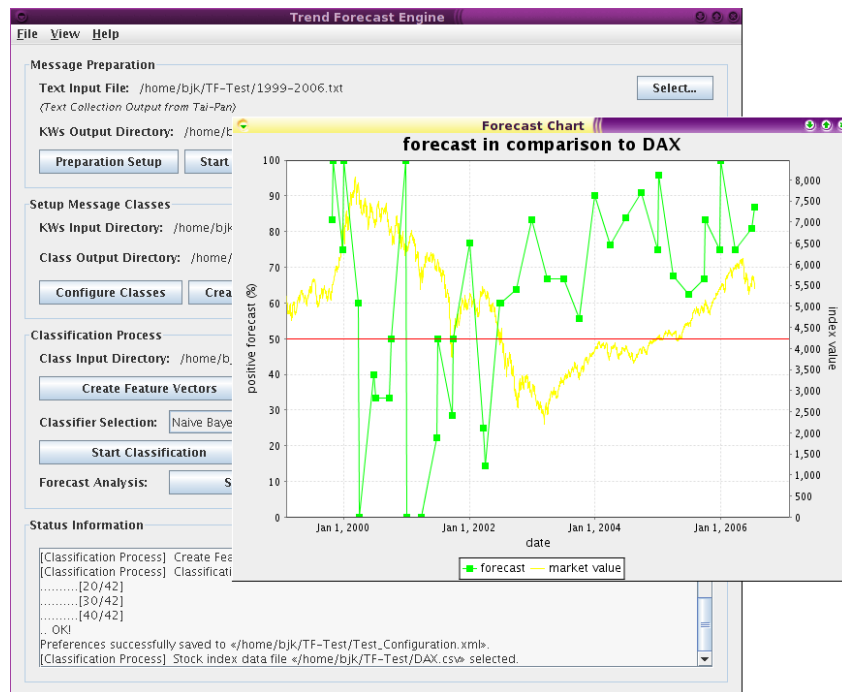
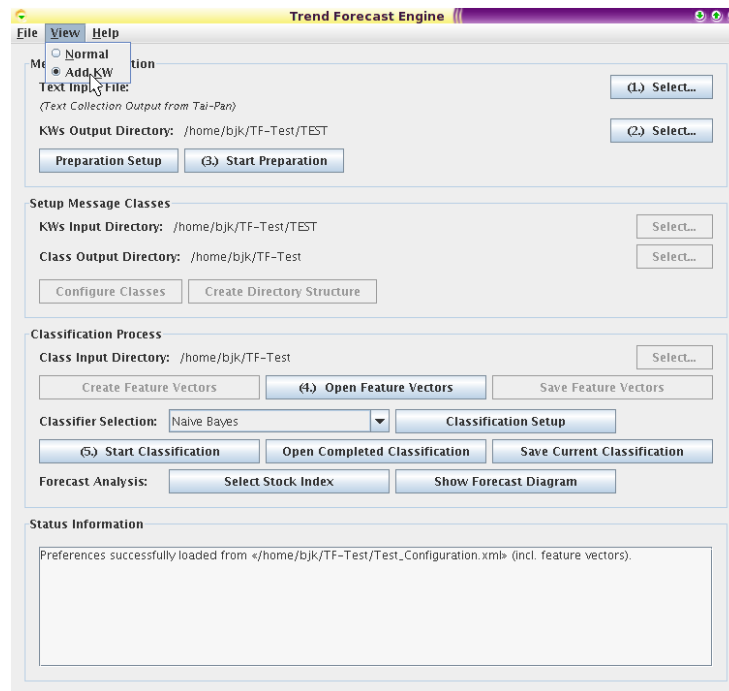


Abbildung 3.10: Trend Forecast Engine: Klassifikationsergebnis anschauen



**Abbildung 3.11:** Trend Forecast Engine: Ansichtswechsel – Nun nur Optionen zum Klassifizieren neuer Texte sichtbar

Testklassen auf *TEST* muss auch ein solches Verzeichnis als Ausgabeverzeichnis gewählt werden. Automatisch wird dann der Klasseneingabepfad auf das Verzeichnis eine Ebene höher festgelegt.

Abbildung 3.12 zeigt, dass das System stets in dem Panel, was „aktiv“ ist, die Funktionalität sperrt. Bei den wahrscheinlichen zeitintensiven Funktionen werden jedoch separate Threads gestartet, so dass ein Arbeiten mit einem anderen Panel parallel möglich ist.

Das zweite Panel ist im aktuellen Modus jedoch generell ausgegraut, da dessen Sinn ja im Festlegen von Klassen für einen Klassifikator liegt, hier aber nur das Erzeugen von *einer* Klasse notwendig, die ja zudem auch nur zum Testen genutzt wird.

Nach der Vorverarbeitung ist es noch notwendig, bereits generierte Feature Vectors zu laden (entweder indem über *File* → *Open Preferences Set...* eine bekannte Konfiguration, die Vektoren mit gespeichert hat oder indem über den Button *Open Feature Vectors* selbst die Vektoren geladen werden).

Der 5. markierte Schritt ist dann bereits das Klassifizieren, für das aber natürlich die Classifiereinstellungen wie gewohnt durchgeführt werden können. (Achtung: Es wird nicht gespeichert, mit welchem Classifier welche Klassifikationsergebnisse erzielt wurden

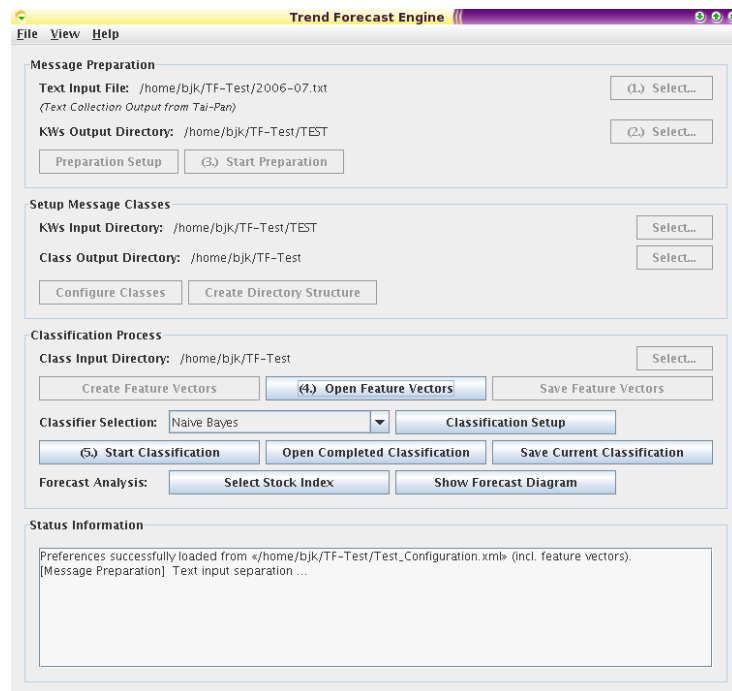


Abbildung 3.12: Trend Forecast Engine: Vorverarbeitung der neuen Testklassen aktiv

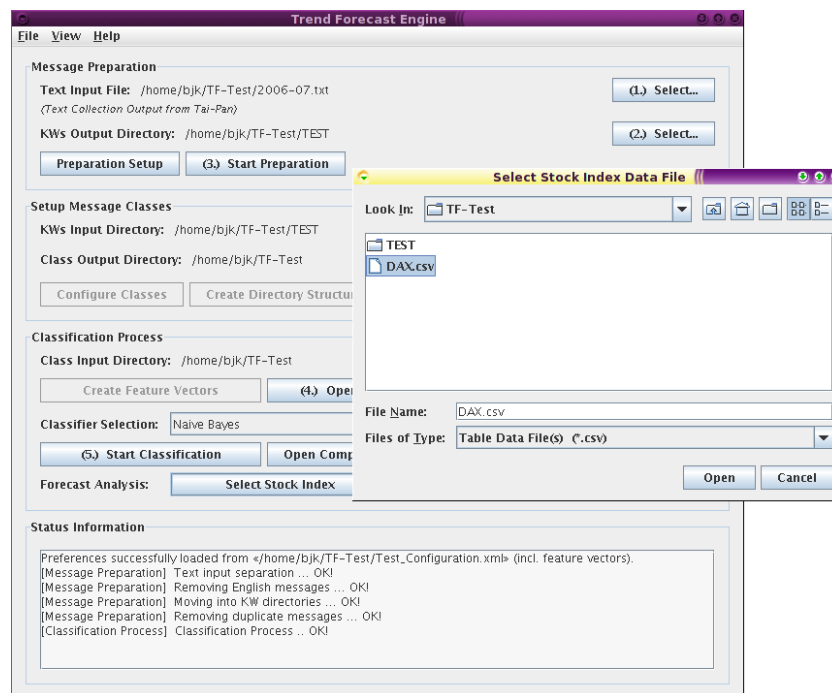
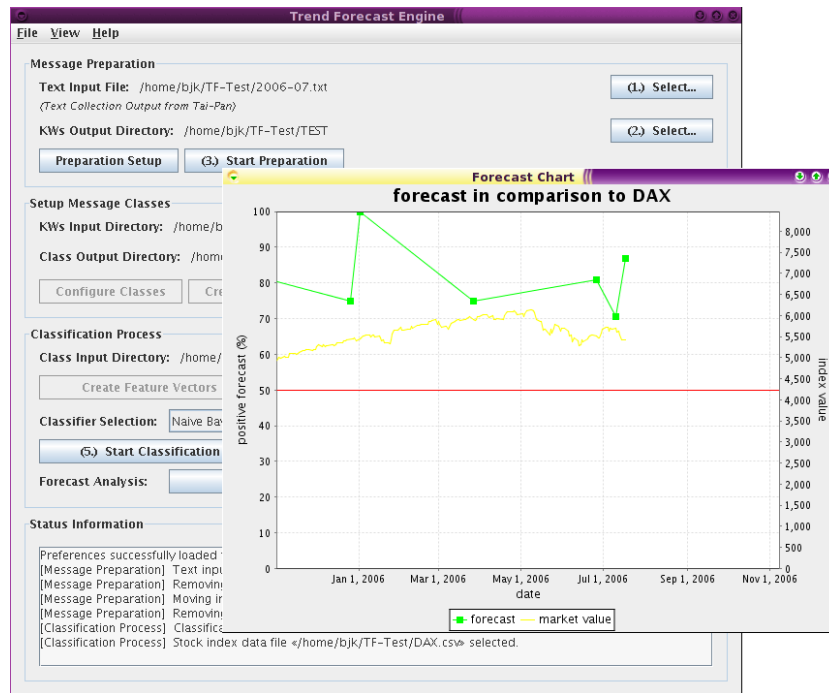


Abbildung 3.13: Trend Forecast Engine: Auswahl der Kursdaten, zu denen die Klassifikationsergebnisse dargestellt werden sollen



**Abbildung 3.14:** Trend Forecast Engine: Erkenntnis, dass in der letzte Woche der Trend nach oben ging

(eine Klassifikation für eine Woche, für die bereits ein Klassifikationsergebnis vorliegt, wird überschrieben.)

Abbildung 3.13 zeigt das Dialogfenster zum Auswählen von Dateien/Ordern JFileChooser (aus der Bibliothek javax.swing), welches im separaten Programm Anwendung findet.

## 3.6 WEBIS-Server-Plugin

Die Manifestdatei, durch die dem WEBIS-Kern das Trendvorhersage-Plug-in „bekannt gemacht“ wird ist in Codelistung 3.2 dargestellt.

```
<?xml version="1.0"?>
<plugin name="de.tu_chemnitz.webis.trendforecast" version="1.0">
  <bind-pluginSocket
    class="de.tu_chemnitz.webis.trendforecast.ClassifierServer"
```

```

    name="de.tu_chemnitz.webis.core.RemotePluginSocket" />

<classpath name="commons-collections-3.2.jar" />
<classpath name="commons-configuration-1.2.jar" />
<classpath name="commons-lang-2.1.jar" />
<classpath name="commons-logging-1.1.jar" />
<classpath name="mallet-deps.jar" />
<classpath name="mallet.jar" />

</plugin>

```

*Listing 3.2: Manifestdatei plugin\_server.xml*

Mit `de.tu_chemnitz.webis.trendforecast.ClassifierServer` wird die Serverinstanz als Remote-Plug-in gebunden, deren Methoden dem Client durch eine Stub-Datei bekannt sind.

## 3.7 WEBIS-Client-Plug-in

Die Manifestdatei, durch die dem WEBIS-Client das Trendvorhersage-Client-Plug-in „bekannt gemacht“ wird ist in Codelisting 3.3 dargestellt.

```

<?xml version="1.0"?>

<plugin name="de.tu_chemnitz.webis.trendforecast" version="1.0">

  <bind-pluginSocket
    class=""
    name="de.tu_chemnitz.webis.gui.components.navigation.
      NavigationPanelPluginSocket" />

  <navigationNode id="User Interface" controlledByParent="false"
    path="/Trend Forecast"
    doubleClickAction="de.tu_chemnitz.webis.gui.trendforecast.
      WebisMainWindow"/>

  <dependency name="de.tu_chemnitz.webis.gui.components.navigation"
    version="1.0" />
  <dependency name="de.tu_chemnitz.webis.trendforecast" version="1.0"
    type="remote" />

```

```

<classpath name="jcommon-1.0.0.jar" />
<classpath name="jfreechart-1.0.1.jar" />
<classpath name="org-jdesktop-layout.jar" />

</plugin>

```

Listing 3.3: Manifestdatei `plugin_client.xml`

### 3.7.1 RemoteFileDialog – Eigenimplementierung zum Wählen von Dateien/Ordernern auf dem WEBIS-RMI-Server

Der Versuch der Verwendung von `JFileChooser` im WEBIS-Client-Plug-in zum Wählen von Dateien bzw. Ordnern auf dem Server wurde nach zähen und langwierigen Versuchen aufgegeben. Obwohl sowohl verschiedene Methodenüberladungsmöglichkeiten existieren als auch z. B. ein eigener `FileFilter` festgelegt werden kann, ist es nicht gelungen alle Aufrufe über RMI zu realisieren, so dass also jede Dateiinformationsanfrage an den Server geschickt wird. Innerhalb der Swing-Bibliothek gibt es gehäuft direkte `File`-Aufrufe an den Filtern bzw. den überladenen Methoden vorbei, die dann zu einem gemischten Datenwirrwarr führen. So konnte man z. B. zwar die Dateinamen vom Server aufgelistet sehen, das Anklicken warf aber eine `Exception`, weil dann die Dateiinformationen wieder lokal versucht wurden zu bekommen. Von Problemen, wenn Client und Server auf unterschiedlichen Dateisystemen laufen, gar nicht erst zu reden.

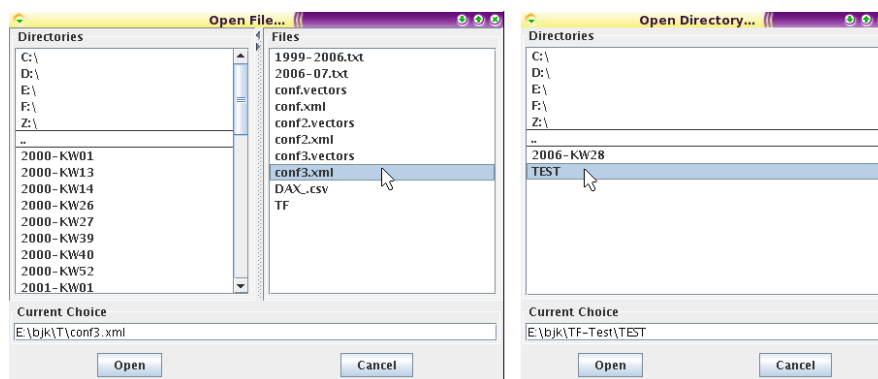
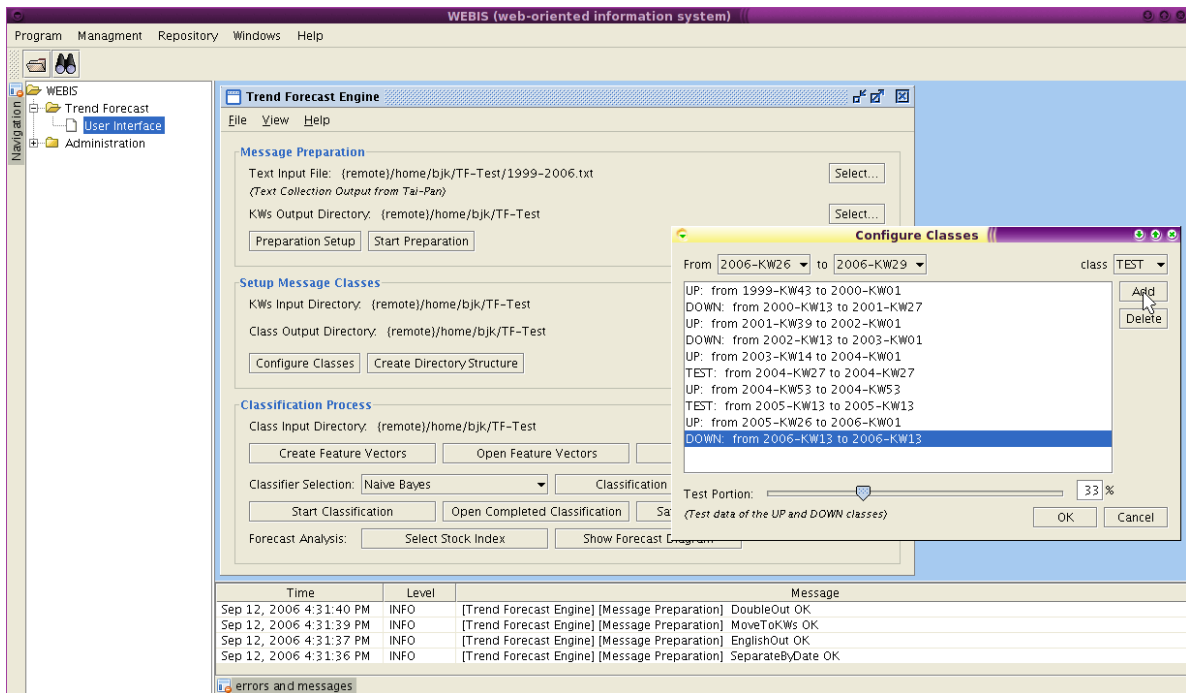


Abbildung 3.15: Datei öffnen / Verzeichnis öffnen (Zugriff auf Server, der unter Microsoft Windows läuft, im Beispiel der Client hingegen im X11-System unter Linux)

Die sowieso bereits geschriebenen Entfernte-Methoden-Aufrufe an die Server-RMI-Schnittstelle (und dort an die `File`-Bibliothek) wurden um weitere notwendige Rufe erweitert und ein simples zweigeteiltes Design (bei Verzeichnisauswahl nur noch einspaltig) entworfen, um diesen Auswahldialog zu ermöglichen. Auch wenn diese Funktionalität momentan in



**Abbildung 3.16:** Trend Forecast Engine im WEBIS: Der Logger wird für die Statusmeldungen genutzt.

die Plug-in-Schnittstelle dieser Vorhersagesoftware integriert ist, sollte es keine großen Schwierigkeiten machen, sie an anderer Stelle im WEBIS wiederzuverwenden. Abbildung 3.15 zeigt die beiden möglichen Dialogvarianten. Die Wechsellmöglichkeit von Laufwerken z. B. auf Microsoft-Windows-Systemen wurde als oberste Einträge in der Verzeichnisliste aufgenommen und erscheint natürlich auch auf Clienten, die selbst mit einem ganz anderen Dateisystem arbeiten – es wird ja eine Datei bzw. ein Verzeichnis auf dem Server gewählt.

### 3.7.2 Look and Feel als WEBIS-Client-Plug-in

Die Abbildungen 3.16–3.18 zeigen, wie sich die entwickelte Vorhersagesoftware in WEBIS darstellt.

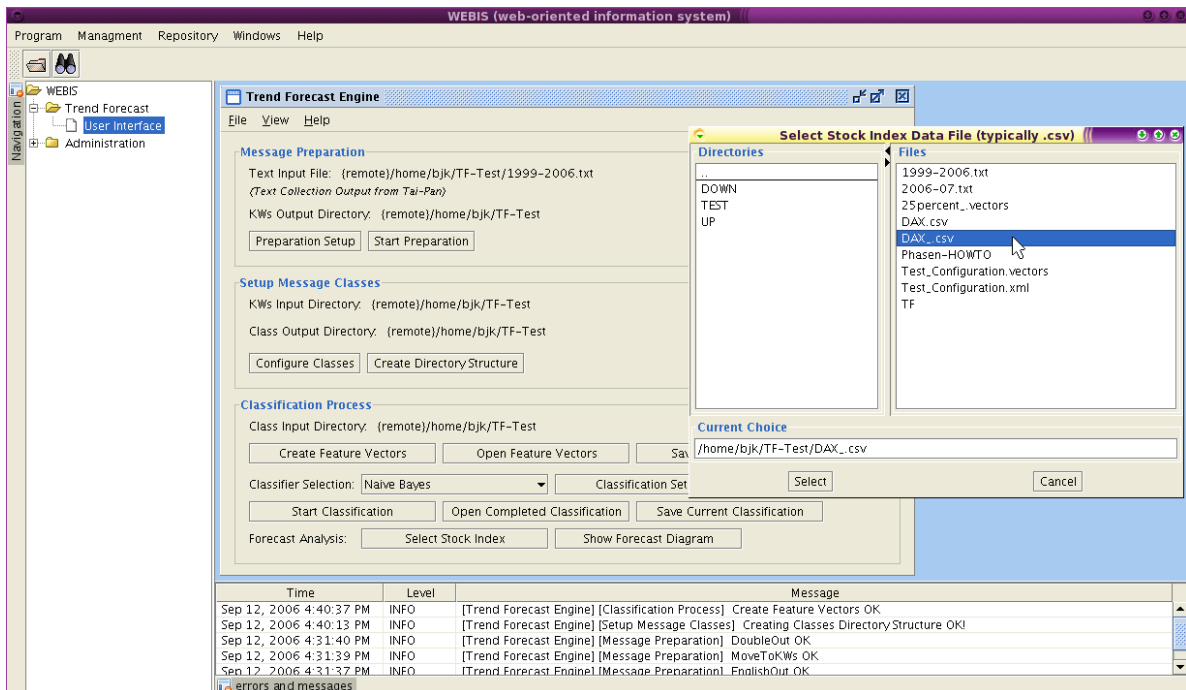


Abbildung 3.17: Trend Forecast Engine im WEBIS: Öffnen einer CSV-Datei

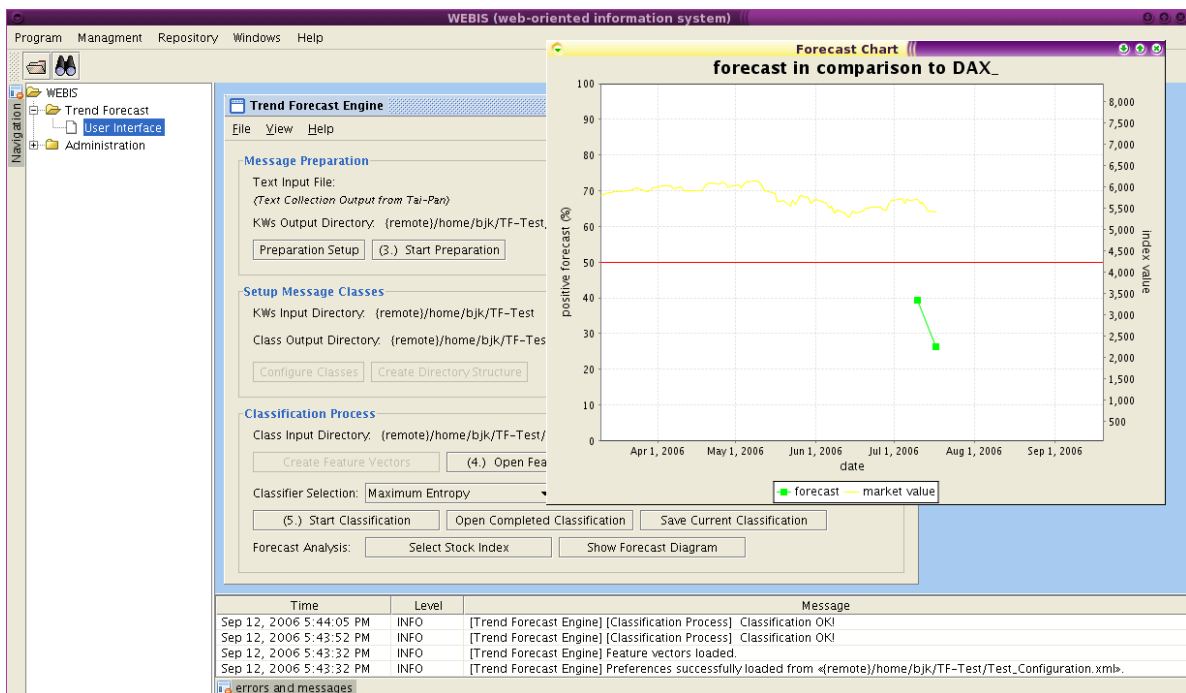


Abbildung 3.18: Trend Forecast Engine im WEBIS: Grafische Auswertung

# 4 Klassifikationsergebnisse

## 4.1 Festlegen der Nachrichtenklassen und der zu untersuchenden Auffälligkeiten

Von insgesamt 639.844 Nachrichten aus einem Zeitraum von fast 7 Jahren, die aus dem Tai-Pan-System exportiert sind, wurden 441.890 vorausgewählt, in Tabelle 4.1 sind die ausgewählten Kategorien angegeben.

Basierend auf den bereits in [Kre06] verwandten Phasendefinitionen wurde die Trainingsphasen mit den nun verfügbaren Textdaten für den Zeitraum vom 21.10.1999 bis zum 17.07.2006 wie in Tabelle 4.2 angegeben festgelegt. In der Abbildung 4.1 sind sowohl die Kursverläufe der vier größten deutschen Aktienindizes (DAX, MDAX, SDAX und TecDAX) für den Testzeitraum zu sehen als auch die festgelegten Grenze. Die Bereiche, die mit einem ? versehen sind, wurden als reine Testphasen deklariert. Um in den Klassifikationsgraphen gut erkennen zu können, welche Zeiträume dies betrifft, wurde diese spezielle Kennzeichnung gewählt.

Alle Vorverarbeitungs- und Klassifikationsläufe wurden auf dem CPU-Server `hercules.hrz` des URZ durchgeführt. Dieser Rechner hat 4 Intel-Xeon<sup>TM</sup>-Prozessoren mit je 2,40 GHz, von denen durch Starten von 2 Instanzen teilweise 2 Prozessoren gleichzeitig genutzt wurden. Der Server hat einen Arbeitsspeicher von 5,87 GiB, von denen pro gestarteter Instanz bis zu 2 GiB genutzt wurden. Als Betriebssystem stand Linux mit Kernel Version 2.6 zur Verfügung und als JRE Sun Java 2 Version 1.5.

Folgende Klassifikationsläufe wurden festgelegt und durchgeführt, der verwendete Testanteil der Trainingsklassen *UP* und *DOWN* war stets 12 %:

- (1): Um 25 % reduzierte Feature Vectors für komplette *UP*- und *DOWN*-Klassen erzeugt, mit unreduzierten Testklassen (inklusive *UP*- und *DOWN*-Testanteil, der vorher trainiert wurde) mit Naive Bayes getestet
- (2): Um 25 % reduzierte Feature Vectors für komplette *UP*- und *DOWN*-Klassen erzeugt, mit um 25 % reduzierten Testklassen (inklusive *UP*- und *DOWN*-Testanteil, der vorher trainiert wurde) mit Naive Bayes getestet

x	Ad Hoc	13.437 Nachrichten
x	Allgemeines	231.407 Nachrichten
	Edelmetalle	145 Nachrichten
	Empfehlungen und Tips	74.672 Nachrichten
	Meinungen	2.471 Nachrichten
x	Neuer Markt	12.401 Nachrichten
x	Rohstoffe	3.714 Nachrichten
	Unternehmen Ausland	94.659 Nachrichten
x	Unternehmen Inland	135.299 Nachrichten
	Wirtschaft Ausland	24.262 Nachrichten
x	Wirtschaft Inland	11.442 Nachrichten
x	Zinsen und Devisen	34.190 Nachrichten

**Tabelle 4.1:** Texteingabedaten reduziert auf diese Kategorien (Weitere Kategorien (z. B. UPDATE AKTUELL oder HOTLINE) werden stets herausgefiltert.)

21.10.1999 - 07.03.2000 42. KW 1999 - 9. KW 2000	Up	20 Wochen
08.03.2000 - 21.09.2001 10. KW 2000 - 38. KW 2001	Down	81 Wochen
22.09.2001 - 19.03.2002 39. KW 2001 - 11. KW 2002	Up	25 Wochen
20.03.2002 - 12.03.2003 12. KW 2002 - 10. KW 2003	Down	51 Wochen
13.03.2003 - 23.01.2004 11. KW 2003 - 4. KW 2004	Up	46 Wochen
08.03.2004 - 24.03.2004 11. KW 2004 - 12. KW 2004	Down	2 Wochen
13.08.2004 - 03.01.2005 34. KW 2004 - 53. KW 2004	Up	20 Wochen
11.05.2005 - 27.02.2006 19. KW 2005 - 8. KW 2006	Up	42 Wochen
09.05.2006 - 13.06.2006 19. KW 2006 - 24. KW 2006	Down	6 Wochen
<b>Up</b>		<b>153 Wochen</b>
<b>Down</b>		<b>140 Wochen</b>

**Tabelle 4.2:** Definierte Up- und Downphasen für den Testzeitraum

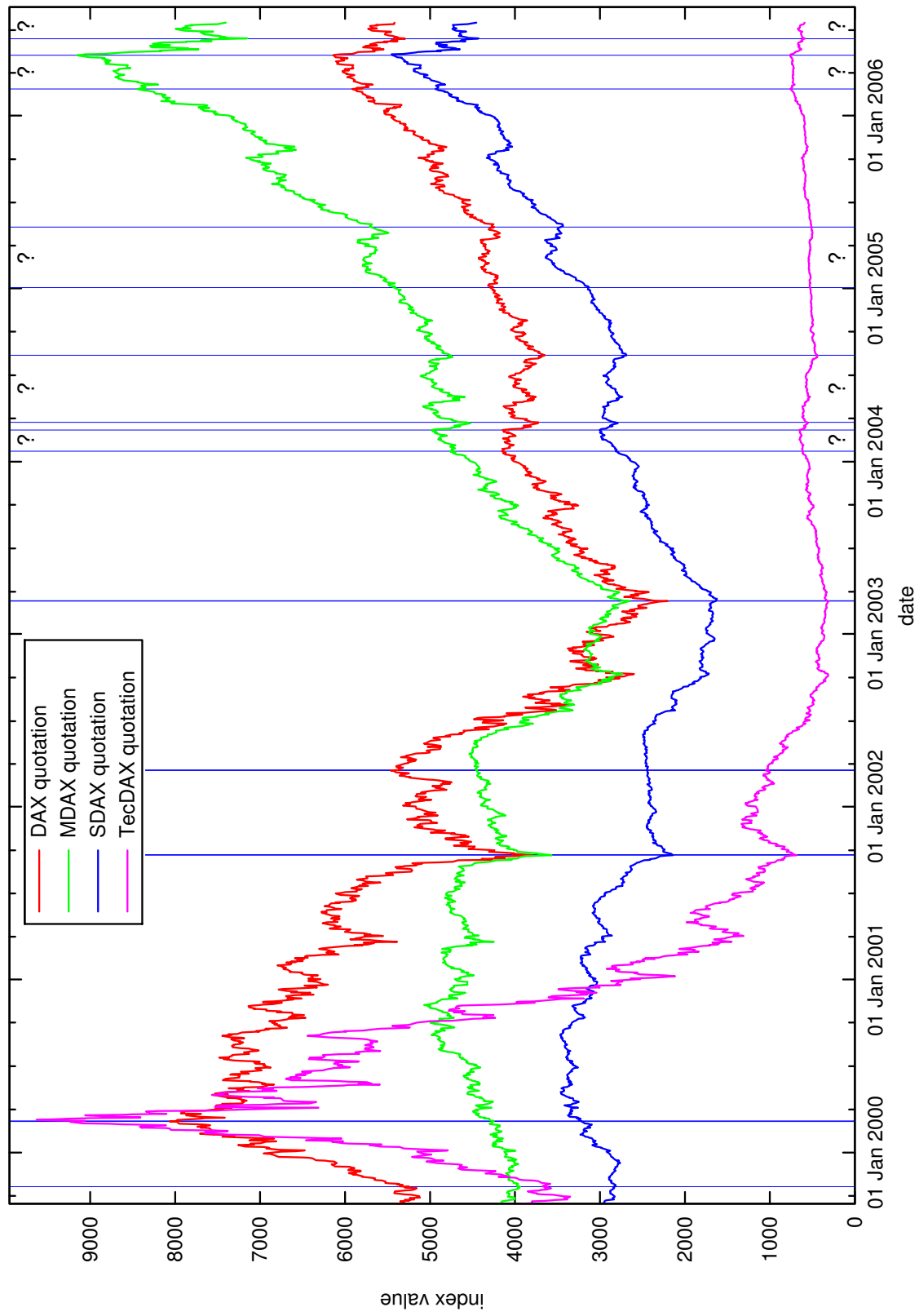


Abbildung 4.1: Definierte Phasen und der Kursverlauf der verschiedenen DAX-Werte für den Testzeitraum

- (3): Um 50 % reduzierte Feature Vectors für komplette UP- und DOWN-Klassen erzeugt, mit unreduzierten Testklassen (inklusive UP- und DOWN-Testanteil, der vorher trainiert wurde) mit Naive Bayes getestet
- (4): Um 50 % reduzierte Feature Vectors für komplette UP- und DOWN-Klassen erzeugt, mit um 50 % reduzierten Testklassen (inklusive UP- und DOWN-Testanteil, der vorher trainiert wurde) mit Naive Bayes getestet
- (5): Um 75 % reduzierte Feature Vectors für komplette UP- und DOWN-Klassen erzeugt, mit unreduzierten Testklassen (inklusive UP- und DOWN-Testanteil, der vorher trainiert wurde) mit Naive Bayes getestet
- (6): Um 75 % reduzierte Feature Vectors für komplette UP- und DOWN-Klassen erzeugt, mit um 75 % reduzierten Testklassen (inklusive UP- und DOWN-Testanteil, der vorher trainiert wurde) mit Naive Bayes getestet
- (7): Um 25 % reduzierte Feature Vectors erzeugt, mit unreduzierten Testklassen mit Naive Bayes getestet
- (8): Um 25 % reduzierte Feature Vectors erzeugt, mit um 25 % reduzierten Testklassen mit Naive Bayes getestet
- (9): Um 50 % reduzierte Feature Vectors erzeugt, mit unreduzierten Testklassen mit Naive Bayes getestet
- (10): Um 50 % reduzierte Feature Vectors erzeugt, mit um 50 % reduzierten Testklassen mit Naive Bayes getestet
- (11): Um 75 % reduzierte Feature Vectors erzeugt, mit unreduzierten Testklassen mit Naive Bayes getestet
- (12): Um 75 % reduzierte Feature Vectors erzeugt, mit um 75 % reduzierten mit Naive Bayes Testklassen getestet
- (13): Um 25 % reduzierte Feature Vectors erzeugt, mit um 50 % reduzierten Testklassen mit Naive Bayes getestet
- (14): Um 25 % reduzierte Feature Vectors erzeugt, mit um 75 % reduzierten Testklassen mit Naive Bayes getestet
- (15): Um 75 % reduzierte Feature Vectors erzeugt, mit unreduzierten Testklassen mit Maximum Entropy getestet
- (16): Um 75 % reduzierte Feature Vectors erzeugt, mit unreduzierten Testklassen mit Decision Tree getestet
- (17): Um 75 % reduzierte Feature Vectors erzeugt, mit unreduzierten Testklassen mit Balanced Winnow getestet

Angemerkt sei, dass für die 3 anderen Classifier nur die stark reduzierten Feature Vectors genommen wurden, weil sonst der Rechenzeit- und Arbeitsspeicheraufwand nicht mehr im Rahmen gewesen wäre. In den nachfolgend aufgeführten Abbildungen sind die Klassifikationsgraphen jeweils leicht geglättet dargestellt, da sonst keine sinnvolle Vergleichsvisualisierung möglich gewesen wäre.

Aus den Abbildungen 4.2, 4.3 und 4.4 geht hervor, dass die Klassifikation mit unreduzierten Testvektoren für den Fall, dass die Testvektoren der festgelegten Phasen vorher auch Trainingsvektoren waren, sich kaum von der Klassifikation unterscheidet, bei der die Testvektoren wirklich Testvektoren sind. Werden die Testvektoren jedoch im gleichen Maße wie die Trainingsvektoren reduziert, sind Abweichungen in der Klassifikationen speziell bei starker Feature Reduction zu sehen.

Abbildung 4.5 zeigt, dass sich der Naive-Bayes-Klassifikationsverlauf erst aber bei 75% reduzierten Trainingsvektoren anders präsentiert. Dies wird in den Abbildungen 4.6 und 4.7 bei Verwendung von ebenfalls reduzierten Testvektoren bestätigt.

In Abbildung 4.8 ist der Klassifikationsverlauf für die im entwickelten System zur Verfügung stehenden vier Classifier dargestellt. Das Entscheidungsbaumverfahren (*Decision Tree*) fällt dabei negativ auf – die Klassifikation ist für eine Verwendung von einer so großen Eingabedatenmenge nicht sinnvoll. Das Maximum-Entropie- und das Balanced-Winnnow-Verfahren verhalten sich dagegen eher wie erwartet und speziell das Balanced-Winnnow-Verfahren zeigt einen sehr überzeugenden Verlauf in den Testphasen mit bekannter Klassifikation, was evtl. auch ein Hinweis auf Übertrainiertheit sein könnte.

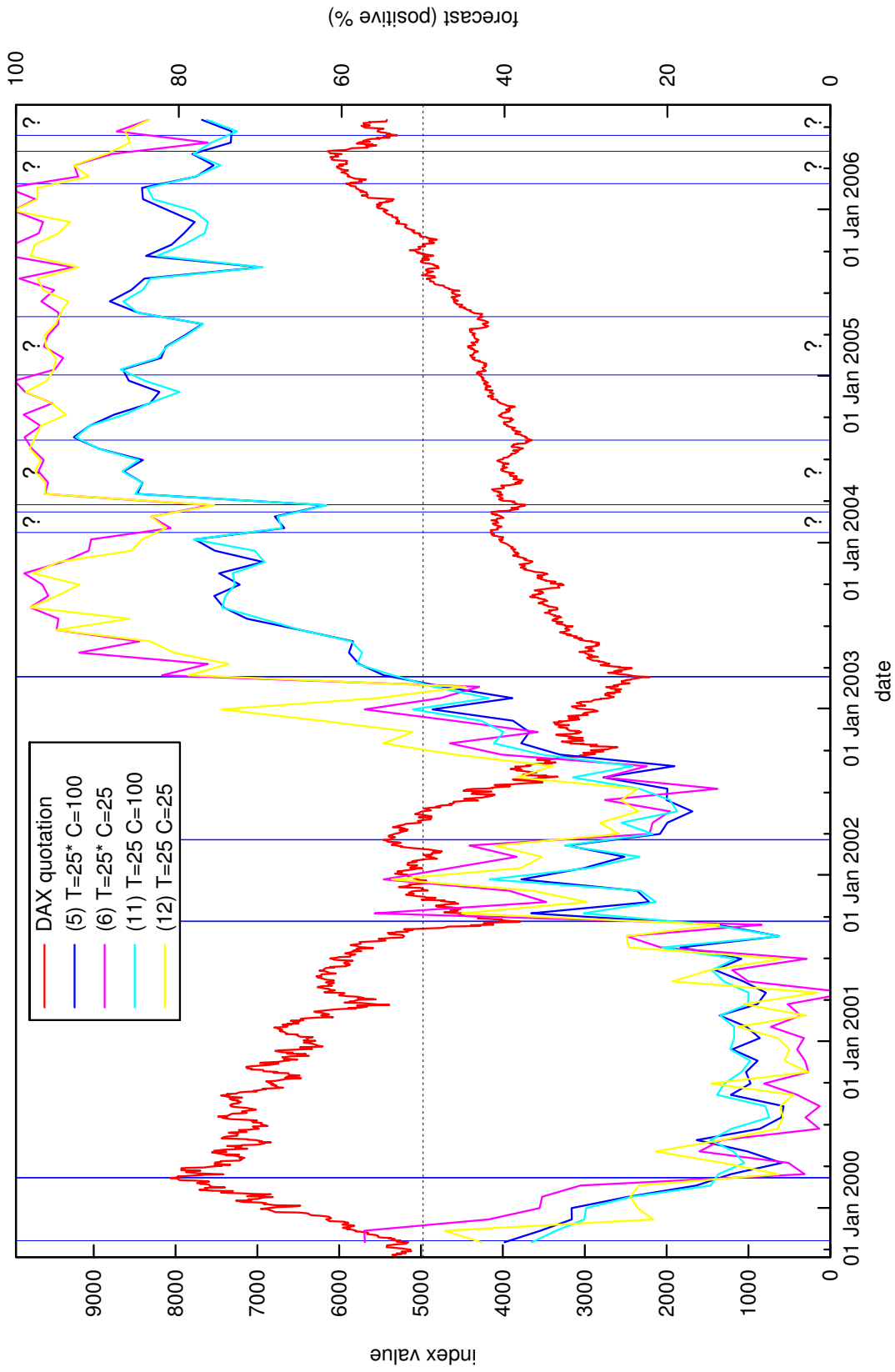


Abbildung 4.2: Alle Naive-Bayes-Klassifikationen mit um 75 % reduzierten Feature Vectors

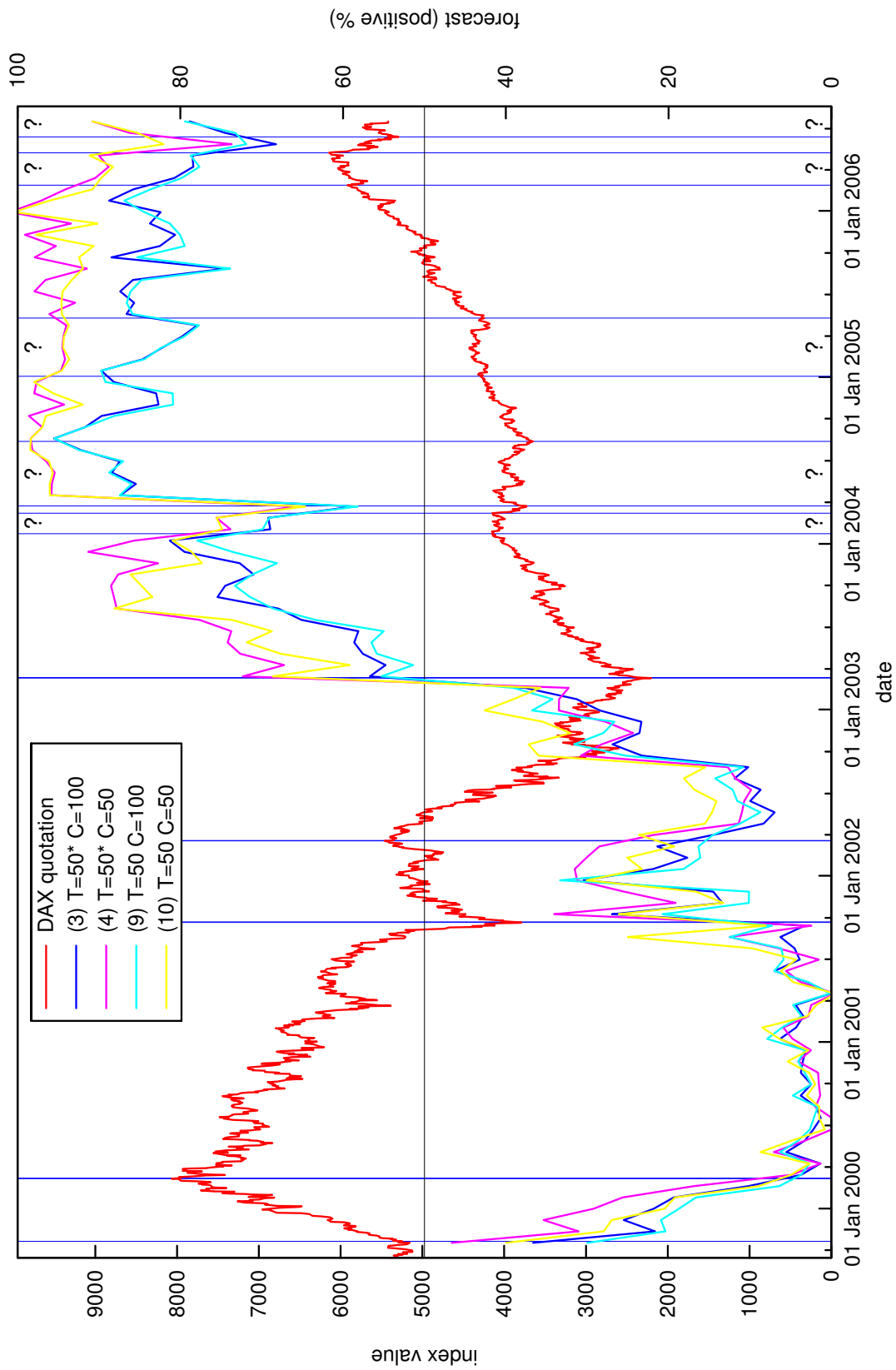


Abbildung 4.3: Alle Naive-Bayes-Klassifikationen mit um 50 % reduzierten Feature Vectors

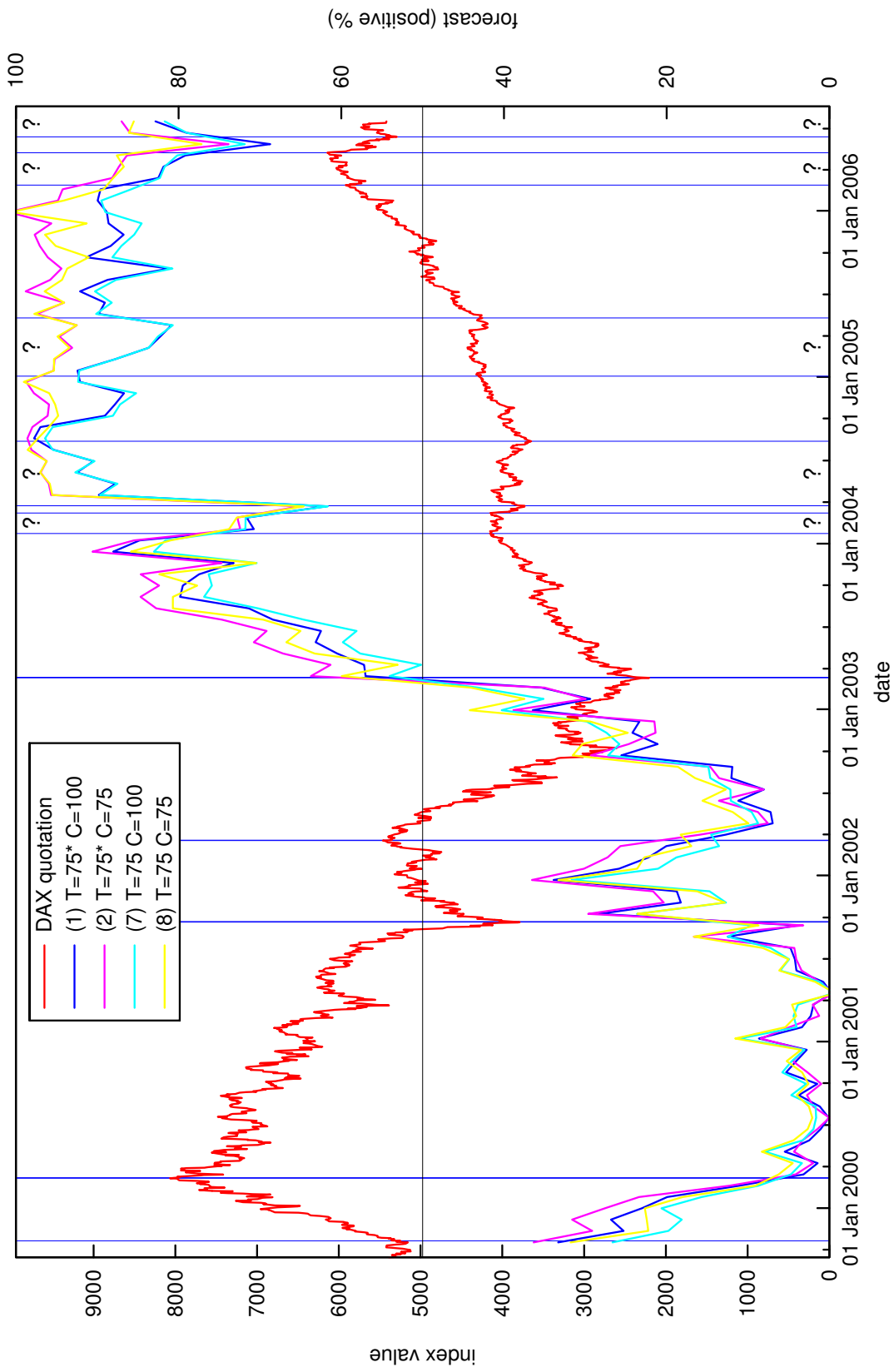


Abbildung 4.4: Alle Naive-Bayes-Klassifikationen mit um 25 % reduzierten Feature Vectors

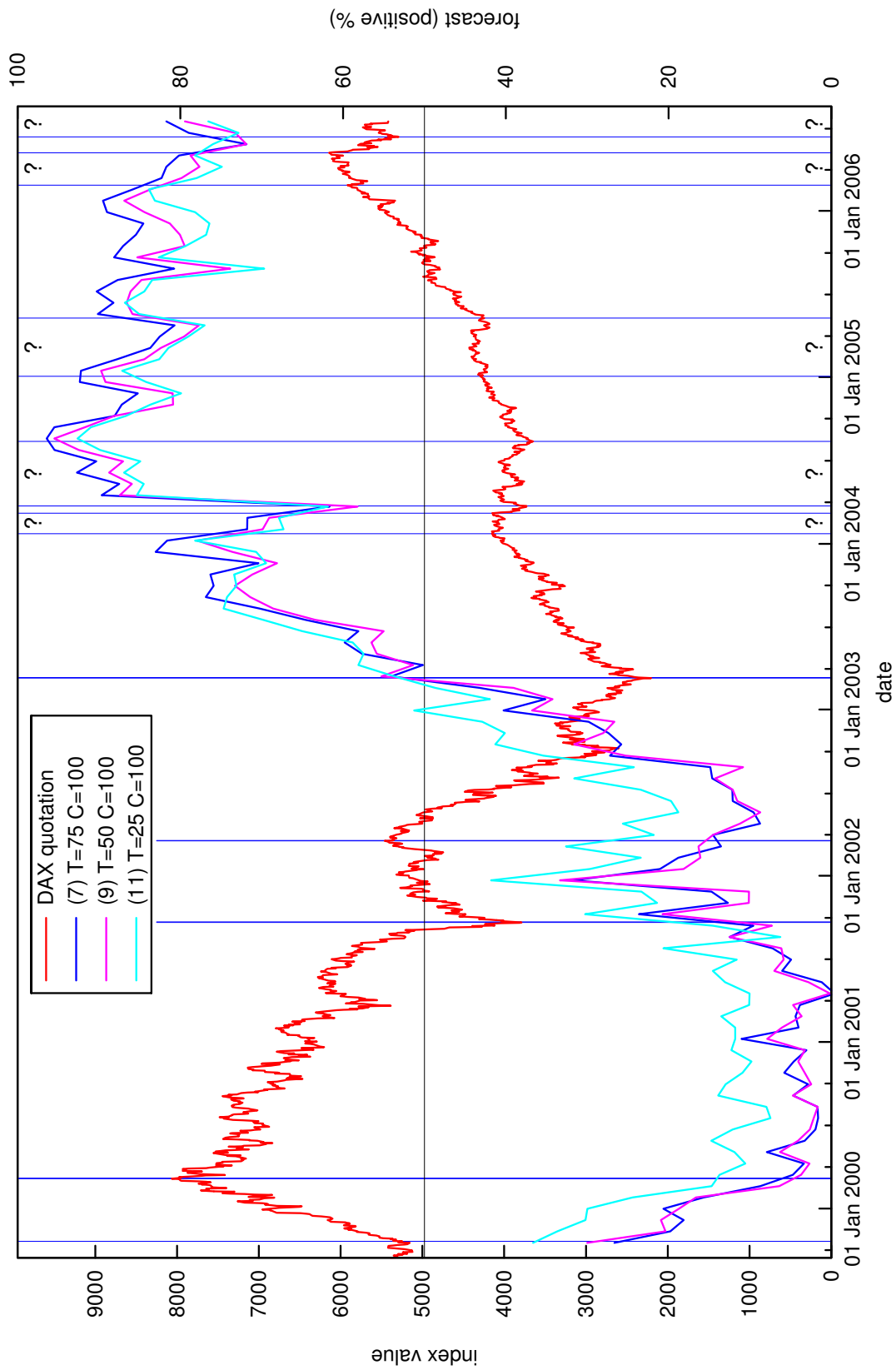
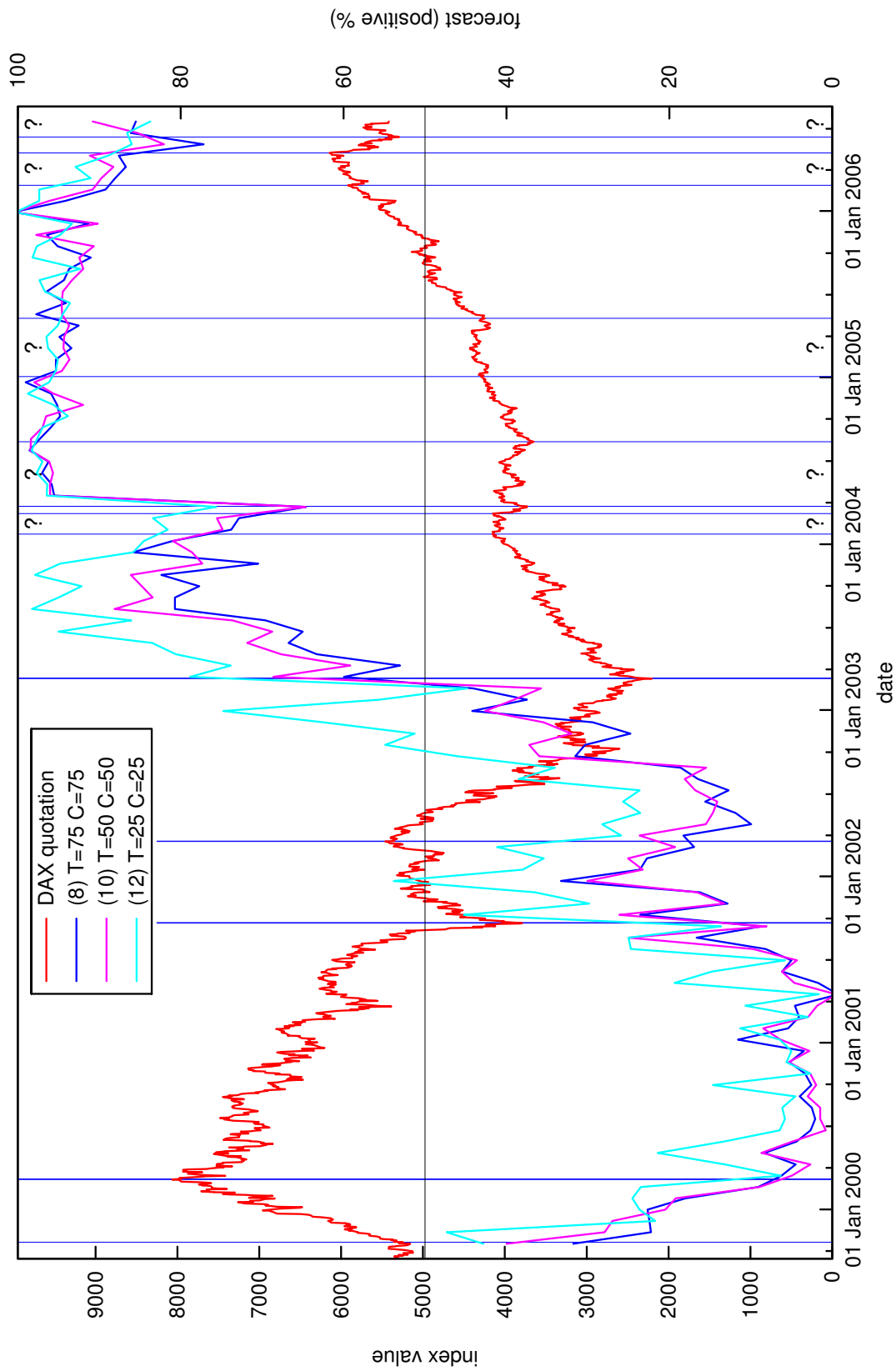
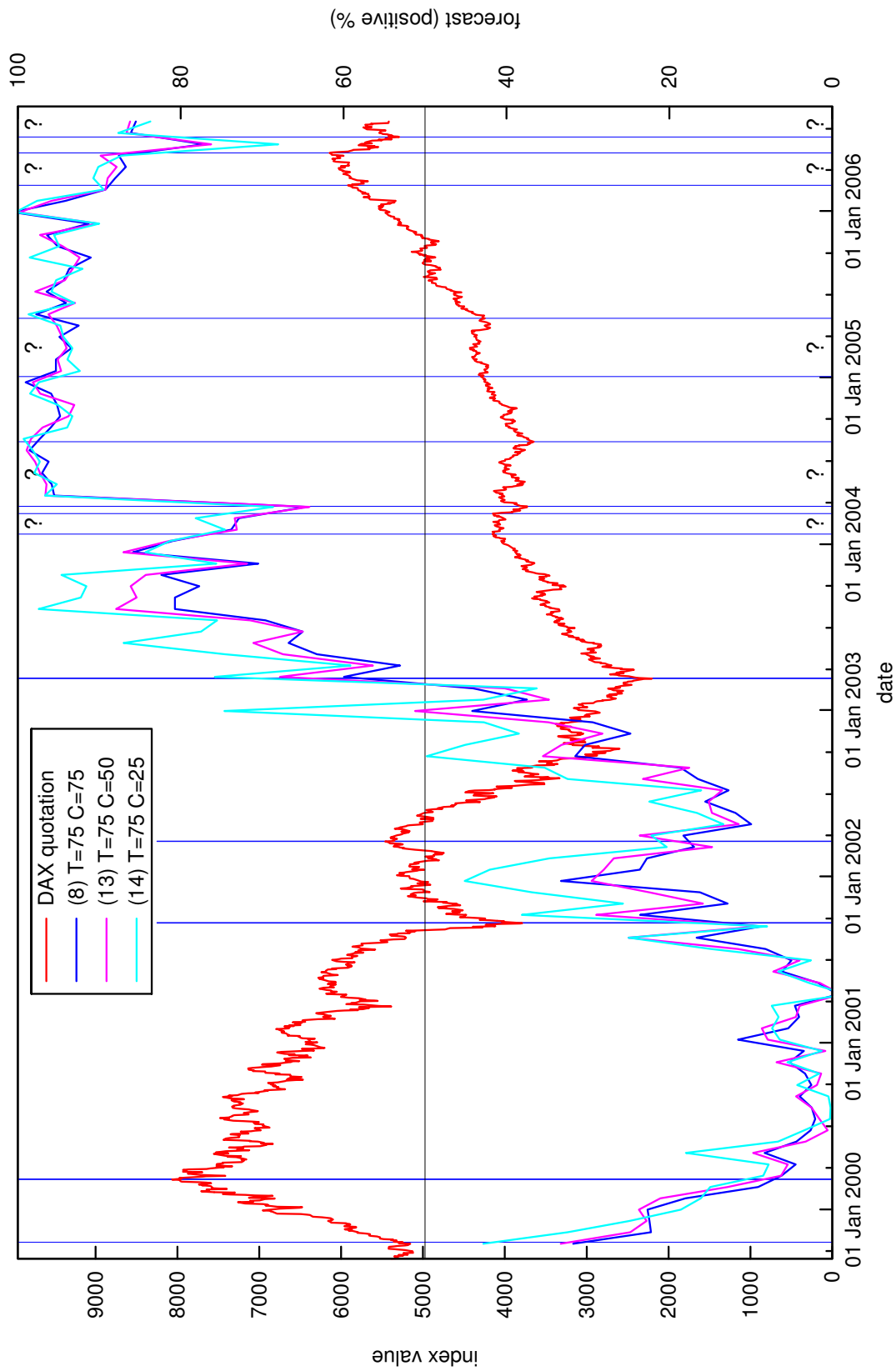


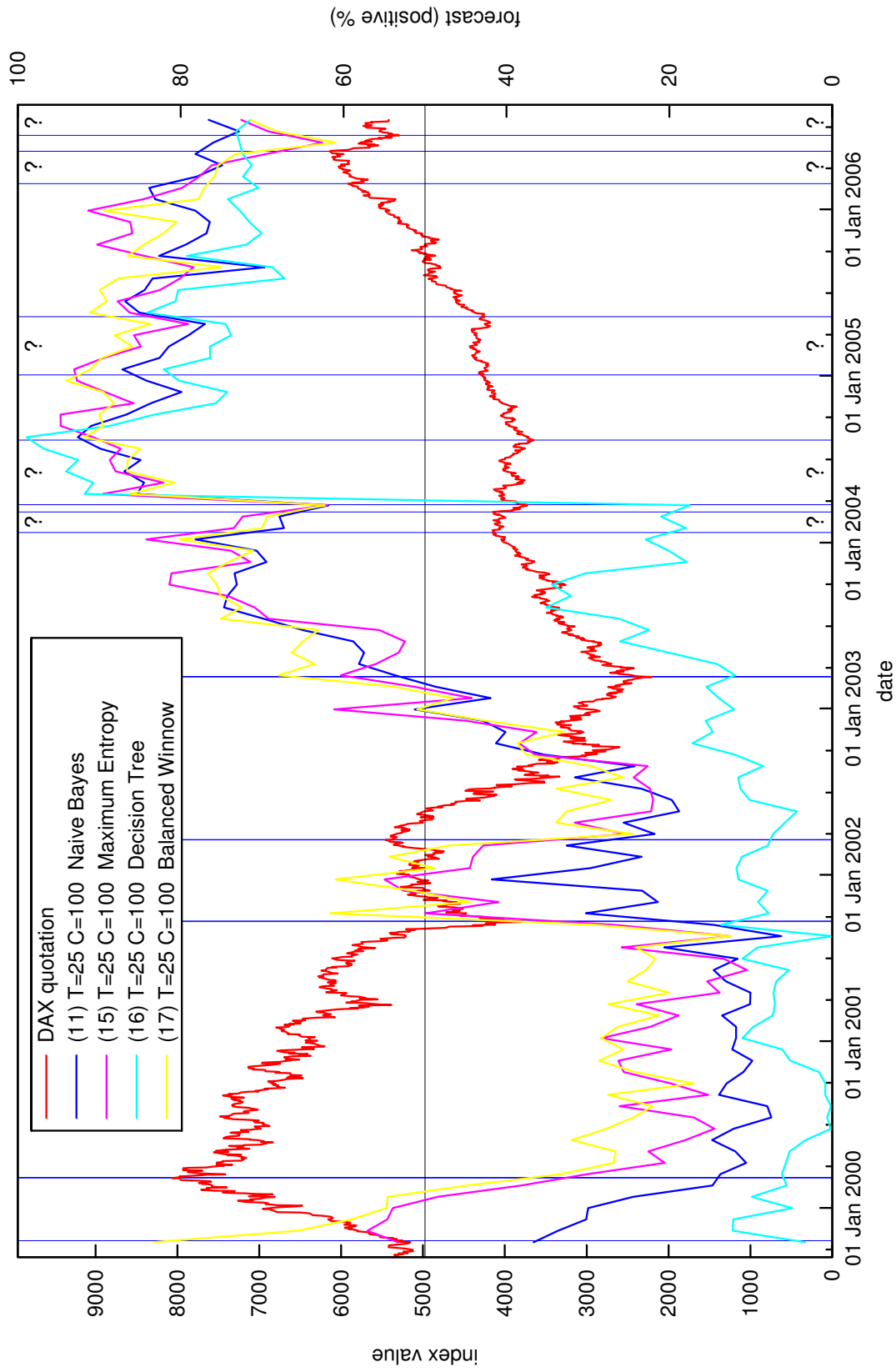
Abbildung 4.5: Alle Naive-Bayes-Klassifikationen mit unreduzierten Testvektoren



**Abbildung 4.6:** Alle Naive-Bayes-Klassifikationen, bei denen die Trainings- und die Testvektoren im gleichen Maße reduziert wurden



**Abbildung 4.7:** Alle Naive-Bayes-Klassifikationen mit um 25 % reduzierten Trainingsvektoren und bis zu 75 % reduzierten Testvektoren



**Abbildung 4.8:** Klassifikationsergebnisse von 4 verschiedenen Classifiern für jeweils um 75 % reduzierte Trainingsvektoren und unreduzierten Testvektoren

# 5 Vergleich mit bisherigen Ergebnissen

## 5.1 Vergleich mit Klassifikationsergebnissen mit einer $ll(k)$ -Grammatik [Rei06]

Thomas Reichel hat englischsprachige kurze e-Mail-Börsenmeldungen (Zeitraum von etwa 5 Jahren, Februar 2001 bis November 2005) als Testdaten für die Trendvorhersage durch grammatikalische Analyse verwendet. Die Daten der 250 Wochen wurden für das Testen in 94 Wochen *UP*, 69 Wochen *DOWN* und 87 Wochen *TEST* aufgeteilt (Abbildung 5.1).

Die Dateilistings 5.1–5.6 zeigen beispielhaft die bereits entsprechend vorverarbeitete Beispielnachrichten, die direkt in Klassifikatorklassen eingeordnet werden konnten:

```
Amgen is in discussions to buy Immunex for about $17 billion in cash
and stock, in a deal that would unite the world's largest
biotechnology concern with a smaller but fast-growing rival fueled
by a blockbuster rheumatoid-arthritis treatment.
```

*Listing 5.1: 2001.12.14-00.00.00-01.txt*

```
Telecommunications, technology and consumer stocks, which led the U.
S. market's rebound during the past three months, stumbled Thursday,
amid new earnings worries. The losses dragged down the broader
market and raised new concerns about the direction of stocks.
```

*Listing 5.2: 2001.12.14-00.00.00-11.txt*

```
Investors are filing arbitration claims against their brokers in
record numbers in the wake of the bull market's collapse. The amount
being awarded to investors is soaring and the size of the disputes
has risen.
```

*Listing 5.3: 2003.05.27-00.00.00-01.txt*



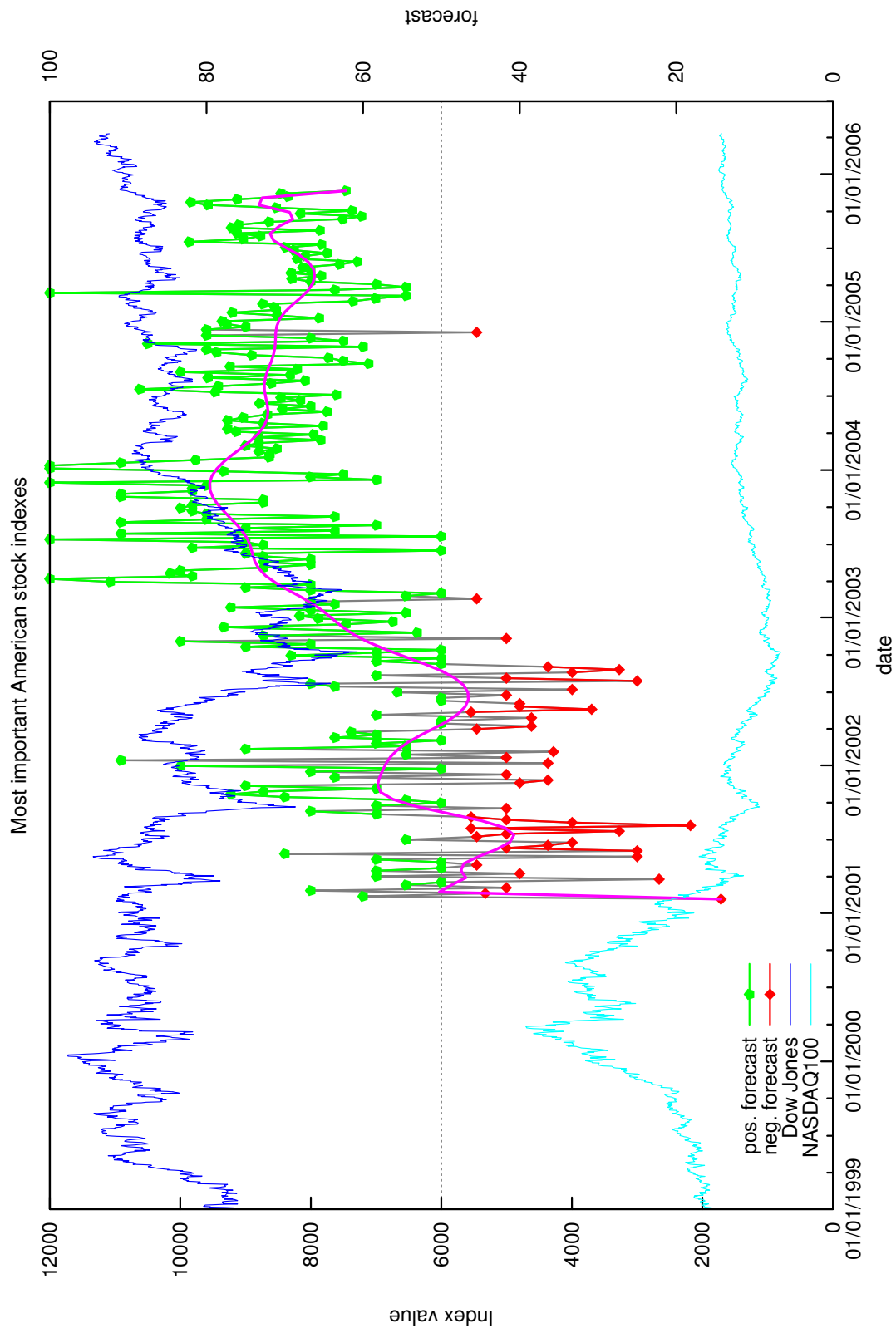


Abbildung 5.2: Ungeglätteter und geglätteter Naive-Bayes-Klassifikationsverlauf für die englischen Eingabedaten

```
With more restaurants shutting their doors and consumers looking for
ways to tighten their belts, the industry is grappling with its
most-prolonged downturn in over two decades — and hard-hit eateries
are trying a range of tactics to stay afloat.
```

*Listing 5.4: 2003.05.27-00.00.00-07.txt*

```
The Fed raised short-term rates a quarter point to 4% and suggested
that it will continue with increases until the economy slows. The
central bank indicated optimism about economic growth but voiced
inflation concerns.
```

*Listing 5.5: 2005.11.02-03.05.00-00.txt*

```
Microsoft is creating online services coupled with its two major
software lines, Windows and Office, in a move to help the software
maker tap into the swelling market for Internet advertising.
```

*Listing 5.6: 2005.11.02-03.05.00-08.txt*

In Abbildung 5.2 ist der konkrete und der geglättete Klassifikationsverlauf mit dem hier verwendeten Naive-Bayes-Verfahren dargestellt. Die Abbildung 5.3 zeigt dann den Verlaufsvergleich der beiden jeweils geglätteten Klassifikationsverläufe, dem von Thomas Reichel (*thomr*) und den mit der hier entwickelten Software durchgeführten (*bjk*). Die Tendenzen stimmen überraschenderweise sehr gut überein, jedoch sind die Amplituden verschieden.

## 5.2 Vergleich mit den Ergebnissen des Prototyps [Kre06]

Die Testergebnisse abschließend wurde noch ein Vergleich (Abbildung 5.4) der Klassifikationsverläufe für ja gleiche Eingabetexte aus dem Tai-Pan-System durchgeführt. Der erhoffte Effekt, dass ein Trendumschwung wie im März 2003 für den DAX geschehen einen kurzen Zeitraum vorher klassifiziert wird, wurde nicht konkret an der 50%-Schwelle wiederentdeckt. Auch die neuen Klassifikationsergebnisse steigen jedoch bereits vor Erreichen des Indextiefwertes stetig an, was ja ebenfalls eine nützliche Information sein kann, die aus den Klassifikationen gewonnen werden kann.

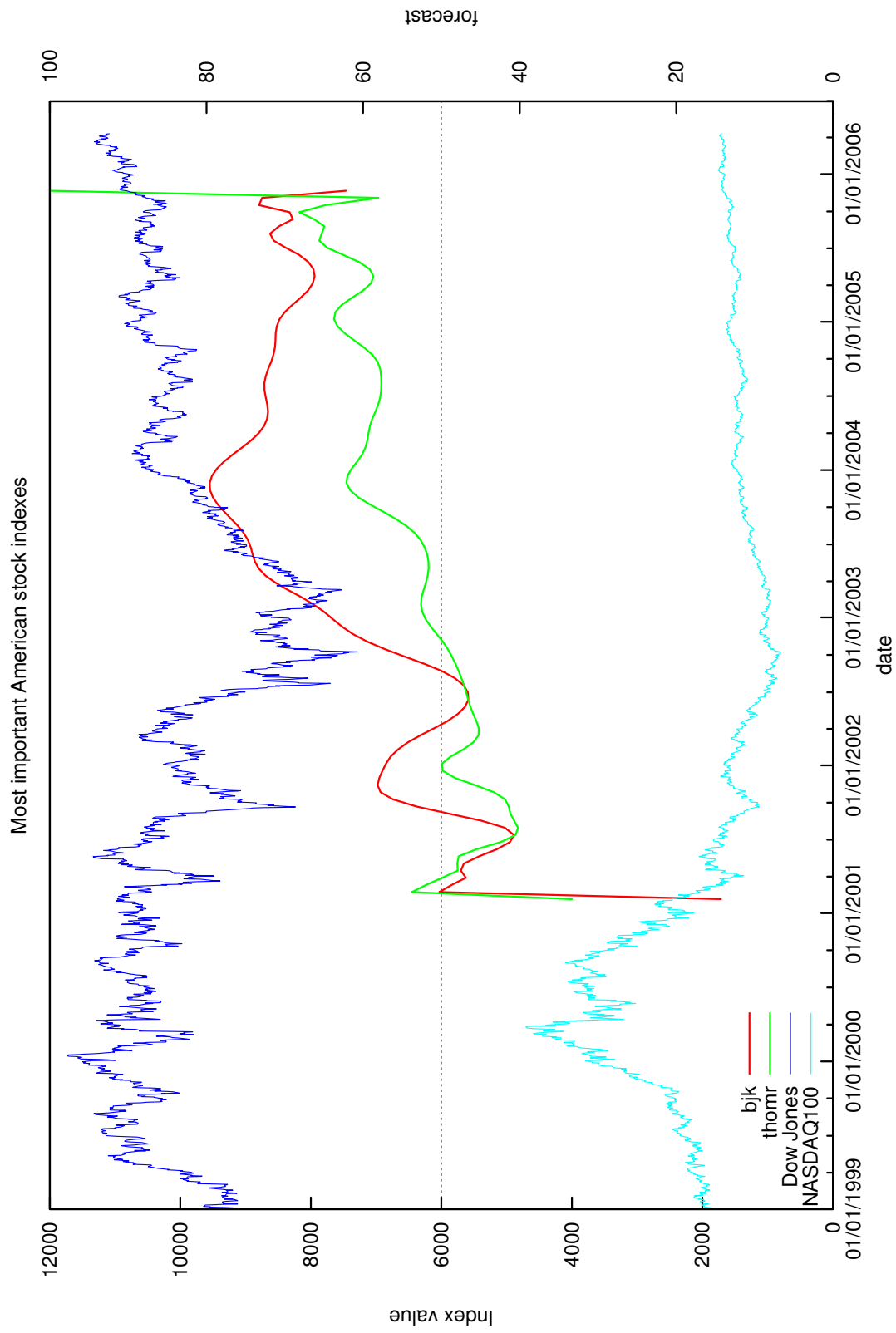


Abbildung 5.3: Klassifikationsvergleich mit den Ergebnissen aus [Rei06]

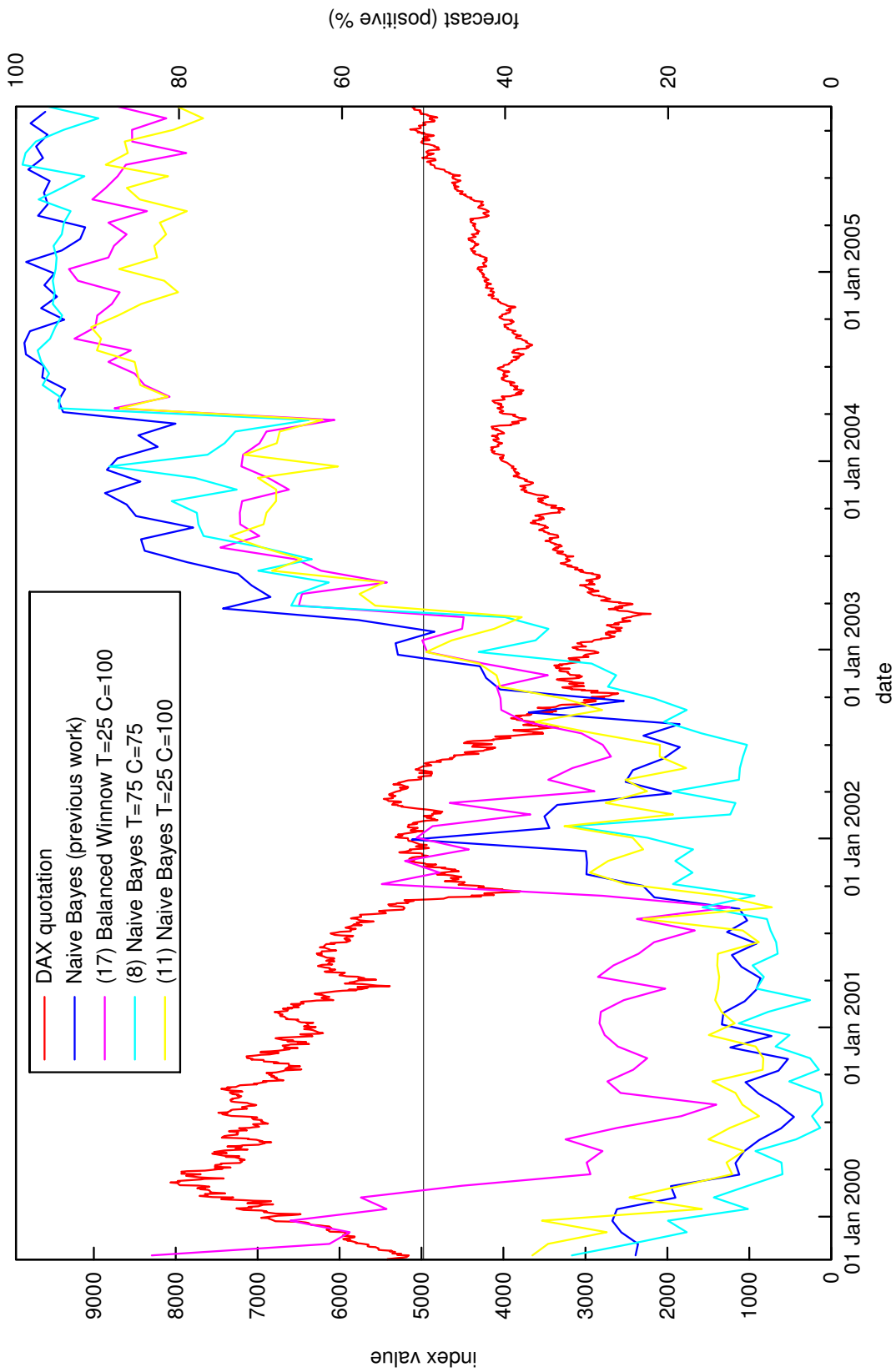


Abbildung 5.4: Klassifikationsvergleich mit den Ergebnissen aus [Kre06]

## 6 Rückblick und Ausblick

In der entwickelten Software sind noch recht viele fest programmierte Einstellungen vorhanden, die es wert wären, dem Benutzer die Auswahl vornehmen zu lassen. Genannt seien z. B. die Statusanzeige, die nicht dynamisch an die Eingabegröße angepasst ist, sondern beim Qualifizieren für alle Eingabedaten nur nach festgelegten 10 Schritten ein Statusupdate vornimmt. Beim Erzeugen der Vektoren verhindert die Schnittstelle zu Mallet ein Auslesen des aktuellen Fortschritts.

Die Auswahlmöglichkeit von 4 verschiedenen Classifiern bietet neue Möglichkeiten, eventuelle Datenzusammenhänge zu untersuchen. Das Entscheidungsbaumverfahren ist für das komplexe Training, wie es solch lange Zeiträume (1999–2006) und Datenmengen (knapp 450.000 Eingabenachrichten) erfordern nicht geeignet, wurde aber integriert, um auch für kleinere Szenarien den dann mit hoher Wahrscheinlichkeit besser arbeitenden Classifier wählen zu können. Detaileinstellmöglichkeiten zu den Classifiern und eine bessere Beeinflussung des Feature-Reduction-Verfahrens würden zwar die Komplexität des Systems erhöhen, sind aber als weitere Punkte, an denen das System dem Benutzer durchaus noch mehr Eingriff erlauben können sollte, zu nennen.

Zusammenfassend kann gesagt werden, dass die Funktionalität des Prototyps erhalten und erweitert werden konnte. Das eingeführte Verfahren zur Feature-Reduzierung konnte zeigen, dass es für bestimmte weitere Systemparameter eine verbesserte Effizienz des nachfolgenden Klassifikationsprozesses mit unverminderter Klassifikationsqualität (im Vergleich zu Klassifikationen ohne diese Eigenschaft) bietet.

Der Balanced-Winnow-Classifier ist es wert, genauer für die Nutzung für die Trendvorhersage getestet zu werden, das Naive-Bayes-Verfahren muss wohl aber dennoch nicht um seinen guten Ruf als hervorragend geeigneter Classifier bei ausgewogener Klassifikatorerstellung fürchten.

# Abkürzungsverzeichnis

- CSV ..... Comma Separated Value, oder auch: Character Separated Value.  
Textdatei zur Speicherung und zum Austausch von strukturierten Daten. Einzelne Werte sind dabei durch ein spezielles Trennzeichen (z. B. ein Komma) voneinander getrennt.
- DAX ..... Deutscher Aktienindex.  
Leitindex der deutschen Börse, welcher sich aus 30 der 35 größten deutschen börsennotierten Aktiengesellschaften zusammensetzt. Weitere wichtige deutsche Indizes sind der *MDAX*, der *SDAX* sowie der *TecDAX*.
- DJIA ..... Dow Jones Industrial Average.  
Ältester noch bestehender Aktienindex der USA, der sich heute aus den 30 größten US-Unternehmen zusammensetzt. Auch kurz *Dow Jones Index* genannt.
- GUI ..... Graphical User Interface.  
Gebräuchlicher Begriff für eine grafische (meist fensterbasierte) Benutzeroberfläche.
- HTML ..... Hypertext Markup Language.  
Textbasierte Auszeichnungssprache zur Darstellung von Text- und Medieninhalten inklusive Hyperlinks zu anderen Dokumenten. *XHTML*-Dokumente sind nach den Syntaxregeln von *XML* aufgebaute *HTML*-Dokumente.
- JAR ..... Jave Archive.  
Komprimiertes oder unkomprimiertes Archiv (.zip-Format) mit zusätzlichen Metadaten, welches vor allem zum Zusammenstellen von miteinander abhängigen Java-Klassen verwendet wird.

- JDK ..... Java Development Kit.  
Java-Entwicklungsumgebung, die den Compiler und andere Tools (u. a. die *JRE*) zum Entwickeln von Java-Applikationen enthält.
- JRE ..... Java Runtime Environment.  
Java-Laufzeitumgebung, die alle Programme und notwendigen Dateien enthält, um Java-Applikationen ausführen zu können.
- Mallet ..... A Machine Learning for Language Toolkit ([[McC02](#)]).  
Sammlung von Java-Code für die statistische Verarbeitung natürlicher Sprache, Dokumentklassifikation, Clustering, Informationsextraktion und weitere Machine-Learning-Anwendungen für Text.
- NASDAQ ..... Akronym für National Association of Securities Dealers Automated Qotations.  
Börse, die von der National Association of Securities Dealers betrieben wird. Gleichzeitig wichtiger US-amerikanischer Aktienindex.
- RMI ..... Remote Method Invocation.  
Kommunikationsprotokoll und Programmierschnittstelle der Programmiersprache Java.
- URZ ..... Universitätsrechenzentrum.  
Computerrechenzentrum einer Universität, in dieser Arbeit ist stets die Technische Universität Chemnitz gemeint.
- WEBIS ..... Weborientiertes Informationsystem.  
Informationssystem, das seine zu speichernden Eingabedaten aus dem World Wide Web bezieht und im XML-Format abspeichert. Mittels entsprechender Abfragen können aus dem Datenbestand Informationen extrahiert werden.
- WWW ..... World Wide Web.  
Weltweites über das Internet verfügbares Hypertextsystem.
- XML ..... Extensible Markup Language.  
Vom World Wide Web Consortium standardisierte Spezifikation einer Metasprache zur maschinen- und menschenlesbaren Beschreibung von Datenobjekten in Form einer Baumstruktur.

# Literaturverzeichnis

- [Ahn05] AHNERT, Frank: *Weiterentwicklung des Web-orientierten Informationssystems (WEBIS)*. Diplomarbeit, Technische Universität Chemnitz, 2005
- [BYRN99] BAEZA-YATES, Ricardo ; RIBEIRO-NETO, Berthier: *Modern Information Retrieval*. Santiago, Chile and Belo Horizonte, Brazil : ACM Press, 1999
- [CPS98] CIOS, Krzysztof ; PEDRYCZ, Witold ; SWINIARSKI, Roman: *Data Mining Methods for Knowledge Discovery*. Boston, Dordrecht, London : Kluwer Academic Publishers, 1998. – ISBN 0-7923-8252-8
- [FBG<sup>+</sup>06] FELDEN, Carsten ; BOCK, Heiko ; GRÄNING, André ; MOLOTOWA, Lana ; SAAT, Jan ; SCHÄFER, Rebecca ; SCHNEIDER, Bernhard ; STEINBORN, Jenny ; VOECKS, Jochen ; WOERLE, Christopher: Evaluation von Algorithmen zur Textklassifikation / Fakultät für Wirtschaftswissenschaften, Technische Universität Bergakademie Freiberg. 2006 (Freiberger Arbeitspapiere #10). – Forschungsbericht. [http://www.tu-freiberg.de/~wwwfak6/files/paper/2006/felden\\_10\\_2006.pdf](http://www.tu-freiberg.de/~wwwfak6/files/paper/2006/felden_10_2006.pdf)
- [Gem01] GEMEINHARDT, Lars: *Ein web-orientiertes Informationssystem (WEBIS)*. Diplomarbeit, Technische Universität Chemnitz, 2001
- [GGLL00] GÓMEZ, Manuel M. ; GELBUKH, Alexander ; LÓPEZ-LÓPEZ, Aurelio: Mining the News: Trends, Associations, and Deviations. In: *Computación y Sistemas, Vol. 5 No. 1, July-September 2001*, 2000
- [KBY04] KROHA, Petr ; BAEZA-YATES, Ricardo: Classification of Stock Exchange News – A Draft / Department of Computer Science, Engineering School, Universidad de Chile. 2004 (January 11). – Forschungsbericht
- [KGP<sup>+</sup>04] KONTOSTATHIS, April ; GALITSKY, Leon M. ; POTTENGER, William ; ROY, Soma ; PHELPS, Daniel J.: A Survey of Emerging Trend Detection in Textual Data Mining. In: BERRY, Micheal W. (Hrsg.): *Survey of Text Mining – Clustering, Classification, and Retrieval*. Springer-Verlag New York, LLC, 2004
- [Kre05] KRELLNER, Björn: *Text-Mining-Technologien zur Trendvorhersage*. Seminararbeit, Technische Universität Chemnitz, 2005

- [Kre06] KRELLNER, Björn: *Nachrichtenklassifikation unter Nutzung regulärer Ausdrücke*. Studienarbeit, Technische Universität Chemnitz, 2006
- [Kro97] KROHA, Petr: *Softwaretechnologie*. München [i. e. Haar], London, Mexiko, New York, Singapur, Sydney, Toronto : Prentice Hall, 1997. – ISBN 3–8272–9537–8
- [Kro05] KROHA, Petr: *Using Regular Expressions in Text Mining – Draft*. 2005. – Department of Computer Science, Chemnitz University of Technology
- [Lit87] LITTLESTONE, Nick: *Learning Quickly When Irrelevant Attributes Abound: A New Linear-threshold Algorithm*. Version: 1987. <http://www.cse.ucsc.edu/classes/cmcs242/Fall02/paps/winnow.ps>. – Online Ressource
- [Löf02] LÖFFLER, Dany: *Erweiterung des web-orientierten Informationssystems (WEBIS)*. Diplomarbeit, Technische Universität Chemnitz, 2002
- [McC96] MCCALLUM, Andrew K.: *Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering*. <http://www.cs.cmu.edu/~mccallum/bow/>. Version: 1996
- [McC02] MCCALLUM, Andrew K.: *MALLET: A Machine Learning for Language Toolkit*. <http://mallet.cs.umass.edu>. Version: 2002
- [Mei04] MEICHSNER, Christian: *Plug-in Subsystem Prototype*. 2004. – WEBIS project paper, Department of Computer Science, Chemnitz University of Technology
- [Mei05] MEICHSNER, Christian: *Trading System Integration into WEBIS*. Diploma Thesis, Chemnitz University of Technology, 2005
- [Obe05a] OBERHAUSER, Otto: *Automatisches Klassifizieren: Entwicklungsstand – Methodik – Anwendungsbereiche*. Frankfurt am Main [u. A.] : Lang, 2005. – Europäische Hochschulschriften, Reihe XLI, Informatik; 43. – ISBN 3–631–53684–4
- [Obe05b] OBERHAUSER, Otto: *Automatisches Klassifizieren und Bibliothekskataloge*. 2005. – <http://eprints.rclis.org/archive/00004833/01/KUB-60.pdf>
- [PB96] POMBERGER, G. ; BLASCHEK, G.: *Software Engineering – Prototyping und objektorientierte Software-Entwicklung*. München, Wien : Hanser, 1996
- [Rei06] REICHEL, Thomas: *Nachrichtenklassifikation mit einer  $l(k)$ -Grammatik*. Studienarbeit, Technische Universität Chemnitz, 2006
- [Seb02] SEBASTIANI, Fabrizio: *Machine Learning in Automated Text Categorization*. Version: March 2002. <http://nmis.isti.cnr.it/sebastiani/Publications/ACMCS02.pdf>. In: *ACM Computing Surveys, Vol. 34, No. 1*

- [Sun04] Sun Microsystems, Inc.: *Java™ 2 Platform Standard Edition 5.0 API Specification*. 2004. <http://java.sun.com/j2se/1.5.0/docs/api/index.html>
- [TP] Lenz+Partner AG: *Tai-Pan Börsensoftware*. <http://www.lp-software.de/produkte/tai-pan/>