Diplomarbeit

Spamfilterung mittels Texterkennungsverfahren

TU Darmstadt

Fachbereich Informatik

Fachgebiet Sicherheit
in der Informationstechnik

Prüfer: Prof. Dr. Claudia Eckert
Betreuer: Nicolai Kuntze und Andreas U. Schmidt

Ulrike Petersen
Matrikelnummer: 1096615

19. Februar 2007

Zusammenfassung

Spam stellt ein immer größeres Problem im E-Mail Verkehr dar und erfordert zunehmend den Einsatz von Spamfiltern. Die aktuell verfügbaren Spamfilter arbeiten entropiebasiert und stützen sich in der Regel auf so genannte Bayes'sche Filter, die Spam aufgrund bedingter Wahrscheinlichkeiten klassifizieren, oder auf die Markov-Diskriminierung, die Zeichenketten in den E-Mails identifiziert und zuordnet.

Die bisherigen Filtermethoden liefern zwar gute Ergebnisse, es ist jedoch sinnvoll, verschiedene Verfahren zur Filterung von Spam miteinander zu kombinieren, um bessere Ergebnisse zu erzielen. Weiterhin erschwert die Kombination verschiedener Techniken das Umgehen derartiger Filter, da verschiedene Verfahren auf unterschiedliche Charakteristiken der E-Mails abstellen.

Diese Arbeit behandelt die Filterung von Spam mittels eines Textanalyseverfahrens. Dazu wird zunächst der Begriff der Entropie definiert, der weiter zum Begriff des informationsbasierten Abstands zwischen Texten und zur relativen Entropie führt. Mittels dieser relativen Entropie erfolgt nun die Klassifizierung der E-Mails. Im Gegensatz zu Bayes'schen Filtern, die jedem Wort der E-Mail eine Wahrscheinlichkeit zuordnen um die Spamwahrscheinlichkeit der E-Mail zu berechnen, betrachtet die Textanalyse den Informationsgehalt, der durch die Entropie definiert wird.

Inhaltsverzeichnis

1 Einführung **8**
- 1.1 Motivation . . . 8
- 1.2 Aufbau der Arbeit . . . 10

2 E-Mails **12**
- 2.1 Aufbau einer E-Mail . . . 12
  - 2.1.1 SMTP-Envelope . . . 12
  - 2.1.2 E-Mail Header . . . 13
  - 2.1.3 E-Mail Body . . . 13
- 2.2 Protokolle . . . 14
  - 2.2.1 Simple Mail Transfer Protocol (SMTP) . . . 14
  - 2.2.2 Domain Name Service (DNS) . . . 14
  - 2.2.3 Post Office Protocol (POP) . . . 15
  - 2.2.4 Internet Message Access Protocol (IMAP) . . . 15
  - 2.2.5 mbox . . . 15
- 2.3 Multipurpose Internet Mail Extension (MIME) . . . 16
  - 2.3.1 MIME Header-Felder . . . 16
  - 2.3.2 S/MIME . . . 17

3 Spam **18**
- 3.1 Was ist Spam? . . . 18
  - 3.1.1 Definition . . . 19
  - 3.1.2 Spamkategorien . . . 20
- 3.2 Möglichkeiten der Spamfilterung . . . 21
  - 3.2.1 Statische Verfahren . . . 22
  - 3.2.2 Dynamische Verfahren . . . 24
- 3.3 Falsch-positiv, falsch-negativ . . . 26
- 3.4 Behandlung der als Spam erkannten E-Mail . . . 27
  - 3.4.1 Ablehnung der E-Mail . . . 27
  - 3.4.2 Löschen der E-Mail mit Benachrichtigung des Absenders 28

3.4.3 Verwerfen der E-Mail . . . 28
3.4.4 Markieren und Weiterleiten der E-Mail . . . 28
3.4.5 Umleiten der E-Mail an eine andere Adresse . . . 29
3.5 Umgehen von Spamfiltern . . . 29
3.5.1 Maskieren von Wörtern . . . 29
3.5.2 Unwichtige Sequenzen . . . 30
3.5.3 Einbetten der Texte in Bilder . . . 30
3.5.4 Codierung der Nachricht . . . 30

4 Filtermethoden **31**
4.1 Unterscheidung der Filtermethoden . . . 31
4.2 Training des Filters . . . 32
4.3 Simulation . . . 32

5 Informationstheorie **34**
5.1 Kommunikationssysteme . . . 34
5.2 Stochastische Prozesse . . . 35
5.2.1 Markov-Prozesse . . . 36
5.2.2 Ergodische Prozesse . . . 36
5.3 Information und Entropien . . . 37
5.3.1 Entropien . . . 37
5.3.2 Redundanz . . . 38

6 Datenkompression **40**
6.1 Zeichencodierung . . . 40
6.2 Redundanz . . . 41
6.3 Grundlagen der Datenkompression . . . 42
6.4 Probleme der Datenkompression . . . 43
6.4.1 Modellierung der Statistiken der Quelle . . . 43
6.4.2 Darstellung der Nachricht . . . 45
6.5 Verlustfreie Kompressionsstrategien . . . 46
6.5.1 Algorithmen der LZ-Familie . . . 47
6.5.2 Prediction by Partial Matching (PPM) . . . 48
6.6 Implementierung der Kompressionsstrategien . . . 49
6.6.1 gzip . . . 49
6.6.2 rar . . . 50

7 Problemfelder der Texterkennung **51**
7.1 Sequenzerkennung . . . 52
7.2 Sequenzklassifizierung . . . 53
7.3 Spezialfall: E-Mail Erkennung . . . 53

7.3.1 Berechnung der Filterentscheidungen 53
7.3.2 Abgrenzung der Spamfilterung mittels Texterkennungsverfahren zu bereits bestehenden Verfahren 54

8 Spamfilterung mittels Sequenzklassifizierung 57
8.1 Berechnung des Informationsabstands 57
8.1.1 Kolmogorov-Komplexität 58
8.1.2 Relative Entropie 59
8.2 Berechnung der (relativen) Entropie 61
8.2.1 Abschätzung der Entropie 61
8.2.2 Berechnung der Entropie H(A) 62
8.2.3 Berechnung der bedingten Entropie H(X,A) 63
8.3 Ergebnis 63

9 Implementierung des Filters 66
9.1 Einordnung der E-Mails mittels `procmail` 67
9.2 Berechnung der Korpusgröße 68
9.3 Berechnung der Werte für die Filterentscheidungen 69
9.4 Berechnung der Filterentscheidung 72
9.5 Integration des Filters in ein bestehende Strukturen 72
9.5.1 Integration der Filterroutine in einen bestehenden Spamfilter 73
9.5.2 Integration des Filters in ein Firmennetz 75

10 Analyse 77
10.1 Bestimmung eines geeigneten Kompressionsgrades 79
10.2 Vergleich der Werte der relativen Entropie und Delta 82
10.3 Darstellung in Abhängigkeit von der Korpusgröße 84
10.4 Bewertung des Filters in Abhängigkeit von den Lernkurven . . 86
10.5 Bewertung des Filters in Abhängigkeit vom verwendeten Wörterbuch 88
10.6 Bewertung in Abhängigkeit von der Berechnungszeit 90

11 Fazit 92

I Anhang 96
Abkürzungsverzeichnis 97
Stichwortverzeichnis 98
Literaturverzeichnis 100

Abbildungsverzeichnis

1.1 Darstellung der Arten empfangener E-Mails 8

3.1 Darstellung der verschiedenen Kategorien, denen Spam-Mails angehören können 21

3.2 Übersicht über die technischen Ansätze zur Spamfilterung . . 22

5.1 Schematisches Darstellung eines Kommunikationssystems . . . 35

9.1 Einbettung der Texterkennung in einen Spamfilter. Jede einkommende E-Mail wird analysiert und ein Ergebnis berechnet. Anhand dieses Ergebnisses wird eine Filterentscheidung getroffen 73

9.2 Möglicher Aufbau eines Firmennetzwerkes mit einem Mailserver und einem in einem Gateway integrierten Spamfilter . . . 76

10.1 Klassifikationsergebnisse für die Kompressionsalgorithmen gzip (links) und rar (rechts) in Abhängigkeit vom Kompressionsparameter und von Header und Body. Dargestellt sind die prozentualen Fehlentscheidungen, aufgeschlüsselt in falsch-positive (fp), falsch-negative (fn) und keine mögliche Entscheidungen (kE) 80

10.2 Klassifikationsergebnisse (links) und Lernkurven (rechts) für die Filterentscheidungen unter mittels *Delta* und der relativen Entropie unter Verwendung der Kompressionsalgorithmen gzip und rar in Abhängigkeit vom analysierten Bereich der E-Mail (Header/Body) bei der Analyse von 10000 E-Mails. Dargestellt sind die Klassifikationsergebnisse für falsch-positive (fp) und falsch-negative Entscheidungen (fn), sowie keine Entscheidung (kE) 83

10.3 Klassifikationsergebnisse (links) und Lernkurven (rechts) für die Filterentscheidungen unter mittels *Delta* und der relativen Entropie unter Verwendung der Kompressionsalgorithmen gzip und rar in Abhängigkeit vom analysierten Bereich der E-Mail (Header/Body) bei der Analyse von 10000 E-Mails. Dargestellt sind die Klassifikationsergebnisse für falsch-positive (fp) und falsch-negative Entscheidungen (fn), sowie keine Entscheidung (kE) 84
10.4 Differenz der Korpusgrößen in Bezug auf die Anzahl von Spam- und Ham-Mails (links) und korrekte Entscheidungen in Abhängigkeit vom verwendeten Korpus (rechts) 85
10.5 Absolute Differenz der Korpusgrößen des Body in Bit in Abhängigkeit vom verwendeten Korpus (links) und Gesamtgrößen der verwendeten Korpora in Bits (rechts) 85
10.6 Lernkurven unter Verwendung von gzip bei Analyse des Header (links) und des Body (rechts) in Abhängigkeit von unterschiedlicher Zusammensetzung des verwendeten Korpus 86
10.7 Lernkurven unter Verwendung von rar bei Analyse des Header (links) und des Body (rechts) in Abhängigkeit von unterschiedlicher Zusammensetzung des verwendeten Korpus 88
10.8 Darstellung der fehlerhaften Entscheidungen, die mittels verschiedener Wörterbuchgrößen unter Verwendung des Kompressionsverfahrens rar bei der Analyse des Header (links) und des Body (rechts) erreicht wurden. Dargestellt sind falsch-positive Entscheidungen (fp), falsch-negative Entscheidungen (fn) und keine Entscheidung (kE) 89
10.9 Durchschnittliche Berechnungszeit in Abhängigkeit von der Menge der eintreffenden E-Mail und dem verwendeten Kompressionsverfahren für die Berechnung von 10000 E-Mails (links) und in Abhängigkeit von der Größe des verwendeten Wörterbuchs bei dem Kompressionsverfahren rar bei Berechnung von 1000 E-Mails (rechts) 90

Listings

9.1 Perl-Programm für das Erzeugen von `procmail`-Rezepten . . . 67
9.2 Ausschnitt des Perl-Programms für die Durchführung der Kompression mittels gzip . . . 68
9.3 Ausschnitt des Perl-Programms für das Auslesen der Dateigröße mittels des Moduls `File::stat` . . . 69
9.4 Ausschnitt des Perl-Programms für die Initialisierung des Filters mittels Spam-Mails . . . 69
9.5 Initialisierung des Korpus mittels dem Header der E-Mails . . . 70
9.6 Exemplarische Berechnung der für die Filterentscheidung relevanten Werte in Bezug auf den Korpus des Body der Spam-Mails 71

Kapitel 1

Einführung

1.1 Motivation

E-Mail ist mittlerweile neben dem Telefon das wichtigste Kommunikationsmittel. Dies gilt sowohl für den privaten als auch für den geschäftlichen Bereich. Mit dem zunehmenden Versand von E-Mails steigt auch das Spamaufkommen und wird vor allem für Unternehmen zu einem immer größeren Problem. Aktuell liegt der Anteil unerwünschter E-Mails in Deutschland bei ungefähr 71% [fWuT06]. Grundsätzlich können alle empfangenen E-Mails einer der in Abbildung 1.1 dargestellten Art von E-Mails zugeordnet werden.

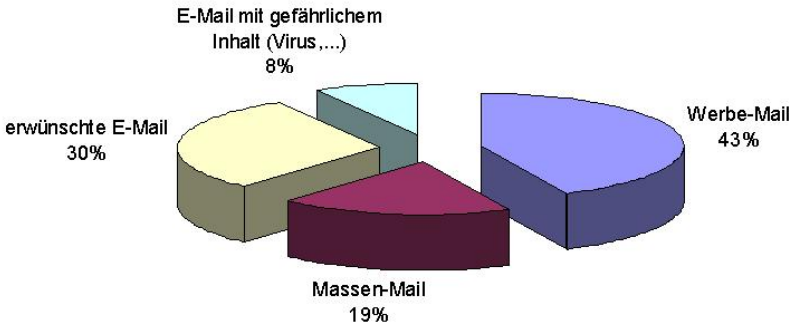

Abbildung 1.1: *Darstellung der Arten empfangener E-Mails [fWuT06]*

Mit dem steigenden Aufkommen von Spam-Mails steigen auch die mit dem E-Mail Verkehr verbundenen Kosten. Zu diesen Kosten gehören zum

einen Anschaffungskosten für neue Server zur Verwaltung des E-Mail Verkehrs, als auch Kosten die aufgrund von bösartigem Code entstehen, der in Spam-Mails enthalten sein kann. Weitere Kosten entstehen für Unternehmen, deren Angestellte einen steigenden Anteil ihrer Arbeitszeit für das manuelle Nachsortieren der eintreffenden E-Mails aufwänden müssen. Während Spam für viele Unternehmen mit finanziellen Belastungen verbunden ist, scheint sich der Versand für die Spammer zu lohnen. So zeigt eine Studie des *Wall-Street Journal* aus dem Jahr 2002, dass bereits eine Antwortrate von 0,001% auf Verweise auf kostenpflichtige Webseiten oder Produkte ausreichend ist, um den Versand von Spam profitabel zu machen [WS02]. So lange der Versand von Spam profitabel ist, wird auch dessen Aufkommen steigen.

Aufgrund des steigenden Spamaufkommens werden zunehmend Spamfilter eingesetzt. Da einzelne Verfahren zur Spamfilterung bei Kenntnis der zugrundeliegenden Mechanismen leicht umgangen werden können, werden oft Kombinationen verschiedener Filtermöglichkeiten verwendet. Kennen die Spammer die den Filtern zugrunde liegenden Techniken, können jedoch auch derartige Filter umgangen werden. Eine beständige Weiterentwicklung bekannter und eine Neuentwicklung weiterer Verfahren zur Spamfilterung ist daher unumgänglich. Meist werden lernfähige (oder dynamische) Filter eingesetzt, deren Vorteil darin besteht, dass die Filterentscheidungen für die Einteilung der E-Mails anhand konkreter Benutzerordner getroffen werden und somit für verschiedene Anwender auch unterschiedlich ausfallen können. Ein Spammer müsste somit das Verhalten einzelner Benutzer berücksichtigen um sicherzustellen, dass die von ihm versendeten Spam-Mails dem Anwender zur Kenntnis gelangen.

Wie in den folgenden Abschnitten dargelegt, gibt es bereits verschiedene dynamische Spamfilter. Ein von den meisten Spamfiltern verwendetes Verfahren ist die Bayes'sche Klassifizierung, die die Klassifizierungsentscheidung anhand der statistischen Wahrscheinlichkeit für das Auftreten charakteristischer Wörter trifft. Derartige Filter können jedoch umgangen werden, wenn auf für Spam charakteristische Wörter (weitestgehend) verzichtet wird. Diese Arbeit beschäftigt sich mit der Entwicklung einer neuen Filtermethode, die auf den Inhalt der zu filternden E-Mails abstellt. Dazu wird der relative Informationsgehalt einer neu ankommenden E-Mail in Bezug auf die bereits gefilterten Nachrichten abgeschätzt und anhand des erhaltenen Wertes eine Entscheidung getroffen. Im Gegensatz zu den bisher bekannten dynamischen Filtern stellt die Filterung mittels Texterkennung nicht auf die Wahrscheinlichkeiten einzelner Wörter ab, sondern auf den gesamten Inhalt der E-Mail. Für jede neu eintreffende E-Mail wird anhand des gesamten Inhalts, der sowohl Header als auch den (gesamten) Body der E-Mail betrifft, eine Filterentscheidung anhand bisheriger Entscheidungen getroffen. Um einen derar-

tigen Filter zu umgehen, müssten Spammer somit den gesamten Inhalt ihrer E-Mails verändern, was sehr schwer oder sogar unmöglich ist. Ein Umgehen eines derartigen Spamfilters mittels bisher bekannter Angriffe auf lernfähige Filter ist gar nicht oder nur sehr schwer möglich.

1.2 Aufbau der Arbeit

Vorliegende Arbeit ist in sechs Teile gegliedert. In **ersten Abschnitt** wird zunächst die Motivation für die Forschung im Rahmen von Spamfiltern dargelegt. Dies beinhaltet die Darlegung der allgemeinen Problematik, die sich aus dem steigenden Spamaufkommen ergibt, sowie Gründe für das Entwickeln weiterer Filtermöglichkeiten. Der **zweite Abschnitt** beinhaltet die notwendigen Grundlagen. Dies betrifft zunächst die Darstellung der E-Mail als Kommunikationsmittel mit den zugehörigen Mechanismen, sowie die Klärung des Begriffes "Spam" und einen Überblick über eine Auswahl aktuell verwendeter Filtermethoden. Weiterhin werden Verfahren zur Datenkompression und Informationstheorie dargestellt, die die Grundlage für die Spamfilterung mittels Texterkennungsverfahren liefern.

Abschnitt drei befasst sich zunächst mit der Analyse von Texten und den damit verbundenen Problemen. Aus den verschiedenen Möglichkeiten zur Analyse von Texten wird ein Verfahren zur Analyse von E-Mails hergeleitet, das die Grundlage für den implementierten Spamfilter liefert. Dieses Verfahren liefert die theoretischen Grundlagen zur inhaltsbasierten Spamfilterung mittels Textanalyseverfahren und stellt den Bezug zu den im vorhergehenden Abschnitt behandelten Grundlagen her. In **Abschnitt vier** erfolgt die Darstellung der Implementierung des in Abschnitt drei dargestellten Filters. Dies betrifft vor allem die (verkürzte) Darstellung des Quellcodes und die Erklärung der Funktionalitäten des Filters. Desweiteren wird im Rahmen dieses Abschnitts eine mögliche Einsatzumgebung des Filters dargestellt. Diese Einsatzumgebung kann in den Spamfilter selbst und in das Netzwerk, in das der Filter integriert werden soll, unterteilt werden.

In **Abschnitt fünf** erfolgt die Analyse der E-Mails mittels des im vorhergehenden Abschnitt beschriebenen Filters. Zunächst erfolgt eine zusammenfassende Darstellung der mittels des Filters erreichten Ergebnisse. Weiterhin wird eine vergleichende Darstellung der Ergebnisse vorgenommen, die unter Verwendung der Kompressionsverfahren gzip und rar erreicht wurden, sowie eine Analyse der zugehörigen Berechnungszeiten. Schließlich ist die Analyse des Kompressionsverfahrens rar bei Verwendung unterschiedlich großer Wörterbücher dargestellt. Ziel dieses Abschnitts ist die detaillierte Analyse des Filters, um die Leistungsfähigkeit und die für den Filter relevanten

Parameter identifizieren zu können. **Abschnitt sechs** liefert zunächst eine Zusammenfassung der aus der Analyse erhaltenen Ergebnisse. Im Anschluss daran werden im Rahmen des Ausblicks Möglichkeiten zur Verbesserung der Filterentscheidungen dargestellt. Diese Möglichkeiten stützen sich auf die im vorhergehenden Abschnitt erlangten Erkenntnisse.

Kapitel 2

E-Mails

Die E-Mail stellt mittlerweile das Hauptkommunikationsmittel des Internet dar und ermöglicht den weltweiten Austausch elektronischer Nachrichten. Der Umgang mit E-Mails erfordert eine Menge festgelegter Protokolle und Mechanismen, die dem Erstellen, Versenden, Empfangen, Betrachten und Speichern der E-Mails dienen und die Kommunikation mittels E-Mails erst möglich machen.

2.1 Aufbau einer E-Mail

Eine elektronische Nachricht (E-Mail) besteht zunächst nur aus einzelnen Zeichen, die als ASCII interpretiert werden. Jede E-Mail hat einen Umschlag (SMTP-Envelope) und einen Inhalt [Res01, Tan03].

Der Umschlag enthält die Informationen die zur Übertragung und Zustellung einer Nachricht benötigt werden. Der Inhalt einer Nachricht wird weiter in Header und Body unterteilt und umfasst den Teil, der dem Empfänger zugestellt wird. Der Header enthält unter anderem Informationen über Absender, Empfänger und Versandweg der E-Mail, während der Body die eigentlichen Nachricht enthält. Header und Body werden mittels einer Leerzeile voneinander getrennt.

2.1.1 SMTP-Envelope

Als SMTP-Envelope bezeichnet man die für die Zustellung einer E-Mail relevanten Informationen. Er stellt den "Briefumschlag" einer E-Mail-Nachricht dar und ist in dieser Form für den Anwender nicht sichtbar.

Der SMTP-Envelope wird als Reihe von SMTP Protokolleinheiten versendet und enthält die Informationen, die ein Mailserver beim Versand ei-

ner E-Mail benötigt. Dies betrifft die Absenderadresse (an die auch Fehler gemeldet werden), einen oder mehrere Empfänger und optionale Protokollerweiterungen. Ein Großteil der enthaltenen Informationen gehen nach der Zustellung der E-Mail verloren; einige Teile, wie Absenderadresse und die Route der E-Mail, werden in den Header übernommen [Res01, Kle01].

2.1.2 E-Mail Header

Der Header einer E-Mail besteht aus einer Sequenz von Zeichen mit einer speziellen Syntax. Jede Headerzeile beginnt mit einem Schlüsselwort, ihrem Namen, gefolgt von einem Doppelpunkt und dem Inhalt. Beendet wird eine Headerzeile durch einen Zeilenumbruch. Die Reihenfolge der Headerzeilen ist beliebig und von der verwendeten Software abhängig. Headerdaten können grundsätzlich nicht als authentisch vorausgesetzt werden, da sie beliebig durch den Absender verändert werden können.

Die meisten Headerfelder sind optional, notwendig sind lediglich das Absender- und das Datumfeld [Res01, Pal97]. Viele Headerzeilen werden vom E-Mail Programm des Absenders und von den Servern, von denen die E-Mail beim Versand weitergeleitet wird, hinzugefügt. Nicht standardisierte Headerzeilen werden mit einem vorangestellten "X" markiert. Bei derartigen Headerzeilen handelt es sich zum Beispiel um Informationen von vorangeschalteten Spamfiltern oder um Informationen des Mail-Clients. Diese Zeilen können beliebig eingefügt werden. Die einzige Bedigung ist, dass diese Zeilen mit X beginnen [Res01].

2.1.3 E-Mail Body

Der Body einer E-Mail enthält die eigentliche Nachricht. Er besteht aus einer Sequenz von Zeichen, die auf den Header folgt und von diesem durch eine Leerzeile getrennt sind. Eine Zeile des Body darf maximal 998 Zeichen enthalten, sollte jedoch auf maximal 78 Zeichen beschränkt werden (exklusive Steuerzeichen für den Zeilenumbruch). Weitere Beschränkungen und Untergliederung des Body erfolgen nicht.

Die Codierung der Zeichen des Body ist zunächst auf 7-Bit ASCII festgelegt [Res01]. Zur Übertragung multimedialer Inhalte oder Zeichen, die nicht mittels 7-Bit ASCII dargestellt werden können, wird MIME (Multipurpose Internet Mail Extension) eingesetzt [FB96a, Moo96].

2.2 Protokolle

Das Erstellen, Versenden, Empfangen und Speichern von E-Mails erfordert verschiedene Protokolle und Mechanismen. Die Notwendigkeit verschiedener Mechanismen wird bereits bei der Betrachtung des (vereinfachten) Versands einer E-Mail deutlich [Tan03].

Beim Versand einer E-Mail wird diese zunächst auf einem Mailserver für ausgehende E-Mails abgelegt. Das Simple Mail Transfer Protocol (SMTP) regelt die Übertragung [Kle01]. Anschließend erfolgt mittels DNS (Domain Name Service) die Bestimmung des Rechners, zu dem die E-Mail übertragen werden soll [Moc87]. Auch dieser Vorgang wird über das SMPT-Protokoll geregelt und kann sich mehrfach wiederholen. Ist die E-Mail im Postfach des Empfängers angelangt, kann sie mittels geeigneter Protokolle, wie zum Beispiel POP (Post Office Protocol) oder IMAP (Internet Message Access Protocol) abgeholt werden [MR96, Cri03]. Das Speichern einer E-Mail auf dem lokalen Computer erfolgt im mbox-Format, das die Speicherstruktur der E-Mails definiert [Hal05].

2.2.1 Simple Mail Transfer Protocol (SMTP)

Bei SMTP handelt es sich um das Basis-Protokoll zum Versand von E-Mails, das einen zeilenbasierten ASCII-Dialog zu einer sicheren und effizienten Übertragung standardisiert [Kle01, BW02]. SMTP ist unabhängig vom zugrundeliegenden Übertragungssystem.

Zur Übertragung einer Nachricht baut der SMPT-Client zunächst einen Übertragungskanal zum SMTP-Server auf und fordert somit Aktionen des empfangenden Rechners an. Nach einem ersten Handshake, initiiert der Client den Versand der Nachricht. Die Übertragung erfolgt entweder direkt zwischen dem eigentlichen Sender und dem Empfänger, oder über Zwischenknoten (Gateways). Das nächste Gateway wird unter Verwendung des Mail eXchanger Mechanismus (MX) des DNS bestimmt. Ist eine Übertragung der Nachricht nicht möglich, so hat der Client die Aufgabe, einen Fehlerbericht zu erstatten.

2.2.2 Domain Name Service (DNS)

Bei DNS handelt es sich um einen hierarchischen Verbund von Servern, mit deren Hilfe die Adressauflösung im Internet erfolgt [Moc87, BW02]. Eine E-Mail-Adresse ist von der Form *user@domain*; für die Auflösung der Domäne auf eine IP-Adresse werden DNS-Server benötigt.

Da nicht alle Computer oder Domänen in der Lage sind E-Mails zu empfangen, werden so genannte *mail eXchanger (MX)* Einträge verwendet, in denen die Namen für eine Domäne zuständigen Mail-Server abgelegt sind. Ist für eine Domäne kein MX-Eintrag vorhanden, so wird versucht, die E-Mail direkt bei der IP-Adresse abzuliefern, die sich aus der DNS-Auflösung der Domäne ergeben hat.

2.2.3 Post Office Protocol (POP)

Das *Post Office Protocol* in der Version 3 (POP3) soll Arbeitsplatzrechnern den dynamischen Zugriff auf einen Mailserver ermöglichen, um die dort abgelegten E-Mails herunterladen zu können [MR96]. POP3 dient lediglich dem Abrufen von E-Mails und soll keine weiteren Manipulationsmöglichkeiten der sich auf dem Server befindlichen E-Mails bieten. Grundsätzlich wird die E-Mail lediglich heruntergeladen und anschließend vom Server gelöscht. Ein komplexeres Protokoll, IMAP, wird in Sektion 2.2.4 dargestellt.

2.2.4 Internet Message Access Protocol (IMAP)

Das Internet Message Access Protocol in der Version 4 (IMAP4) ermöglicht einem Client den Zugriff und die Manipulation von E-Mails auf einem Server [Cri03]. Die Funktionalität entspricht der einer lokalen Mailbox, so dass Ordner, die sich auf dem Server befinden, bearbeitet und verändert werden können. Weiterhin wird einem Client die Möglichkeit gegeben, nach der Einwahl in das Internet eine Resynchronisierung mit dem Server vorzunehmen.

Der Zugriff auf Nachrichten mittels IMAP4 erfolgt unter Zuhilfenahme von Zahlen. Hier handelt es sich entweder um Sequenznummern der Nachrichten oder um eindeutige Identifikatoren, die jeder Nachricht zugeordnet werden.

2.2.5 mbox

Das Format *mbox* beschreibt eine Gruppe von Dateiformaten zur Speicherung von E-Mails [Hal05]. *mbox* ist kein Standard, sondern beschreibt eine Speicherkonvention für E-Mails, die vor allem auf `Unix`-Systemen Anwendung findet.

Eine *mbox* ist eine einzelne Datei, die lineare Sequenzen einer oder mehrerer E-Mails enthält, die von E-Mail Clients als logische Ordner behandelt werden. Jede Nachricht beginnt mit einem Separator, der den Absender der Nachricht, sowie das Datum und die Uhrzeit identifiziert, zu der die Nachricht

empfangen wurde. Einzelne Nachrichten sind durch eine Leerzeile voneinander getrennt. Die einzelnen Nachrichten einer *mbox* müssen dem in [Res01] definierten Format einer E-Mail folgen. Die Codierung der E-Mails erfolgt mit 7 Bit.

2.3 Multipurpose Internet Mail Extension (MIME)

Multipurpose Internet Mail Extensions (MIME) ist eine Spezifikation für Datenformate [FB96a, Tan03]. Es handelt sich um einen Standard zur Festlegung von Struktur und Aufbau von E-Mails. Ziel ist es, das in RFC 2822 definierte Format weiterhin zu nutzen [Res01], zusätzlich jedoch Codierungsregeln für Nachrichten zu definieren, die nicht ASCII verwenden. Eine Erweiterung des Standards MIME erfolgt mittels S/MIME, wodurch das Verschlüsseln und signieren digitaler Nachrichten ermöglicht wird [Ram99].

2.3.1 MIME Header-Felder

Mittels MIME kann zwischen Sender und Empfänger ein Austausch über die Art der Daten und den Übertragungsweg erfolgen. So ist auch die Übertragung von Zeichen, die nicht dem ASCII-Standard entsprechen, möglich. Dies betrifft vor allem die Übertragung von Bildern und multimedialen Inhalten. Dazu definiert MIME eine Menge zusätzlicher RFC822-konformer Headerfelder, die zur Beschreibung des Inhalts einer MIME-Einhiet verwendet werden [FB96a]. Im Einzelnen sind dies die Header-Felder *MIME-Version*, *Content-Type* und *Content-Transfer-Encoding*.

MIME-Version

Mittels MIME wird ein zusätzliches Header-Feld definiert, das die MIME-Version festlegt, auf der die übertragene Nachricht basiert. Ist dieses Feld im Header enthalten, so wird damit verdeutlicht, dass die Nachricht gemäß RFC2045 codiert wurde. Aktuell ist die MIME-Version auf "1.0" festgelegt [FB96a]. Die Behandlung von Nachrichten, die nicht MIME-Version 1.0 entsprechen, ist derzeit nicht festgelegt. Eine Nachricht, die nicht MIME-Version 1.0 entspricht, muss nicht notwendigerweise mittels 7-Bit ASCII codiert sein, sondern kann einen beliebigen Typ enthalten.

Content-Type

Das Header-Feld *Content-Type* beschreibt die Art der Daten im Body einer E-Mail [FB96a, Moo96, Lev98], um dem Empfänger eine korrekte Darstellung der Daten zu ermöglichen. Die Kennzeichnung der Daten erfolgt mittels des Tupels `Type/Subtype`, wobei `Type` die allgemeine Gruppe der Applikationen und `Subtype` das exakte Datenformat zur Verarbeitung der einzelnen Bit des Nachrichtenkörpers angibt (zum Beispiel `image/jpg`). Die im Body enthaltenen Daten werden auf diese Art und Weise so detailliert wie möglich beschrieben, um dem E-Mail Programm des Empfängers die Möglichkeit der korrekten Darstellung der Daten zu geben. In RFC 2046 werden sieben MIME-Typen beschrieben [FB96b].

Content-Transfer-Encoding

Einige Daten, wie zum Beispiel Binärdaten oder Zeichensätze mit 8-Bit, können nicht mittels SMTP übertragen werden [Kle01], da die Codierung der Zeichen auf 7-Bit ASCII beschränkt ist. Auch die Länge der einzelnen Zeilen des Body ist auf 1000 Zeichen beschränkt. Daher ist es notwendig, einen Mechanismus zum Umgang mit Daten, die nicht mittels 7-Bit ASCII codiert sind, zu spezifizieren.

Das MIME-Feld *Content-Transfer-Encoding* spezifiziert das für den Body verwendete Kodierungsverfahren. Aktuell können drei Formen der Kodierung unterschieden werden: *identity*, *quoted-printable* und *base64*, die den Domänen der binären Codierung, sowie der Codierung mittels 8-Bit und mittels 7-Bit angehören.

2.3.2 S/MIME

S/MIME (Secure/Multipurpose Internet Mail Extensions) ist eine Erweiterung von MIME um die Möglichkeit der Verschlüsselung und (digitalen) Signatur zur Sicherung der Authentizität, Integrität und Vertraulichkeit im Umgang mit elektronischen Daten [Eck04, Ram99]. Dies betrifft nicht nur 7-Bit ASCII, sondern Daten beliebigen Binärcodes. Dazu wird mittels des Content-Type `application/x-pkcs7` ein zusätzlicher MIME-Typ definiert.

Kapitel 3

Spam

Bei Spam handelt es sich grundsätzlich um unerwünschte E-Mails, die meist in Massen versendet werden. Eine einheitliche Definition für Spam gestaltet sich schwierig und hat sich bisher nicht durchgesetzt.

Aufgrund des steigenden Spamaufkommens wird die Kommunikation mittels E-Mail immer mehr erschwert, so dass zunehmend Mechanismen zur Verhinderung von Spam notwendig sind. Diese Mechanismen betreffen nicht nur technische Ansätze, die in Spamfiltern realisiert sind, sondern auch weitere Ansätze, wie gesetzliche, soziale oder ökonomische Ansätze. Im Rahmen der technischen Ansätze zur Spambekämpfung muss nach dem Treffen einer Filterentscheidung ein Mechanismus zum Umgang mit der als Spam erkannten E-Mail zur Verfügung gestellt werden. Zu den derartigen Mechanismen gehört zum Beispiel das Löschen oder Ablehnen der als Spam erkannten E-Mail, aber auch das Umleiten an eine andere Adresse.

Weiterhin ist es notwendig, Schwachpunkte eingesetzter Verfahren zu identifizieren, um diese zu beseitigen und es Spammern so zu erschweren, bestehende Filter zu umgehen. Die Möglichkeiten zum Umgehen von Filtern sind stark von der Art und der vorgenommen Kombination der verschiedenen Filterverfahren abhängig. Eine Kombination verschiedener Filtermethoden erschwert das Umgehen von Spamfiltern zwar, macht es jedoch nicht unmöglich.

3.1 Was ist Spam?

Der Name *Spam* steht ursprünglich für "Spiced Ham", eine Art 'gekochter Schinken aus der Dose'. Die Übertragung des Begriffs auf unerwünschte E-Mails geht auf einen Sketch von Monty Python zurück [Spa06].

Eine Besucherin fragt in einem Restaurant, in dem es ausschließlich Ge-

richte mit Spam gib, nach einem Gericht ohne Spam. Der Dialog zwischen Gast und Kellnerin wird permanent durch Spam-Loblieder eines Wikinger-Chors unterbrochen, der schließlich jede Unterhaltung unterdrückt: "Spam, Spam, Spam, Spam. Lovely spam! Wonderful Spam!" Ebenso wie der Gesang jede Unterhaltung unmöglich macht, so erschweren Spam-Mails die Kommunikation per E-Mail.

3.1.1 Definition

Da es keine einheitliche Begriffsbestimmung von Spam gibt, gestaltet sich das Aufstellen einer Definition für Spam schwierig. Grundsätzlich kann Spam als unerwünschte E-Mail verstanden werden. Darüber hinaus variieren die Definitionen. So kann Spam zum Beispiel als unerwünschte Werbemail, aber auch als unerwünschte Massen-Mail verstanden werden [Zdz05].

Die OECD definiert Spam in einem zweistufigen Verfahren und unterscheidet primäre und sekundäre Charakteristiken [OEC04]. Primäre Charakteristiken sind Merkmale, die allen Spam-Mails gemeinsam sind, während sekundäre Charakteristiken zwar in vielen, aber nicht in allen Spam-Mails enthalten sind. Demnach kann die Trennung von Spam und Ham mittels primärer Merkmale nach Definitionen 1 und 2 erfolgen.

Definition 1 (Spam) *Spam-Mails sind elektronische Nachrichten mit meist kommerziellem Charakter, die in Massen und unaufgefordert versendet werden.*

Definition 2 (Ham) *Unter Ham-Mails werden alle E-Mails verstanden, die nicht Spam sind. Die Mengen "Ham" und "Spam" sind disjunkt.*

Zur Verdeutlichung der Unterschiede bedürfen die primären und sekundären Merkmale der Konkretisierung, die in den folgenden Unterabschnitten vorgenommen wird.

Primäre Merkmale von Spam

Primäre Merkmale sind Merkmale, die allem Spam-Mails gemeinsam sind. Demnach handelt es sich bei Spam zunächst um elektronische Nachrichten, nicht notwendigerweise nur um E-Mails. So kann Spam zum Beispiel auch über SMS versendet oder in Foren geposted werden.

Spam wird unaufgefordert versendet, das heißt der Versand erfolgt ohne Einwilligung oder Zustimmung seitens des Empfängers. Außerdem haben Spam-Mails in der Regel einen kommerziellen Charakter in Form von Gewinnerzielungsabsicht. Der Inhalt ist demnach oft Produktwerbung. Andere

mögliche Inhalte sind politische Themen oder bösartiger Code. Schließlich erfolgt der Versand in Massen, das heißt, die E-Mail ist an viele Empfänger adressiert.

Sekundäre Merkmale von Spam

Sekundäre Merkmale sind die Merkmale, die auf die meisten Spam-Mails zutreffen, aber nicht notwendigerweise vorhanden sein müssen. So verwenden Spammer in der Regel Adressen, die sie ohne Zustimmung des Empfängers gesammelt bzw. erhalten haben. Oft werden Wörterbuchangriffe auf Domänen vorgenommen; möglich ist aber auch die Extraktion der E-Mail Adressen aus Foren oder Newsgroups, ebenso wie der käufliche Erwerb.

Spam ist in der Regel unerwünscht und/oder nutzlos für den Empfänger. Der Versand erfolgt nicht zielgerichtet sondern wahllos, das heißt der Spammer hat außer der E-Mail Adresse keine Informationen über den Empfänger. Spam wird wiederholt verschickt, es handelt sich entweder um exakte Duplikate bereits versendeter Spam-Mails oder es erfolgen nur minmale Änderungen. Weiterhin enthält Spam oft illegale oder anstößige Inhalte. Spam ist oft nicht aufzuhalten, das heißt der Empfänger hat keine Möglichkeit, den Empfang von Spam zu stoppen. Schließlich wird Spam meist anonym oder mit falscher Identität versendet. Viele Spammer verwenden dazu fremde E-Mail Server.

3.1.2 Spamkategorien

Der Inhalt verschiedener Spam-Mails kann variieren, von Angeboten für Produkte und Dienstleistungen über Informationen zu illegaler Software bis hin zu betrügerischen Angeboten. In einer Studie der Firma Brightmail von 2003 [TSo07] erfolgte eine Aufschlüsselung der Inhalte von Spam-Mails. Demnach können, wie in Abbildung 3.1 dargestellt, verschiedene Kategorien von Spam unterschieden werden.

Der Großteil der Spam-Mails (75%) kann demnach einer von drei großen Gruppen zugeordnet werden: So beinhalten 33% der versendeten Spam-Mails Produktwerbung, 24% enthalten Informationen über Finanzen und 18% der Spam-Mails haben pornographische Inhalte. Die verbleibenden 25% der Spam-Mails können unter anderem den Kategorien Freizeit, Gesundheit, Internet oder Betrug zugeordnet werden.

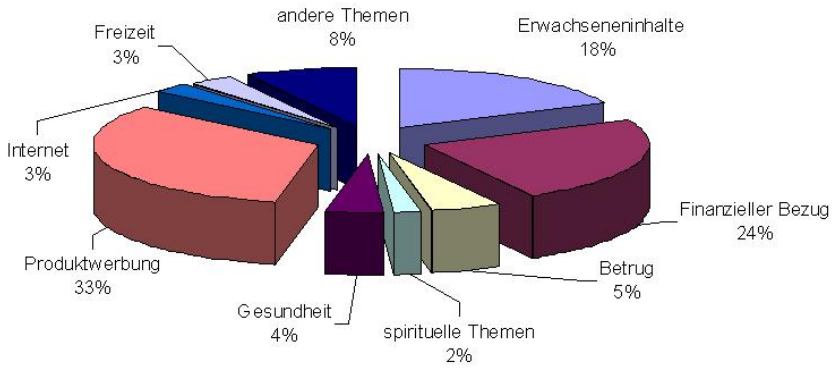

Abbildung 3.1: *Darstellung der verschiedenen Kategorien, denen Spam-Mails angehören können*

3.2 Möglichkeiten der Spamfilterung

Im Rahmen der Spamfilterung können verschiedene grundlegende Ansätze unterschieden werden. Dies betrifft neben den technischen Ansätzen vor allem gesetzliche und regulierende Ansätze, als auch soziale oder ökonimische Ansätze zur Bekämpfung von Spam [Sch04].

Mittels gesetzlicher bzw. regulierender Ansätze erfolgt eine Anpassung bestehender oder das Erstellen neuer Gesetze und Regelwerke, um so die Verbreitung von Spam einzugrenzen. Das größte Problem dieses Ansatzes besteht darin, dass die Gesetzeslage weltweit nicht homogen ist. Viele Länder haben noch immer keine Anti-Spam Gesetze verabschiedet, so dass Spammer derartige Lücken ausnutzen können, indem sie auf diese Länder ausweichen. Demgegenüber betreffen soziale Ansäze weniger die Vermeidung und Verhinderung von Spam, sondern eher die Aufklärung der Anwender über den Umgang mit Spam.

Ökonomische Ansätze hinterfragen die Möglichkeit, den Versand von Spam für die Versender zu verteuern. Dies beinhaltet zum Beispiel die Einführung einer Kostenpauschale für den Versand mehrerer E-Mails, um somit den Versand von E-Mails in Massen zu verhindern. Dieser Ansatz scheint jedoch nicht praktikabel, da vor allem für Firmen enorme Kosten entstehen können. Weiterhin wird auf diese Art und Weise die Freiheit der Internetnutzer eingeschränkt.

Technische Ansätze betreffen die Möglichkeit, bereits versendete Spam-Mails zu filtern. Wie in Abbildung 3.2 dargestellt, sind die technischen An-

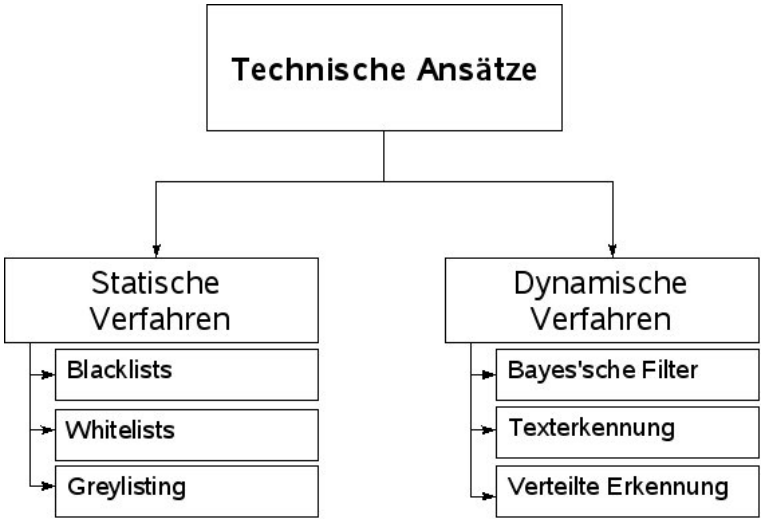

Abbildung 3.2: *Übersicht über die technischen Ansätze zur Spamfilterung*

sätze vielfältig. Sie werden oft miteinander kombiniert, um eine effektivere Filterung zu ermöglichen. Grundsätzlich können in diesem Rahmen statische und dynamische (lernfähige) Verfahren unterschieden werden. Aktuelle Spamfilter verwenden eine Kombination von Algorithmen beider Verfahren, um das Umgehen bestehender Filter zu erschweren.

3.2.1 Statische Verfahren

Bei der Filterung mittels statischen Verfahren werden Regeln angelegt, mittels denen die Filterung erfolgt. Die Regeln sind statisch, das heißt es werden im Laufe der Zeit lediglich Einträge hinzugefügt. Bereits vorhandene Regeln bleiben erhalten; sie werden weder gelöscht noch überarbeitet. Eine Anpassung an sich verändernde Gegebenheiten ist somit schwierig [Zdz05, EW05].

Filter, die statische Regeln verwenden, können relativ leicht umgangen werden da ein Spammer lediglich wissen muss, wie der eingesetzte Filter arbeitet. Statische Filter sollten also immer in Kombination mit weiteren Filtermethoden verwendet werden. Ein großer Vorteil der statischen Filter besteht jedoch darin, dass sie relativ leicht zu implementieren sind und eine gute Performanz bieten. Der Einsatz derartiger Filter in Kombination mit anderen Verfahren ermöglicht somit eine Performanzsteigerung.

Blacklists

Blacklists sind Listen die Wörter, Absender oder ganze Sätze enthalten, die häufig in Spam verwendet werden. Eine E-Mail, die eines oder mehrere der Listenelemente enthält, wird als Spam aussortiert. Die Entscheidung für das Aussortieren einer E-Mail kann sowohl von einem Schlagwort als auch von einer Kombination verdächtiger Wörter abhängig gemacht werden.

Derartige Filter können durch das maskieren "verdächtiger" Wörter umgangen werden, indem diese Wörter zum Beispiel mittels Zahlen oder Sonderzeichen unkenntlich gemacht werden (zum Beispiel "H0se" oder "Labe|"). Weiterhin müssen derartige Listen manuell erstellt werden, was zu einem erhöhten Arbeitsaufwand führt. Der Einsatz von Blacklists ist jedoch sinnvoll, um eine Vorsortierung der erhaltenen E-Mails vorzunehmen, da das Verfahren effizient implementiert werden kann. So können diese Listen unerwünschte Absender oder Mailserver enthalten, um offensichtlich unerwünschte E-Mail direkt auszusortieren.

Whitelists

Whitelists sind Listen, die vertrauenswürdige Absender enthalten, das heißt Absender, deren E-Mails ausdrücklich erwünscht sind. Hier können zum Beispiel die Einträge des Adressbuches verwendet werden. E-Mails von in Whitelists enthaltenen Absendern werden nicht weiter analysiert, sondern direkt an das Postfach des Empfängers weitergeleitet.

Das große Problem beim Whitelisting besteht darin, dass die Entscheidungen anhand des *From:* Feldes des Header getroffen werden. Dieses Feld kann jedoch nicht als authentisch vorausgesetzt werden, da es beliebig verändert werden kann. Weiterhin stellen viele Spamfilter lediglich die E-Mails der Absender zu, die im Adressbuch enthalten sind. Dies kann vor allem bei Newsgroups und Mailinglisten, die E-Mails mit variierenden Adressen versenden, zu Problemen führen.

Greylisting

Bei Greylisting handelt es sich um ein Verfahren, das eine zugestellte E-Mail mit einem temporären Fehler abweist, wenn die Kombination aus Client-IP, Absender- und Empfängeradresse noch nicht in der Datenbank gespeichert wurde. Korrekt implementierte E-Mail Server stellen die E-Mail nach einer kurzen Wartezeit erneut zu.

Heutzutage werden viele Spam-Mails von Rechnern versendet, auf denen eingebrochen wurde oder bei denen der Versand aufgrund einer Viren-Infektion erfolgte. Da hier meist nur eine minimale SMTP-Implementierung

vorliegt, können derartige Systeme die E-Mails nicht puffern, um sie bei Fehlern abermals zu versenden. Daher können mittels Greylisting bereits viele der Spam-Mails abgefangen werden. Problematisch beim Greylisting ist jedoch die Zeitverzögerung, die zwischen Versand und Empfang der E-Mail auftritt. Auch hier können Probleme mit Mailinglisten, die ihre E-Mails mit zufälliger Absenderadresse versenden, auftreten. Schließlich führt das Greylisting zusätzlich zu einem erhöhten E-Mail Aufkommen.

Verteilte Erkennung

Bei der verteilten Erkennung wird eine Datenbank verwendet, die eine Liste aller bekannten, von Spammern verwendeten Netzwerke enthält. So ist es möglich, Spam herkunfts- statt inhaltsbasiert zu filtern [Zdz05].

Ein großes Problem dieses Verfahrens besteht in der Identifikation der Netzwerke. Der Spam einer Quelle kann erst gefiltert werden, wenn diese Quelle identifiziert ist. Das zweite Problem besteht in der Mobilität der Spammer. Dies betrifft das kontinuierliche Wechseln der verwendeten Adressräume. Mit jedem Wechsel des Adressraumes ist eine neue Identifikation der Netzwerke notwendig.

3.2.2 Dynamische Verfahren

Dynamische Verfahren zeichnen sich dadurch aus, dass eine ständige Anpassung an die aktuellen Gegebenheiten erfolgt. Entscheidungen dynamischer Spamfilter werden nicht nur aus der Analyse der eingehenden E-Mail getroffen, sondern auch aus Vergleichen mit früheren E-Mails. Weiterhin können zusätzliche Eingaben seitens der Benutzer berücksichtigt werden. Derartige Filter extrahieren die Charakteristiken für die Filterung von Spam und Ham direkt aus den eintreffenden E-Mails und sind somit nur noch bedingt auf Benutzereingaben angewiesen [Zdz05, EW05].

Ein großer Vorteil dynamischer Filter besteht darin, dass sie schwer zu umgehen sind, weil der Spammer dazu Kenntnis über die bereits getroffenen Entscheidungen des Filters haben müsste. Da derartige Filter auf dem Kontext der E-Mails arbeiten, müssten Spammer weiterhin den Inhalt der von ihnen versendeten E-Mails ändern. Dynamische Filter können jedoch zu Performanzproblemen führen. Dies betrifft zum einen den von ihnen benötigten Speicherplatz, als auch die Rechenleistung, die zum Treffen neuer Entscheidungen notwendig ist. In der Regel ist es sinnvoll, vor dem Einsatz dynamischer Filter eine Vorsortierung mittels statischer Filter vorzunehmen.

Bayes'sche Filter

Bayes'sche Filter sind lernende Filter, die vom Benutzer zunächst trainiert werden müssen, um zur Filterung von Spam eingesetzt werden zu können. Derartige Filter verwenden statistische Werte um eine Wahrscheinlichkeit dafür zu bestimmen, ob es sich bei der eintreffenden E-Mail um Spam handelt oder nicht [Gra02, Gra03].

Bayes'sche Filter verwenden bedingte Wahrscheinlichkeiten, um eine E-Mail zu klassifizieren. In der Trainingsphase wird dem Filter eine Menge an E-Mails übergeben, um jedem Wort der E-Mail eine Wahrscheinlichkeit zuzuordnen. Je häufiger ein Wort in einer Spam-Mail vorgekommen ist, desto größer ist der Wahrscheinlichkeitswert. Bei jeder neu eintreffenden E-Mail wird nun eine bestimmte Anzahl an Wörtern (in der Regeln 15 oder 27 [Zdz05]) zur Berechnung der Spam-Wahrscheinlichkeit herangezogen und mittels der Auftrittswahrscheinlichkeit dieser Wörter in Spam bzw. Ham-Mails deren bedingte Wahrscheinlichkeit berechnet. Ein Schwellenwert legt fest, ab welchem Wahrscheinlichkeitswert die E-Mail als Spam behandelt wird.

Bayes'sche Filtern liefern zwar gute Werte, der alleinige Einsatz derartiger Filter ist jedoch nicht sinnvoll. So kann es zum einen zu Performanzproblemen kommen, da Bayes'sche Filter einen großen Rechenaufwand benötigen, zum anderen können auch Bayes'sche Filter umgangen werden, da auch sie nur mit einem Ausschnitt der E-Mails arbeiten.

Markov-Diskriminierung

Die Markov-Diskriminierung stellt eine Implementierung der Markov-Ketten zur Spamfilterung dar [Zdz05]. Dabei handelt es sich um eine Beschreibung voneinander abhängiger Ereignisse, mittels derer viele natürliche Prozesse und Sprachen beschrieben werden können.

Verwendet werden Markov-Ketten, um die Wahrscheinlichkeiten für Zustandsübergänge anzugeben. Markov-Ketten beschreiben Ketten von Zeichen oder Wörtern und ordnen diesen Ketten eine Wahrscheinlichkeit für deren Auftreten zu. Im Gegensatz zu Bayes'schen Filtern, die die Wörter einer E-Mail unabhängig voneinander betrachten, steht die Reihenfolge der Wörter bei der Markov-Diskriminierung im Vordergrund. Für jede neu eintreffende E-Mail wird geprüft, ob und wo Zeichenketten dieser E-Mail im Ham- und Spamkorpus auftreten. Ausschlaggebend ist die Länge der Kette. Da es möglich ist, dass Zeichenketten in beiden Korpora auftreten, werden die Zeichenketten abhängig von deren Länge gewichtet und diese Gewichtung für die Filterentscheidung herangezogen.

Der Vorteil der Verwendung von Markov-Ketten besteht darin, dass eine

konzeptuelle und lexikalische Analyse von E-Mails möglich ist, die ein höheres Maß an Korrektheit als Bayes'sche Filter liefern. Das Problem der Markov-Diskriminierung besteht in dem zur Klassifizierung benötigten Zeitaufwand. Daher wird die Länge der betrachteten Markov-Ketten beschränkt.

Texterkennung

Verfahren zur Texterkennung arbeiten auf dem Inhalt eintreffender E-Mails. Im Gegensatz zu Bayes'schen Filtern oder Markov-Ketten wird jedoch die gesamte E-Mail und nicht nur ein Ausschnitt in Form bestimmter Wörter oder Sätze betrachtet. Auch hier ist ein Training des Filters erforderlich. Im Rahmen der Texterkennung erfolgt die Berechnung der Spamwahrscheinlichkeit mittels Verfahren zur Datenkompression und relativer Entropien. Der Vorteil derartiger Verfahren im Vergleich zu den Bayes'schen Filtern besteht darin, dass Texterkennungsverfahren auf den gesamten Inhalt der E-Mails abstellen und somit schwieriger zu umgehen sind. Auf die Verfahren der Texterkennung wird im weiteren Verlauf genauer eingegangen.

3.3 Falsch-positiv, falsch-negativ

Bei der Filterung von E-Mails können zwei Arten von Fehlern unterschieden werden: falsch-negativ und falsch-positiv.

Definition 3 (Falsch-negativ) *Falsch-negativ sind die E-Mails, die als Ham klassifiziert werden, obwohl es sich um Spam handelt.*

Falsch-negative zeichnen sich dadurch aus, dass hier ein manuelles Nachsortieren der E-Mails des Posteingangs notwendig ist. Dieser Fall ist zwar nicht wünschenswert, da kein Informationsverlust auftritt jedoch unbedenklich.

Definition 4 (Falsch-positiv) *Falsch-positiv sind die E-Mails, die als Spam klassifiziert werden, ohne dass es sich um Spam handelt.*

Problematischer als falsch-negative stellen sich falsch-positive dar. Werden die als Spam erkannten E-Mails direkt vom Mailserver gelöscht, so können hohe Kosten entstehen, wenn wichtige Nachrichten nicht zugestellt werden. Dies spielt vor allem für Unternehmen eine große Rolle.

Paul Graham definiert den Unterschied zwischen falsch-positiven und falsch-negativen wie folgt:

> *Filtering rate is a measure of performance. False positives I consider more like bugs. I approach improving the filtering rate as optimization, and decreasing false positives as debugging* [Gra03].

Paul Graham versteht falsch-positive als Fehler des Systems, die behoben werden müssen. Die Implementierung eines einzelnen Verfahrens, das keine falsch-positiven liefert, ist jedoch kaum möglich, da sich Spam im Laufe der Zeit ändert, um bestehende Spamfilter zu umgehen. Häufig stellt sich das Problem, dass eine Optimierung der Filterleistung eine Erhöhung der falsch-positiven zur Folge hat. Dies ist bei vielen Verfahren der Fall, die einer eintreffenden E-Mail einen prozentualen Wert für die Spamwahrscheinlichkeit zuweisen. Bei der Filterleistung und dem Auftreten von falsch-positiven kann es sich somit um gegenläufige Parameter handeln, die lediglich durch eine Kombination verschiedener Verfahren verbessert werden können. Da falsch-positive bei der Filterung nicht auftreten dürfen, können Spamfilter auch nicht mit einer Equal-Error-Rate beschrieben werden.

3.4 Behandlung der als Spam erkannten E-Mail

Wurde eine E-Mail als Spam erkannt, so muss festgelegt werden, wie mit dieser E-Mail verfahren wird. Hier gibt es verschiedene Möglichkeiten [EW05]. Neben dem Ablehnen oder Löschen einer als Spam erkannten E-Mail ist es zum Beispiel möglich, diese auf einen für diesen Zweck eingerichteten Ordner oder E-Mail Account umzuleiten. Welcher Ansatz implementiert und realisiert wird, hängt zum einen vor allem von der Einsatzumgebung ab. Nicht alle Ansätze sind praktikabel oder sinnvoll.

3.4.1 Ablehnung der E-Mail

Es besteht die Möglichkeit, die als Spam erkannte E-Mail abzulehnen, bevor die Annahme der E-Mail gegenüber dem absendenden Host bestätigt wird. Auf diese Art und Weise erhält der Absender der E-Mail Kenntnis, dass die Annahme der E-Mail verweigert wurde. Die Erkennungsverfahren müssen jedoch bereits während der Annahme der E-Mails durchgeführt werden, so dass Performanzprobleme auftreten können. Weiterhin erhalten Spammer so Kenntnis, dass der E-Mail Account, an den die E-Mail gesendet wurde, verwendet und von einem Spamfilter geschützt wird. Daher ist es in der Regel sinnvoller, die E-Mail zunächst anzunehmen und die Erkennung zu einem späteren Zeitpunkt durchzuführen.

3.4.2 Löschen der E-Mail mit Benachrichtigung des Absenders

Bei diesem Verfahren handelt es sich um eine Alternative zum Ablehnen der E-Mail. Auch hier erhält der Absender Kenntnis, dass die von ihm gesendete E-Mail nicht zugestellt worden ist. Dieses Verfahren ist jedoch aus zwei Gründen problematisch.

Zum einen werden viele Spam-Mails mit gefälschten Absenderadressen zugestellt. Inhaber dieser E-Mail-Adressen erhalten nun als unbeteiligte Dritte Benachrichtigung über die Unzustellbarkeit der Nachrichten und haben ihrerseits eine Art Spam-Problem. Das zweite Problem besteht darin, dass viele Spam-Mails mittels Wörterbuchverfahren an so viele Mitglieder eine Domäne gesendet werden, wie möglich. Erfolgt eine Antwort auf eine Spam-Mail, so erhalten Spammer die Bestätigung, dass die betroffene E-Mail Adresse gültig ist. Außerdem ist es so unter Umständen möglich, Spam-Mails derart anzupassen, dass die Spamfilter umgangen werden können. Diese Methode sollte in der Praxis somit nicht eingesetzt werden.

3.4.3 Verwerfen der E-Mail

Eine weitere Möglichkeit besteht darin, die als Spam erkannten E-Mails direkt zu verwerfen, anstatt sie den Benutzerordnern zuzuführen.

Dieses Verfahren ist zwar einfach zu Implementieren, Fehler bei der Klassifizierung der E-Mails können jedoch nicht behoben werden. Bei einer falschpositiven Klassifizierung ist es möglich, dass wichtige E-Mails gelöscht werden, ohne dass der Empfänger davon Kenntnis erhält. Dies kann im Unternehmensbereich zu enormen Kosten füren. Von diesem Ansatz der Behandlung erkannter E-Mails sollte daher Abstand genommen werden.

3.4.4 Markieren und Weiterleiten der E-Mail

Bei diesem Verfahren werden dem E-Mail Header zusätzliche Informationen hinzugefügt, die der Empfänger zur weiteren Filterung nutzen kann. Derartig markiere E-Mails können zum Beispiel in einem separaten Postfach verwaltet werden.

Dieses Verfahren kann bei gemeinsamer Nutzung des Spamfilters eines Servers eingesetzt werden, um eine grobe Vorsortierung der eingehenden E-Mails zu erreichen. Die eigentlichen Empfänger können dabei selbst entscheiden, welche E-Mails für sie tatsächlich von Relevanz sind. Dieses Verfahren wird unter anderem im Spamfilter SpamAssassin implementiert, der mittels eines Punkt-Wertes die Spam-Wahrscheinlichkeit einer E-Mail angibt. Die

Anwender können nun den Punktwert, ab dem eine E-Mail als Spam eingestuft wird, selbst festlegen [EW05, Sch05].

3.4.5 Umleiten der E-Mail an eine andere Adresse

Eine als Spam eingestufte E-Mail kann an eine eigens dafür eingerichtete E-Mail Adresse umgeleitet werden. So können Fehler in der Implementierung und Einstellung des Spamfilters erkannt werden. Weiterhin wird auf diese Art und Weise eine Art "Quarantänestation" eingerichtet, die vor allem dann sinnvoll ist, wenn die Spam-Mails bösartien Code enthalten. Hier ist zu beachten, dass dieses Verfahren zusätzlichen administrativen Aufwand erfordert, da ein weiterer E-Mail Account zu verwalten und zu betreuen ist.

3.5 Umgehen von Spamfiltern

Trotz der stetigen Weiterentwicklung der Spamfilter finden Spammer immer neue Möglichkeiten Spamfilter zu umgehen [Sop06, GC06, Zdz05]. Daher ist eine Kombination verschiedener Filtermöglichkeiten notwendig, um das Umgehen der Filter zumindest zu erschweren. Wie ein Spamfilter umgangen werden kann, hängt von den im Rahmen der Filterung eingesetzten Verfahren ab. Zwar gibt es eine Vielzahl verschiedener Verfahren, viele Spamfiler verwenden jedoch ähnliche Algorithmen, so dass Spammer auch ohne Kenntnis der konkreten Verfahren einen Angriff auf den Filter starten können. Oft ist es möglich, mehrere Verfahren mittels eines Angriffs zu umgehen.

3.5.1 Maskieren von Wörtern

Die meisten Verfahren, die auf der Analyse von Zeichensequenzen beruhen, können durch Maskierung der Wörter, die als Indikator für Spam verstanden werden, umgangen werden. Diese Maskierung kann durch das Ersetzen oder Vertauschen einzelner Buchstaben erfolgen, aber auch durch Ersetzen des Leerzeichen durch spezielle Zeichen oder durch Einfügen von Leerzeichen zwischen allen Buchstaben, um die Wörter für den Spamfilter unkenntlich zu machen. Dies betrifft zum einen die Verwendung von Blacklists, aber auch der Einsatz Bayes'scher Filter, die nur eine geringe Menge an Wörtern als Referenz für die Spamfilterung verwenden.

Dieser Angriff kann zumindest abgeschwächt werden, indem vor der Auswahl verdächtiger Wörter alle Sonderzeichen entfernt werden, die oft für die Maskierung verdächtiger Worte verwendet werden. Eine weitere Möglichkeit

besteht darin, unter Zuhilfenahme regulärer Ausdrücke Teilstücke von verdächtigen Wörtern zu filtern.

3.5.2 Unwichtige Sequenzen

Weiterhin ist es möglich, Spam-Mails mit einer großen Menge unwichtiger Zeichensequenzen aufzufüllen. Diese Sequenzen können als HTML-Kommentar, als weißer Text auf weißem Hintergrund oder als "Microdot", das heißt in einer sehr kleinen Schrift (Schriftgröße 1), eingefügt werden. Vor allem Bayes'sche Filter haben hier einen Schwachpunkt, da sie zur Berechnung der Auftrittswahrscheinlichkeiten von Wörtern den gesamten Inhalt der E-Mail betrachten, zur eigentlichen Filterung jedoch nur einen Ausschnitt der Wörter verwenden.

3.5.3 Einbetten der Texte in Bilder

Eine weitere Möglichkeit Spamfilter zu umgehen besteht darin, den Text der E-Mail in ein Bild einzubetten. Da diese Bilder nicht von den aktuellen Spamfiltern interpretiert werden, kann anhand dieser Daten auch keine Spamfilterung erfolgen.

3.5.4 Codierung der Nachricht

Viele Spamfilter treffen ihre Filterentscheidungen anhand des Inhalts des `Betreff`-Feldes einer E-Mail. Spammer können die Nachrichtenfelder mittels eines in [Moo96] definierten Verfahrens codieren, das nicht 7-Bit ASCII entspricht. Viele Spamfilter führen im Rahmen der Spamfilterung keine Decodierung der Headerzeilen durch. Die `Betreff`-Zeile ist für den Spamfilter nicht lesbar und kann somit auch nicht für die Filterentscheidungen verwendet werden.

Kapitel 4

Filtermethoden

Im Rahmen der technischen Ansätze zur Spamfilterung können verschiedene Formen der Filter unterschieden werden. Dies betrifft zunächst die grundsätzliche Unterscheidung, ob beim Treffen der Filterentscheidungen Eingriffe seitens des Benutzers erfolgen oder nicht (Filter mit/ohne Feedback). Die vom Benutzer vorgenommen Eingriffe können nun im Rahmen des Trainings des Filters weiter unterschieden werden. Hier kann zwischen verschiedenen Möglichkeiten der Eingriffe seitens des Benutzers unterschieden werden. Die Verarbeitung der Ergebnisse erfolgt nun im Rahmen der Simulation, mittels derer der Filter in Bezug auf verschiedene Faktoren, wie die Filterrate oder die Performanz, bewertet wird.

4.1 Unterscheidung der Filtermethoden

Im Rahmen der Filterung von E-Mails können zwei verschiedene Filtermethoden unterschieden werden. Bei einem *Filter mit Feedback* erfolgen Eingriffe seitens des Benutzers. Durch manuelles Nachsortieren der E-Mails und konstantes Kontrollieren der Filterentscheidungen sollen Fehler vermieden werden, die längerfristige Auswirkungen haben. Filter mit Feedback können verschiedene Möglichkeiten des Trainings verwenden, wie sie im folgenden Abschnitt dargestellt werden.

Bei einem *Filter ohne Feedback* erfolgen keine Eingriffe seitens des Benutzers. Einmal vom System klassifizierte E-Mails werden in den Ordnern belassen. Dieses Verfahren kann dazu führen, dass keine korrekten Entscheidungen möglich sind, da jede neue Entscheidung auf bereits getroffenen Entscheidungen aufbaut. Eine falsch-klassifzierte E-Mail führt zu einer Vermischung von Spam- und Ham-Mails, die im Rahmen weiterer Entscheidungen problematisch werden kann.

4.2 Training des Filters

Beim Einsatz lernfähiger Spamfilter ist ein Training des Filters notwendig. Im Rahmen des Trainings wird dem Spamfilter eine Menge bekannter Beispiele (der *Korpus*) übergeben, die vom Filter unter Überwachung bearbeitet werden. Anhand dieses Korpus werden nun Statistiken angelegt, die zum Training und für weitere Filterentscheidungen verwendet werden können. Im Rahmen der Spamfilterung können verschiedene Arten des Trainings des Filters unterschieden werden [Yer05, Zdz05].

Bei der Methode $Train - Everything$ (TEFT) wird der Spamfilter mit jeder eintreffenden E-Mail trainiert. Der Vorteil dieses Verfahrens besteht darin, dass durch konsequente Überwachung der Filterentscheidung eine sofortige Anpassung an das E-Mail Verhalten des Benutzers möglich ist. Der Nachteil dieses Verfahrens ist, dass dieses Verfahren auf jede kleinste Veränderung sofort reagiert und somit sehr unbeständig ist. Dies kann vor allem dann zu Fehlern führen, wenn die Anzahl der Spam-Mails im Vergleich zu den Ham-Mails sehr hoch ist.

Bei Verwendung der Methode $Train-on-Error$ (TOE) erfolgt nur dann ein Training des Filters, wenn ein Fehler auftritt. Im Rahmen des Trainings wird die E-Mail nun korrekt klassifiziert. Bei der Methode $Train - Until - Mature$ (TUM) wird immer dann ein vollständiges Training gestartet, wenn ein Fehler auftaucht. Wurde eine E-Mail einmal klassifiziert, so wird diese Entscheidung nur dann geändert, wenn ein Fehler auftaucht. Im Rahmen dieser Trainingsmethode entwickelt sich der Filter, bis keine Fehler mehr auftauchen.

Bei der Methode $Train-Until-No-Errors$ (TUNE) wird der Filter so lange trainiert, bis keine Fehler mehr auftauchen. Der Trainingsvorgang wird manuell gestartet und beinhaltet ein Training des vollständigen Mailkorpus. Bei der Methode $Train - Until - Exhaustion$ (TUE) wird ein Text nach dem Training neu klassifiziert um festzustellen, ob eine korrekte Klassifizierung stattgefunden hat. Dieser Vorgang wird wiederholt, bis der Text korrekt gelernt wurde.

4.3 Simulation

Zur Simulation des Filters werden die Ergebnisse des Trainings aufgearbeitet, miteinander kombiniert und verglichen. Ziel der Simulation ist die Bewertung des Filters in Bezug auf verschiedene Faktoren. Dies betrifft zum Beispiel die Filterrate, als auch die Performanz.

Zunächst ist es notwendig, die Korrektheit des Filters zu messen. Dies be-

trifft die Darstellung falsch-positiver und falsch-negativer, sowie der korrekt erkannten E-Mails. Die Korrektheit betrifft das gesamte Training und kann als Zusammenfassung der Trainingsergebnisse gewertet werden. Eine weitere Möglichkeit der Simulation besteht im Loggen der Systemzeit. Anhand der Berechnungszeit kann die Performanz des Filters abgeschätzt werden. Die Performanz betrifft vor allem den Umgang des Filters mit großen Mengen von E-Mails. Filter mit geringer Performanz sind für den Einsatz in der Praxis nur bedingt geeignet.

Im Rahmen der Simulation sind weiterhin die Lernkurven interessant. Lernkurven stellen den prozentualen Anteil korrekt bzw. falsch klassifizierter E-Mails in Abhängigkeit von der Korpusgröße dar und können Aussagen über Filterraten treffen. Mit steigender Korpusgröße sollten die Werte gegen einen Grenzwert konvergieren. Schließlich wird im Rahmen des Trainings mit verschiedenen Parametern gearbeitet. Ein Vergleich der mit den Parametern erzielten Ergebnisse kann zur Verbesserung der Performanz und der Erkennungsraten führen.

Kapitel 5

Informationstheorie

Ein geschriebener Text besteht zunächst nur aus einzelnen Zeichen; der Gesamtzusammenhang, das heißt der semantische Aspekt, ergibt sich durch Interpretation. Bei der Kommunikation können Probleme auf drei Ebenen unterschieden werden [Sha01]:

Level A: Wie exakt können Kommunikationssymbole übertragen werden? (technisches Problem)

Level B: Wie präzise übermitteln die übertragenen Informationen den erwarteten Sinn? (semantisches Problem)

Level C: Wie beeinflussen die empfangenen Informationen das Verhalten in erwünschter Weise? (Effektivitätsproblem)

Die Informationstheorie befasst sich nicht mit der Übertragung von (semantischen) Informationen, sondern mit der Übertragung von Symbolen, die Informationen beinhalten. Im Gegensatz zum semantischen Aspekt wird der Begriff der Information in der Informationstheorie als gezielte Aufhebung von Ungewissheit definiert und durch den Begriff der Entropie beschrieben. Es wird also auf das technische Problem abgestellt.

5.1 Kommunikationssysteme

Ein Kommunikationssystem kann (abstrakt) als ein System verstanden werden, das sich, wie in Abbildung 5.1 dargestellt, aus fünf Teilen zusammensetzt [Sha53, Sha01].

Eine *(diskrete) Informationsquelle* produziert Nachrichten oder Sequenzen von Zeichen, die dem Empfänger-Terminal mitgeteilt werden sollen. Dazu werden nacheinander einzelne Symbole ausgewählt, deren Auswahl von

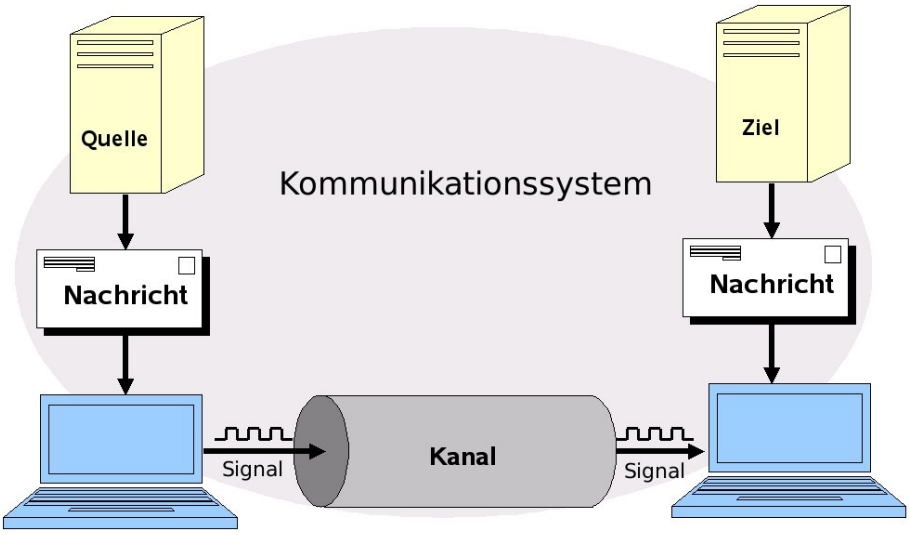

Abbildung 5.1: *Schematisches Darstellung eines Kommunikationssystems*

ihrer Auftrittswahrscheinlichkeit und ihrer Bedingungen in Bezug auf die Kombination von Symbolen abhängen kann. Sie kann als stochastischer Prozess realisiert werden. Ein *Sender* bearbeitet die Nachricht derart, dass sie zur Übertragung über den Übertragungskanal geeignet ist. Dazu werden die Symbole auf eine geeignete Art und Weise codiert. Die Codierung wird durch die Auftrittswahrscheinlichkeit der Symbole und Beschränkungen in Bezug auf deren Kombinationsmöglichkeiten beeinflusst. Mittels eines *Kanals* wird das erzeugte Signal vom Sender zum Empfänger übertragen. Da die Kanalkapazität in der Regel beschränkt ist, ist eine Darstellung der Daten mit möglichst wenigen Zeichen wünschenswert. Der *Empfänger* arbeitet das Signal als Nachricht auf und übergibt diese dem *Ziel*.

5.2 Stochastische Prozesse

In der Regel sind die einzelnen Wahrscheinlichkeiten bei der Auswahl einer Nachricht voneinander abhängig. Ein System, das Symbolsequenzen nach bestimmten Wahrscheinlichkeiten erzeugt, wird als stochastischer Prozess bezeichnet [Sha01].

Bei Berechnung der Auftrittswahrscheinlichkeiten einzelner Symbole sind mögliche Abhängigkeiten zwischen einzelnen Symbolen oder Symbolsequenzen zu berücksichtigen. So hängt das Auftreten einzelner Symbole oder das von Symbolsequenzen meist von den vorhergehenden Symbolen und Symbolsequenzen ab. Das Fehlen von Abhängigkeiten definiert einen theoretischen

Fall, der in der Praxis nicht auftritt.

5.2.1 Markov-Prozesse

Ein Spezialfall des stochastischen Prozesses ist der Markov-Prozess oder das Markov-Modell. Ein Markov-Modell ist eine Möglichkeit zur Beschreibung eines Prozesses mittels einer Menge von Zuständen. Das Modell beschreibt alle möglichen Pfade innerhalb der Zustände und ordnet den Zustandsübergängen Wahrscheinlichkeiten zu. Die Wahrscheinlichkeit für einen Zustandsübergang, der in der Informationstheorie dem Auftreten eines Ereignisses entspricht, hängt nur vom aktuellen Zustand (und nicht von den vorhergehenden Zuständen) ab [Sha01, RN95]. Eine Informationsquelle kann somit mittels eines Markov-Modells dargestellt werden, indem bei jeder Transition ein Symbol erzeugt wird. Diese informelle Darstellung wird in Definition 5 konkretisiert.

Definition 5 (Markov-Modell) *Ein Markov-Modell beinhaltet eine endliche Anzahl von Zuständen eines Systems $S_1 \dots S_n$ und eine Menge an Übergangswahrscheinlichkeiten $p_i(j)$. Die Übergangswahrscheinlichkeit $p_i(j)$ gibt die Wahrscheinlichkeit für den Übergang von Zustand S_i in Zustand S_j an.*

Einen Spezialfall des Markov-Modells stellt das versteckte Markov-Modell (engl. *Hidden Markov-Modell, HMM*) dar. Im Unterschied zu einem Markov-Modell, dessen Ausgabe eine Sequenz von Zustandsnamen ist, besitzt in einem HMM jeder Zustand eine Wahrscheinlichkeitsverteilung möglicher Ausgaben, so dass die gleiche Ausgabe in mehr als nur einem Zustand auftreten kann. Derartige Modelle werden als versteckt bezeichnet, da der tatsächliche Zustand vor dem Betrachter verborgen bleibt.

5.2.2 Ergodische Prozesse

Ergodische Prozesse sind ein Spezialfall der Markov-Modelle. Ergodische Systeme weisen eine besonders sichere Art statistischer Gleichmäßigkeit auf die sich dadurch äußert, dass jede erzeugte Sequenz die gleiche Wahrscheinlichkeit hat. Ergodizität impliziert somit statistische Homogenität. Erfolgt eine Darstellung des ergodischen Prozesses als Graph, der die Transitionen der Zustände modelliert, so handelt es sich um einen zusammenhängenden Graph, das heißt es gibt keine isolierten Teilgraphen. Weiterhin ist der größte gemeinsame Teiler aller Schlingen, das heißt aller zusammenhängenden Wege des Graphen, gleich 1.

Definition 6 *Sei P_i die Wahrscheinlichkeit für das Auftreten des Zustands i und $p_i(j)$ die Wahrscheinlichkeit für die Transition von Zustand i in Zustand j, dann muss gelten [Sha01]:*

$$P_j = \sum_i P_i \, p_i(j) \tag{5.1}$$

Es kann gezeigt werden, dass jeder Zustand j nach N Symbolen mit der Wahrscheinlichkeit $P_j(N)$ erreicht werden kann [Sha01].

5.3 Information und Entropien

In der Informationstheorie wird die Information als aufgehobene Unsicherheit definiert und als Logarithmus zur Basis 2 beschrieben. Die Information eines Zeichens gibt die minimale Anzahl von Bits an die benötigt wird, um das betroffene Zeichen darzustellen. Der mittlere Informationsgehalt eines Zeichens wird mittels der Entropie angegeben, die den Grad der Zufälligkeit eines Zustands bezeichnet. Demgegenüber stellt die Redundanz den Teil der Nachricht dar, der durch deren Struktur vorgegeben ist [Ver98, Sha01].

5.3.1 Entropien

Die Entropie bezeichnet den mittleren Informationsgehalt und wird als Ausdruck über die verschiedenen Wahrscheinlichkeiten dargestellt. Sie kann als Grad der Zufälligkeit verstanden werden: Ist eine Situation stark geordnet, so ist der Grad der zufälligen Entscheidungen gering und die Information (oder deren Entropie) niedrig. Die Entropie gibt also die Menge produzierter Informationen (in diesem Zustand) an.

An eine Funktion zur Darstellung der Entropie werden verschiedene Anforderungen gestellt [Sha01]. So muss es sich um eine stetige Funktion handeln, die für gleich große p_i ($p_i = 1/n$) monoton ist. Bei mehreren zur Auswahl stehenden Ereignissen muss H der gewichteten Summe der einzelnen Werte entsprechen. Sei n die Anzahl verschiedener zur Verfügung stehender Symbole und $p_1, \ldots, p_n$ deren Wahrscheinlichkeiten, so berechnet sich die Entropie H wie in Funktion 5.2 dargestellt [Sha01, BK90].

$$H = -\sum p_i \; log \; p_i \tag{5.2}$$

Sind alle Wahrscheinlichkeiten gleich groß, so wird das Maß der Information und somit die Entropie maximal. Weiterhin steigt die Entropie mit der Anzahl der zur Verfügung stehenden Symbole.

Relative Entropien

Das Verhältnis der aktuellen zur maximalen Entropie wird als relative Entropie oder Kullback-Leibler-Divergenz bezeichnet und beschreibt den Grad der Freiheit in Bezug auf die Wahl der Symbole der Nachricht einer Quelle. Beträgt die relative Entropie einer Quelle zum Beispiel 0,8, so ist die Nachricht in Bezug auf die Wahl der Symbole der Nachricht zu 80% frei. Die relative Entropie zweier Quellen A und X mit zugehörigen Wahrscheinlichkeitsverteilungen p_a und q_x berechnet sich wie in Funktion 5.3 dargestellt.

$$D(A||X) = \sum_{a,x} p_a \, log \, \frac{p_a}{q_x} \tag{5.3}$$

Die relative Entropie gibt die Anzahl von Bits an, die zusätzlich zur Codierung benötigt wird, um die in X enthaltenen Daten mit der für A optimalem Codierung darzustellen [Kal04, Sha01].

Entropien von Informationsquellen

Für jeden Zustand i gibt es eine Entropie H_i, aus deren gewichteter Durchschnitt sich die Entropie H der Informationsquelle ergibt. Die Gewichtung erfolgt in Bezug auf die Wahrscheinlichkeiten P_i des Auftretens der Zustände. Sei $p_i(j)$ die Menge von Auftrittswahrscheinlichkeit für das Auftreten des Symbols j, so berechnet sich die Entropie der Informationsquelle wie in Funktion 5.4 dargestellt [Sha01].

$$\begin{aligned} H &= \sum_i P_i \, H_i \\ &= -\sum_{i,j} P_i \, p_i(j) \log p_i(j) \end{aligned} \tag{5.4}$$

Die Entropie H der Quelle gibt die Menge an Informationen pro Symbol der Quelle an [Sha01, Ver98]. Sind die einzelnen Symbole voneinander unabhängig, so entspricht die Entropie der Quelle $-\sum p_i \log p_i$, wenn p_i der Wahrscheinlichkeit für Symbol i entspricht [Sha01].

5.3.2 Redundanz

Bei der Redundanz handelt es sich um die Struktur der Nachricht, die nicht durch die freie Entscheidung des Senders bestimmt wird, sondern vorgegeben ist. Eine derartige Vorgabe kann zum Beispiel durch die gewählte Sprache

vorliegen. Die Redundanz stellt ein Maß der Differenz zwischen der durchschnittlichen Codewortlänge und dem durchschnittlichen Informationsgehalt dar [Ver98, Sha01]. Sei p_i die Wahrscheinlichkeit für das Auftreten einer Sequenz x_i und l_i deren Länge, so kann die Redundanz $\mathcal{R}$ formell wie in Formel 5.5 bestimmt werden [Ver98, Sha01].

$$\mathcal{R} \quad = \quad \sum p_i l_i - \sum (-p_i \, log \, p_i) \tag{5.5}$$

Die Redundanz in Englisch beträgt zum Beispiel rund 50%, das heißt in etwa die Hälfte der Buchstaben oder Wörter wird durch die statistische Struktur der Sprache bestimmt. Die Minimierung der durchschnittlichen Codewort-Länge führt zu einer Reduzierung der Redundanz.

Kapitel 6

Datenkompression

Die Datenkompression ist ein Spezialfall der Zeichencodierung. Unter Zeichencodierung versteht man die Darstellung eines Zeichens mittels eines (eindeutigen) Codes. In digitalen Dokumenten werden dazu Binärcodes verwendet, aus denen sich verschiedene Standards entwickelt haben, darunter auch ASCII und UNICODE.

Ziel der Datenkompression ist die Minimierung der zur Darstellung einer Nachricht benötigten Bits. Dies ist notwendig, da die Kapazitäten zur Speicherung oder Übertragung einer Nachricht in der Regel beschränkt sind. Um die Darstellungsgröße einer Nachricht zu minimieren, können verschiedene Formen der Redundanz, das heißt sich wiederholende Muster im Eingabestrom, ausgenutzt werden. Die einzelnen Kompressionsverfahren unterscheiden sich, abhängig von den zugrundeliegenden Daten, im Umgang mit der Redundanz.

6.1 Zeichencodierung

Bevor verschiedene Varianten der Zeichencodierung und die damit verbundenen Abbildungseigenschaften erläutert werden muss geklärt werden, was ein Code überhaupt ist und welche Eigenschaften er hat.

Definition 7 (Code) *Ein Code $\mathcal{C}$ ist eine Zuordnung von Nachrichten einer Quelle (Wörter des Quellalphabets α) auf Codewörter (Wörter des Code-Alphabets β). Es gilt:*

$$\mathcal{C} = \alpha \rightarrow \beta$$

Bei einem Code handelt es sich also zunächst nur um eine Abbildungsvorschrift von einer Nachricht auf ein Codewort [LH87]. Bei einer Nachricht kann

es sich sowohl um ein einzelnes Symbol als auch um eine Symbolfolge handeln. Darüber hinaus können verschiedene Arten von Codes unterschieden werden. Ein Block-Block Code bildet Nachrichten fester Länge auf Codewörter fester Länge ab. Ein Beispiel für die Block-Block Codierung ist ASCII, das ein Alphabet von 64 einzelnen Zeichen auf Codewörter mit einer Länge von 7 Bit abbildet. Ein Block-Variabel Code bildet Nachrichten fester Länge auf Codewörter variabler Länge ab, ein Variabel-Block Code bildet Nachrichten variabler Länge auf Codewörter fester Länge und ein Variabel-Variabel Code bildet Nachrichten variabler Länge auf Codewörter variabler Länge ab. Eine weitere wichtige Eigenschaft von Codes ist deren Unterscheidbarkeit, wie in Definition 8 dargestellt.

Definition 8 (unterscheidbare Codes) *Ein Code ist unterscheidbar bzw. verschieden, wenn sich jedes Codewort von den anderen unterscheidet.*

Bei einem unterscheidbaren Code ist es somit nicht möglich, dass zwei Zeichen mit dem gleichen Codewort codiert werden. Um eine korrekte Dekompression zu gewährleisten, müssen Codes jedoch nicht nur unterscheidbar, sondern auch eindeutig decodierbar sein.

Definition 9 (eindeutig decodierbare Codes) *Ein unterscheidbarer Code ist eindeutig decodierbar, wenn jedes Codewort eindeutig identifiziert werden kann.*

Bei eindeutig decodierbaren Codes ist somit kein Codewort Teil eines anderen. Alle eindeutig decodierbaren Codes sind Präfix-Codes, das heißt kein Codewort ist Präfix eines anderen Codes.

6.2 Redundanz

Die Aufgabe der Datenkompression besteht in der Reduzierung des technischen Aufwands bei der Übertragung und Speicherung von Daten, um die benötigten Leitungs- und Speicherkapazitäten zu senken. Dies wird durch eine Codierung bei möglichst geringer Bitrate erreicht. Verfahren, die die Redundanz verschiedener Daten ausnutzen, codieren die einzelnen Zeichen und Sequenzen der Datei auf eine möglichst gute Art und Weise, um eine Reduzierung der Größe der betroffenen Datei zu erreichen.

Die verschiedenen Kompressionsalgorithmen unterscheiden sich darin, wie sie mit dem Auftreten von Redundanz, das heißt mit wiederkehrenden Mustern im Eingabestrom, umgehen. Der Umgang mit der Redundanz gibt die Güte eines Kompressionsalgorithmus wieder. Bedingt durch verschiedene zu

komprimierende Quellen unterscheiden wir vier Arten von Redundanz, die von Kompressionsalgorithmen zum Reduzieren der Dateigröße ausgenutzt werden können [Wel84].

Häufigkeit von Zeichen: In einem geschriebenen Text treten bestimmte Zeichen häufiger auf als andere; einige Zeichenfolgen finden sich in geschriebenen Texten gar nicht. Diese Verteilung kann für die Codierung ausgenutzt werden, da die Anzahl der zur Codierung notwendigen Bits von der Häufigkeit der Zeichen abhängig ist.

Wiederholung von Zeichen: Vor allem in Grafiken, Tabellen oder Listen tritt ein Zeichen mehrfach hintereinander auf. In diesem Fall kann diese Wiederholung auf eine sinnvolle Weise ausgenutzt werden, um die Dateigröße zu reduzieren.

Oft verwendete Zeichenketten: In Abhängigkeit von der zugrundeliegenden Datei treten bestimmte Zeichenfolgen häufiger auf als andere und sollten durch möglichst wenige Bits repräsentiert werden, um die Effizienz der Komprimierung zu steigern. Seltener auftretende Zeichenfolgen können im Gegenzug mit längeren Bitkombinationen dargestellt werden.

Positionierungsredundanz: Einige Zeichen oder Zeichenketten treten oft an bestimmten Stellen eines Dokuments auf. Diese partielle Redundanz kann für die effiziente Komprimierung ausgenutzt werden.

6.3 Grundlagen der Datenkompression

Die verschiedenen Verfahren zur Datenkompression können anhand verschiedener Eigenschaften unterschieden und klassifiziert werden. Dies betrifft zunächst die Unterscheidung in verlustbehaftete und verlustfreie Verfahren.

Definition 10 (Verlustbehaftete Kompression) *Bei der verlustbehafteten Kompression tritt nach der Dekompression ein partieller Datenverlust auf.*

Definition 11 (Verlustfreie Kompression) *Bei der verlustfreien Kompression stellen die dekomprimierten Daten eine exakte Reproduktion der ursprünglichen Eingabedaten dar.*

Ist die exakte Informationswiedergabe wichtig, wie zum Beispiel bei der Kompression von Texten, so wird die verlustfreie Kompression eingesetzt.

Verlustbehaftete Kompressionsverfahren finden zum Beispiel bei der Audio- und Video-Kompression Einsatz. Aufgrund der erlaubten Fehlerquote wird eine höhere Kompression als mittels verlustfreier Kompressionsverfahren ermöglicht.

Weiterhin können zur Datenkompression sowohl statische als auch dynamische Verfahren eingesetzt werden. Statische Verfahren zeichnen sich grundsätzlich dadurch aus, dass sich das Verfahren im Laufe der Kompression nicht ändert, während bei dynamischen (adaptiven) Verfahren im Laufe der Kompression eine Anpassung auf die zugrundeliegenden Datensätze erfolgen kann [LH87].

Definition 12 (statische Methode zur Datenkompression) *Bei der statischen Methode zur Datenkompression wird eine feste Zeichenfolge der Nachricht immer durch das gleiche Codewort repräsentiert.*

Definition 13 (adaptive Methode zur Datenkompression) *Bei der dynamischen Methode zur Datenkompression ist es möglich, dass sich die Codierung der Zeichenketten im Laufe der Zeit ändert.*

Möglich ist auch der Einsatz hybrider Techniken, die k verschiedene Codebooks für statische Codes verwenden. Ein Beispiel für eine hybride Technik ist die LZ-Kompression, die nach einer Zeit t eine neue (angepasste) Codierung für die zu komprimierenden Zeichenketten erzeugt.

6.4 Probleme der Datenkompression

Ziel der Datenkompression ist die Minimierung der zur Darstellung einer Nachricht notwendigen Bits zur besseren Ausnutzung der Speicher- oder Kanalkapazitäten. Die Datenkompression ist somit eng mit der Informationstheorie (*siehe* Kapitel 5) verbunden. Grundsätzlich können demnach alle Ansätze der Datenkompression auf zwei Probleme der Informationstheorie reduziert werden: Ansätze zur Modellierung der Statistiken der Quelle und die Darstellung einer bestimmten Nachricht der Quelle mit einer möglichst geringen Anzahl an Bits [CW84]. Die Codierung erfolgt unter Zuhilfenahme der Ergebnisse aus der Modellierung der Statistiken der Quelle.

6.4.1 Modellierung der Statistiken der Quelle

Die Statistiken der Quelle können sowohl mittels Wörterbüchern, als auch mittels Markov-Ketten modelliert werden. Während Wörterbücher eine Liste der im Text vorkommenden Muster erstellen, modelliert das Markov-Modell

Abhängigkeiten zeitlich aufeinanderfolgender Ereignisse und somit den konkreten zugrundeliegenden Kontext. Ziel ist es, einen Mechanismus zur Verfügung zu stellen, um im nächsten Schritt eine möglichst gute Codierung zu erreichen. Dazu werden die Statistiken der Quelle genutzt.

Wörterbuchverfahren

Bei der textbasierten Datenkompression werden in der Regel Wörterbuchverfahren eingesetzt. Dazu wird eine Liste der im Text vorkommenden Muster konstruiert, die als Indizes der Liste codiert werden. Analog zu den Methoden der Datenkompression unterscheiden wir statische und dynamische (adaptive) Wörterbuchverfahren.

Definition 14 (statische Wörterbuchverfahren) *Bei statischen Wörterbuchverfahren erfolgt die Kompression der Daten unter Zuhilfenahme bereits vorhandener Wörterbücher.*

Statische Verfahren verwenden ein festes Wörterbuch, in dem alle möglichen Eingabezeichen, sowie häufige Zeichenpaare enthalten sind. Es handelt sich zwar um einen einfachen Algorithmus, da das Wörterbuch jedoch nicht auf die zu komprimierenden Daten angepasst wird, ist die Kompressionsrate nur gering.

Definition 15 (adaptive Wörterbuchverfahren) *Bei dynamischen (adaptiven) Wörterbuchverfahren wird das Wörterbuch während der Kompression der Daten erzeugt.*

Adaptive Verfahren werden eingesetzt, wenn keine oder nur wenige Informationen über den zu komprimierenden Text vorliegen. Derartige Verfahren beginnen in der Regel mit einem leeren Wörterbuch und erstellen die Einträge abhängig von den zu komprimierenden Daten. Das Wörbuch wird implizit zusammen mit den komprimierten Daten gespeichert. Adaptive Verfahren werden von den Algorithmen der LZ-Familie realisiert.

Markov-Modellierung

Grundlage der Markov-Modellierung bildet ein (verstecktes) Markov-Modell (engl. *Hidden Markvo-Modell*, HMM), wie in Definition 16 dargestellt. Wie bereits in Abschnitt 5.2.1 dargestellt, besitzt das HMM für jeden Zustand eine Wahrscheinlichkeitsverteilung möglicher Ausgaben. Der konkrete Zustand des Modells bleibt vor dem Betrachter verborgen.

Definition 16 (Verstecktes Markov-Modell) *Ein (verstecktes) Markov-Model wird mittels eines 5-Tupels (Q, V, π_i, A, B) dargestellt:*

- $Q = \{q_1, \ldots, q_k\}$ *die Menge der Zustände,*
- $V = \{v_1, \ldots, v_m\}$ *das Ausgabealphabet*
- π_i *die Wahrscheinlichkeit, sich zur Zeit* $t = 0$ *im Zustand* q_i *zu befinden*
- $A = \{a_{ij}\}$ *die Wahrscheinlichkeiten für den Zustandsübergang von Zustand* i *in Zustand* j *(die Markov-Wahrscheinlichkeit) und*
- $B = \{b_j(n)\}$ *die Ausgabewahrscheinlichkeiten*

Mittels eines Markov-Modells können stochastische Abhängigkeiten zeitlich aufeinanderfolgender Ereignisse beschrieben werden [Shk02]. Wahrscheinlichkeiten des n-ten Elements werden als bedingte Wahrscheinlichkeiten über die vorherigen k Elemente bestimmt. Die Wahrscheinlichkeit für das Auftreten eines Symbols wird also anhand des aktuellen Kontext berechnet. Ist die Menge der betrachteten k Elemente auf D beschränkt ($k \leq D$), so wird das zugrundeliegende Modell als Markov-Modell der Ordnung D bezeichnet. Ein solches Modell kann als $m-$ärer Baum der Ordnung D dargestellt werden.

Um das HMM für die Datenkompression zu nutzen, werden in jedem Zustand die Auftrittswahrscheinlichkeiten für die folgenden Zeichenketten berechnet. In der Regel wird hier ein Markov-Modell der Ordnung D verwendet, das heißt die Auftrittswahrscheinlichkeiten werden nur für die nächsten D Zeichen berechnet. Diese Auftrittswahrscheinlichkeiten können genutzt werden, um die Codierung der Zeichen an die Begebenheiten anzupassen.

6.4.2 Darstellung der Nachricht

Zur Darstellung der Nachricht kann sowohl die Huffman-Codierung, als auch die arithmetische Codierung eingesetzt werden [BA06]. Die arithmetische Codierung stellt eine Generalisierung der Huffman-Codierung dar. Beide Varianten der Codierung arbeiten auf den Auftrittswahrscheinlichkeiten der Symbole, die sie mittels der in Abschnitt 6.4.1 dargestellten Verfahren erhalten. Grundsätzlich werden Zeichen, die eine geringe Auftrittswahrscheinlichkeit haben mit mehr Bits codiert als solche mit geringerer Auftrittswahrscheinlichkeit.

Huffman-Codierung

Die Huffman-Codierung ist ein Codierungsalgorithmus für verlustfreie Datenkompression. Eingabezeichen fester Größe werden in Symbole variabler Länge transformiert. Die Tabelle für die Kompression wird anhand der Auftrittswahrscheinlichkeit P der einzelnen Symbole erzeugt. Jedes Symbol wird mit $log_2(P)$ Bits codiert. Die Größe der Eingabesymbole wird durch die Größe der für die Kompression benötigen Tabelle beschränkt. Kleinere Tabellen beschränken den größtmöglichen Grad der Kompression, während größere Tabellen sehr viel mehr Speicherplatz benötigen.

Die Decodierungstabelle der Huffman-Codierung ist als binärer Baum aufgebaut, der für jedes zu decodierende Symbol durchsucht werden muss. Decodierung mit variabler Symbol-Länge kann bei großem Datenaufkommen zu Problemen mit Kosten und Performanz führen. Weiterhin besteht die Notwendigkeit, Kenntnis über die Auftrittswahrscheinlichkeiten der Symbole zu erhalten. Die Auftrittswahrscheinlichkeiten der Symbole variieren abhängig von der Art der Daten. Um dieses Problem zu lösen, werden in der Regel alle Datenblöcke einzeln analysiert, um die Zeichenverteilung dieser Blöcke zu bestimmen.

Arithmetische Codierung

Die arithmetische Codierung stellt eine Generalisierung der Huffman-Codierung dar und basiert auf der Auftrittswahrscheinlichkeit einzelner Symbole [CW84]. Ist die Auftrittswahrscheinlichkeit jedes Symbols bekannt, so kann man einzelne Zeichenketten durch eindeutige Identifikatoren ersetzen. Diese Identifikatoren entstammen dem Intervall [0,1]. Je höher die Auftrittswahrscheinlichkeit eines Symbols, umso weniger Bit werden zur Darstellung verwendet.

Bei der arithmetischen Codierung wird eine Sequenz $X_1 \dots X_n$ in eine neue Sequenz $Y_1 \dots Y_m$ transformiert. Nach einer Sequenzlänge $n = i$ werden die möglichen Werte der X_i willkürlich geordnet und den einzelnen Werten Wahrscheinlichkeiten $p(w_1) \dots p(w_m)$ zugeordnet. Die Codierung von Y wird ausschließlich anhand dieser Wahrscheinlichkeiten durchgeführt.

6.5 Verlustfreie Kompressionsstrategien

Im Rahmen der verlustfreien Kompressionsstrategien können verschiedene Klassen von Algorithmen unterschieden werden. Eine große Klasse der Kompressionsstrategien wird durch die Algorithmen der LZ-Familie gebildet, die auf Wörterbuchverfahren basieren. Eine zweite Klasse von Algorithmen ist

das PPM-Schema (engl. *Prediction by Partial Matching*), das auf (versteckten) Markov-Modellen basiert [BA06].

6.5.1 Algorithmen der LZ-Familie

1977 und 1978 wurden von Abraham Lempel und Jacob Ziv die Kompressionsverfahren LZ77 und LZ78 veröffentlicht, die mittlerweile die Basis für eine ganze Familie von Kompressionsalgorithmen bilden. Die Algorithmen der LZ-Familie gehören zur Gruppe der verlustfreien, universellen Kompressionsverfahren. Unter Verwendung dynamischer Wörterbücher werden Nachrichten fester Länge auf Codewörter fester Länge abgebildet. Das Wörterbuch wird bei der Dekodierung aus den komprimierten Daten rekonstruiert.

Algorithmen der LZ-Familie benötigen zunächst keine Informationen über die zu komprimierende Quelle, da der Komprimierungsprozess mit einem Lernprozess verbunden ist. Aufgrund des Lernprozesses werden ausreichend lange Sequenzen benötigt, um eine gute Kompressionsrate zu erzielen. Geht die zu komprimierende Sequenz gegen unendlich, so ist die LZ-Codierung nahezu optimal, das heißt die Redundanz geht gegen Null. Die Algorithmen der LZ-Familie unterscheiden sich unter anderem durch die Erkennung der redundanten Daten, sowie durch die Erzeugung und Verwaltung des Wörterbuches. Grundsätzlich können alle Algorithmen der LZ-Familie in eine von zwei Gruppen eingeteilt werden.

Algorithmen der ersten Gruppe repräsentieren das Wörterbuch durch die bereits verarbeitete Datenmenge. Bereits im Wörterbuch enthaltene Zeichenketten werden durch einen Zeiger auf den betroffenen Eintrag dargestellt. Das Wörterbuch ist somit implizit in den komprimierten Daten enthalten. Zu dieser Gruppe gehören alle Varianten des LZ77. Algorithmen der zweiten Gruppe erzeugen bei der Komprimierung und bei der Dekomprimierung ein Wörterbuch aus den Zeichenfolgen des Wörterbuches. Dieses Wörterbuch wird nicht gespeichert, sondern kann aus den komprimierten Daten rekonstruiert werden. Alle Algorithmen des LZ78 gehören dieser Gruppe an.

LZ77

LZ77 verwendet adaptive Wörterbücher, die implizit mit den komprimierten Daten gespeichert werden [LZ77]. Dazu wird ein Fenster verwendet, das aus einem Vorschaupuffer, dem Lexikon und einem Textfenster, dem zu komprimierenden Text, besteht. Der Vorschaupuffer enthält die bereits komprimierten Teile des Textes, während das Textfenster den noch zu komprimierenden Text enthält. Zur Komprimierung mittels LZ77 wird die längste passende Zeichenkette des Textfensters im Vorschaupuffer gesucht. Das Ergebnis wird

als 3-Tupel gespeichert, das die Adresse der bereits enthaltenen Sequenz, deren Sequenzlänge, sowie das erste abweichende Zeichen enthählt. Ein nicht im Vorschaupuffer enthaltenes Zeichen wird mit Adresse und Sequenzlänge 0 angegeben.

Die Güte der Kompression ist direkt vom Lexikon abhängig, das durch den Vorschaupuffer repräsentiert wird. Das Fenster muss also ausreichend groß sein. Ein zu großer Vorschaupuffer verringert die Performanz, da immer das gesamte Lexikon auf passende Einträge geprüft werden muss. Die Dekomprimierung hingegen ist effizienter durchführbar. Weiterhin hängt die Güte der Kompression von der Codierung ab. Zur Codierung eines Zeichens als 3-Tupel werden dreimal mehr Bits benötigt als zur Darstellung eines einzelnen Zeichens. Codiergewinne ergeben sich somit nur, wenn mehr als zwei Zeichen in einem 3-Tupel komprimiert werden können. Die Gesamtlänge der Komprimierung kann also nur dann reduziert werden, wenn die Codiergewinne die Codierverluste kompensieren können.

Vorteile sind vor allem die einfache Erweiterbarkeit, der große Spielraum für Verbesserungen, aber auch die Möglichkeit, mit einfachen Verbesserungen sehr gute Ergebnisse zu erzielen. Dies betrifft vor allem den guten Umgang mit Veränderungen der Redundanz im Laufe der Zeit. Es ergeben sich jedoch auch einige Nachteile. So wird nur ein relativ kleines Fenster als Lexikon verwendet. Nicht im Fenster befindliche Einträge des Lexikons können auch nicht für Vergleiche herangezogen werden. Tritt nur geringe Redundanz auf, so ist auch die Kompressionsrate gering. Weiterhin muss für jedes zu komprimierende Zeichen der gesamte Vorschaupuffer durchsucht werden, was zu Performanzproblemen führen kann.

6.5.2 Prediction by Partial Matching (PPM)

Prediction by partial matching (PPM) verwendet eine Kombination aus einem Markov-Modell der Ordnung D und arithmetischer Codierung [CW84]. Grundlage bilden immer Zeichenketten oder Wörter. Ziel des PPM ist die Vorhersage des nächsten Zeichens anhand bereits bekannter Zeichensequenzen. Bei Verwendung eines Markov-Modells der Ordnung D wird die Vorhersage des nächsten Zeichens einer Zeichenkette der Länge D anhand bereits bekannter Sequenzen getroffen. Zur Vorhersage eines Zeichens eines Modells niedriger Ordnung D wird somit ein Modell hoher Ordnung K, das heißt alle bereits bekannten Sequenzen, verwendet.

Das Modell erfährt mit jeder betrachteten Sequenz eine Aktualisierung. Mit steigendem Kontext verbessert sich die Vorhersage der Wahrscheinlichkeiten, da die Ordnung K des Markov-Modells für die Vorhersage steigt. Zur Behandlung neuer Zeichenketten oder unbekannter Zeichen werden spezielle

Escape-Sequenzen verwendet, um diesen Kombinationen eine zunächst feste Auftrittswahrscheinlichkeit zuzuordnen (in der Regel 1/128).

6.6 Implementierung der Kompressionsstrategien

Die in 6.5 dargestellten Kompressionsstrategien finden in einer Vielzahl von Verfahren Implementierung. Speziell für die Verarbeitung von Texten sind die Kompressionsalgorithmen gzip und rar gedacht. Während es sich bei gzip um eine freie Implementierung des LZ77 handelt, stellt rar ein proprietäres Dateiformat dar, das auf PPM basiert.

6.6.1 gzip

gzip ist ein verlustfreier Kompressionsalgorithmus, der auf dem Deflation-Algorithmus basiert und eine Variation des LZ77 in Kombination mit der Huffman-Codierung darstellt [Deu96, GA06]. Mittels LZ77 erfolgt zunächst eine Kompression der mehrfach auftretenden Zeichenfolgen, anschließend wird das Resultat mittels der Huffman-Codierung codiert. Der Algorithmus ist frei von patentierten Algorithmen und Bestandteil der GPL (General Public License). Zur Kompression der Eingabedaten wird LZ77 verwendet, der bei der Suche nach mehrfach auftretenden Zeichenketten auf die vorhergehenden 32 KByte zurückgreifen kann. Mehrfach auftretende Zeichenketten werden unter Zuhilfenahme mehrstufiger Hash-Tabellen gesucht. Dazu wird eine Hash-Funktion verwendet, die auf 3-Byte-Sequenzen operiert. Ist eine Zeichenfolge länger als 3 Byte, so werden die folgenden Bytes in einer weiteren Hashtabelle abgelegt.

Zur Steigerung der Performanz verwendet gzip ein so genanntes *lazy matching*. Nach einem Treffer der Länge N wird nach einer längeren passenden Zeichenkette gesucht, indem mit dem nächsten Eingabebyte begonnen wird. Konnte eine längere Zeichenkette gefunden werden, so wird das erste Eingabebyte als Treffer der Länge 1 und die längere Zeichenkette separat gespeichert. Ansonsten wird die erste Zeichenkette gespeichert.

Die Performanz kann mittels verschiedener Laufzeitparameter kontrolliert werden. Dies betrifft vor allem das *lazy matching*. Ist die Kompressionsrate wichtig, so wird nach jedem Treffer eine vollständige zweite Suche durchgeführt. Eine vollständige zweite Suche geht jedoch auf Kosten der Laufzeit, so dass die Suche in der Regel nach einem längeren Treffer reduziert wird. Eine weitere Verbesserung der Laufzeit kann außerdem dadurch erreicht werden,

dass nur dann neue Zeichenketten in die Hash-Tabelle aufgenommen werden, wenn kein Treffer gefunden wurde oder der gefundene Treffer nicht lang genug ist. In diesem Fall wird die Kompressionsrate zugunsten der Laufzeit reduziert.

6.6.2 rar

Bei rar handelt es sich um eine Kombination auf Huffman-Codierung, LZ77 und PPM (Prediction with Partial Matching) [RKB06, rar06, Win06]. rar besitzt ein proprietäres Dateiformat, bei dem lediglich die Schnittstellen für die Datendekompression dokumentiert und quelloffen verfügbar sind. Durch spezielle Algorithmen werden mittels rar alle Daten zusammen komprimiert, indem vor der Kompression alle Dateien in eine einzige kopiert werden. Der Vorteil besteht darin, dass Redundanzen, die sich über mehrere Dateien erstrecken, berücksichtigt werden können. Dazu wird ein Wörterbuch mit einer maximalen Größe von 4096KB verwendet. Ein großer Nachteil des rar besteht in dessen Fehleranfälligkeit, da das Archiv für die Dekompression als Ganzes intakt sein muss. Neben der Kompression ist es mittels `rar` außerdem möglich, die komprimierten Daten zu verschlüsseln. Dazu wird eine 128-Bit AES-Verschlüsselung verwendet.

Das Kompressionsverfahren rar unterscheidet sechs verschiedene Kompressionsparameter [Ros06, rar06]. Die erste Methode ($m0$, store) führt keine Kompression durch, sondern führt die ausgewählten Dateien lediglich dem Archiv zu. Die anderen fünf Methoden ($m1, \ldots, m5$) komprimieren die Daten beim Hinzufügen zum Archiv. Sie unterscheiden sich vor allem aufgrund des erreichten Kompressionsfaktors. Dazu werden für die einzelnen Kompressionsoptionen verschiedene Verfahren verwendet. Für die erste und zweite Kompressionsmethode ($m1, m2$) werden generelle Kompressionsalgorithmen verwendet, für die restlichen Parameter ($m3, \ldots, m5$) werden weiterführende Algorithmen, zum Beispiel für Musik- oder Bildberechnung, eingesetzt. Mit Verwendung des vierten oder fünften Parameters ($m4, m5$) wird außerdem der PPM-Algorithmus zur Kompression von Texten verwendet [Ros06].

Kapitel 7

Problemfelder der Texterkennung

Die Texterkennung beschreibt ein Verfahren zur Bestimmung des Typus eines gegebenen Textes. Ziel ist die Einordnung des gegebenen (neuen) Textes relativ zu weiteren (bekannten) Texten, den Korpora. Es gibt verschiedene Kriterien, anhand derer die Klassifizierung von Texten durchgeführt werden kann [DEV02, BCL03]. So kann eine Zuordnung anhand des Autors oder anhand der Sprache des vorliegenden Textes erfolgen. Eine weitere Möglichkeit besteht darin, die Zuordnung von der Ähnlichkeit der Inhalte abhängig zu machen und diese relativ zueinander anzuordnen.

Sei $X = \{x_1, \ldots, x_n\}$ eine Menge von Kategorien bzw. Klassen und sei $A = \{a_1, \ldots, a_m\}$ der Korpus, das heißt eine Menge von Dokumenten, die als Grundlage für die Klassifizierung dienen. Formell beschreibt die Texterkennung ein Verfahren, um einem Paar $\langle a_i, x_j \rangle$ einen Boole'schen Wert k zuzuweisen, $k \in \{false, true\}$. Dieser Wert gibt die Klassifizierungsentscheidung für die Datei x_i unter der Bedingung a_j an [Seb02]. Formell erfolgt der Versuch, eine konkrete Zielfunktion zu approxminieren, die sich durch Einsetzen in die allgemeine Zielfunktion 7.1 ergibt.

$$\Phi : \{x_1, \ldots, x_n\} \times \{a_1, \ldots, a_m\} \rightarrow \{false, true\} \tag{7.1}$$

Zur Bewertung der Klassifizierungsfunktion wird der initiale Korpus A in eine Trainingsmenge Ω und eine Testmenge Ψ aufgeteilt, so dass gilt $\Omega \cup \Psi = A$. Der initiale Korpus $\Omega = \{a_1, \ldots, a_k\}$ beinhaltet eine Menge bereits klassifizierter Texte, während die Menge $\Psi = \{a_{k+1}, \ldots, a_m\}$ zum Training des Programms verwendet wird. Im Rahmen des Trainings können die Ergebnisse, die mittels der Menge Ψ erreicht werden, protokolliert und zur Bewertung herangezogen werden.

Grundsätzlich können alle Probleme der Klassifizierung und Zuordnung von Texten auf zwei Problemfelder reduziert werden: Die Sequenzklassifizie-

rung und die Sequenzerkennung. Während die Sequenzklassifizierung unbekannte Texte bereits bekannten zuordnet, erfolgt mittels der Sequenzerkennung eine relative Zuordnung bekannter Texte untereinander. Das Problem der Klassifizierung von E-Mails lässt sich auf das Problem der Klassifizierung von Texten zurückführen, dessen Ziel es ist, anhand des Inhalts eines Textes eine Zuordnung zu einer Gruppe bekannter Texte vorzunehmen.

7.1 Sequenzerkennung

Bei der Sequenzerkennung erfolgt die Klassifizierung eines unbekannten Textes relativ zu einem Korpus bereits bekannter Texte. Die Sequenzerkennung beschreibt das Verfahren, eine Menge eintreffender Dokumente in Kategorien einzusortieren. Klassifizierungskriterien können zum Beispiel die Sprache des Textes oder dessen Autor sein. Grundlage der Sequenzerkennung bildet eine Sammlung von Texten, der Korpus, die typisch für das betroffene Klassifizierungsmerkmal ist. Klassifiziert werden soll ein neuer, unbekannter Text. Zur Klassifizierung dieses neuen Textes (X) wird dieser in Bezug auf seinen Informationsgehalt relativ zu den Texten des Korpus (A_i) mit diesen verglichen. Je kleiner der Informationsgehalt, desto geringer sind die inhaltlichen Unterschiede des Textes X und des Vergleichtextes A_i und desto wahrscheinlicher ist der Text X dem Text A_i zugehörig. Sei $x_i \in X$ ein eintreffender Text und $A = \{a_1, \ldots, a_m\}$ eine Menge von Korpora, so beschreibt das Verfahren zur Sequenzerkennung formell die Approximation der in Funktion 7.2 dargestellten Zielfunktion, die sich durch Einsetzen in die allgemeine Zielfunktion 7.1 ergibt. Ziel ist es, den konkreten Text x_i in Bezug auf die Eigenschaften des Korpus A zu klassifizieren.

$$\Phi_{se} : X \times \{a_1, \ldots, a_m\} \rightarrow \{false, true\} \tag{7.2}$$

Die Güte der Sequenzklassifizierung hängt zum einen von dem zur Klassifizierung verwendeten Verfahren ab, aber auch von der Güte des Korpus und von der Art der Klassifizierungskriterien. Vor allem Letztere spielen eine große Rolle. Je ausgeprägter diese Kriterien sind, desto besser ist eine Klassifizierung möglich. Eine Unterscheidung verschiedener Texte anhand der Sprache ist zum Beispiel leichter zu treffen als die nach dem Autor eines Textes, da sich verschiedene Schreibstile nur sehr gering unterscheiden. Die Fehlerquote ist somit von der Art des Klassifizierungskriteriums abhängig.

7.2 Sequenzklassifizierung

Ziel der Sequenzklassifizierung ist die Klassifizierung der Texte eines Korpus relativ zueinander. Ausgangspunkt ist ein Korpus, der eine Sammlung von Texten beinhaltet, die nach bestimmten Kriterien klassifiziert werden sollen. Diese Methode kann zum Beispiel zur Identifikation verschiedener Sprachgruppen dienen [DEV02]. Sei $X = \{x_1, \ldots, x_n\}$ eine Menge zu klassifizierender Texte und $A = \{a_1, \ldots, a_n\}$ der gegebene Korpus. Bei der Sequenzerkennung gilt $X = A$, das heißt es wird die Beziehung der Texte der Korpora untereinander berechnet. Die Sequenzklassifizierung beschreibt formell ein Verfahren zur Approximation der in Funktion 7.3 dargestellten Zielfunktion.

$$\Phi_{sk} : \{x_1, \ldots, x_n\} \times \{x_1, \ldots, x_m\} \rightarrow \{false, true\} \tag{7.3}$$

Anhand der errechneten Werte kann eine Entfernungsmatrix konstruiert werden, die die Entfernung der einzelnen Texte zueinander angibt. Grundlage der Sequenzklassifizierung ist somit die Konstruktion einer Entfernungsmatrix, deren Elemente die Entfernung zwischen Paaren von Texten enthalten. Diese Entfernung wird mittels der relativen Entropie der Texte untereinander berechnet. Ausgehend von der Entfernungsmatrix kann nun ein Baum zur Repräsentation der Ergebnisse erstellt werden.

7.3 Spezialfall: E-Mail Erkennung

Im Rahmen der Spamfilterung stellt sich die Frage, welchem Problem der Texterkennung die Klassifizierung von E-Mails zugeordnet werden kann. Ist eine Zuordnung zu einem Problem der Texterkennungsverfahren erfolgt, so kann eine konkrete Zielfunktion aufgestellt werden. Weiterhin ist es notwendig, die Berechnung der Zielfunktion mittels des festgestellten Verfahrens von anderen, bereits zur Spamfilterung verwendeten Verfahren abzugrenzen.

7.3.1 Berechnung der Filterentscheidungen

Bei der Sequenzerkennung wird ein neuer (unbekannter) Text klassifiziert, indem er mit bekannten Texten verglichen und anhand der Entfernung dieses Textes relativ zu den anderen klassifiziert wird, während bei der Sequenzklassifizierung die Einordnung von Texten gemäß deren relativen Bezug zueinander erfolgt. Ziel der Sequenzklassifizierung ist die Bestimmung der Nähe verschiedener Texte zueinander. Ausgangspunkt ist ein Text, der zum Beispiel in verschiedenen Sprachen vorliegt. Allein durch Verdeutlichung des

Zwecks der einzelnen Klassifizierungsmethoden wird deutlich, dass die Klassifizierung von E-Mails Ähnlichkeiten zur Sequenzerkennung aufweist und somit diesem Ansatz folgen muss. Im Rahmen der Spamfilterung wird somit die in Funktion 7.2 dargestellte Zielfunktion für die Sequenzerkennung als Grundlage für die Filterentscheidung verwendet. Die Konkretisierung der Approximierung dieser Funktion erfolgt in den folgenden Abschnitten und ist Bestandteil der Implementierung.

7.3.2 Abgrenzung der Spamfilterung mittels Texterkennungsverfahren zu bereits bestehenden Verfahren

Die Spamfilterung mittels Texterkennungsverfahren erfordert eine ständige Anpassung an aktuelle Gegebenheiten. Somit kann dieses Verfahren als dynamisches Verfahren zur Spamfilterung eingestuft werden (siehe Abschnitt 3.2.2). Hier ist nun eine Abgrenzung zu bereits bestehenden Verfahren notwendig. Dies betrifft zum einen die Differenzierung anhand der Unterschiede als auch die Darlegung der Vor- und Nachteile der einzelnen Verfahren. Im Rahmen der dynamischen Verfahren können neben dem Verfahren zur Texterkennung zwei Methoden zur Spamfilterung unterschieden werden: Bayes'sche Filter und Markov-Diskriminierung [Zdz05].

Allgemeine Unterschiede der Verfahren

Sowohl bei Verwendung von Bayes'sche Filtern als auch bei der Spamfilterung mittels Markov-Diskriminierung erfolgt eine Analyse des Body der E-Mail. Durch das Einfügen zusätzlicher Zeichen, Wörter oder Sätze kann dessen Inhalt verfälscht werden. Weiterhin ist das Filtern der E-Mail grundsätzlich nicht möglich, wenn der Inhalt der E-Mail als Bild eingefügt wurde. Eine Analyse des Header ist mittels dieser Verfahren, im Gegensatz zu dem Verfahren der Texterkennung, nicht vorgesehen.

Bei der Spamfilterung mittels Texterkennungsverfahren ist der gesamte Inhalt der E-Mail für die Spamfilterung relevant. Neben dem Body wird auch der Header der E-Mail betrachtet. Der Header enthält zwar nur eine geringe Menge relevanter Informationen, diese können jedoch sehr aussagekräftig sein und somit den Typ der E-Mail charakterisieren. Zu diesen Informationen gehören zum Beispiel Informationen über den absendenden Host oder über Zwischenknoten, die beim Versand der E-Mail passiert wurden. Weiterhin stehen die Informationen des Headers direkt zur Verfügung und können, im Gegensatz zum Body, nur schwer verfälscht werden. Die Analyse des Body kann zwar, wie auch die anderen dynamischen Verfahren, durch das Einfügen irrelevanter Wörter oder durch das Einfügen von Bildern beeinflusst werden,

da in diesem Zusammenhang jedoch außerdem eine Analyse des Headers erfolgt, ist dieses Verfahren weniger fehleranfällig. Desweiteren ist es möglich, eine größere Menge relevanter Informationen zu betrachten. Im Gegensatz zu den beiden anderen Verfahren erfolgt keine Beschränkung auf einzelne Wörter oder Sätze. Lediglich das Vorschaufenster oder das Wörterbuch des verwendeten Kompressionsverfahren beschränkt das Lernverhalten des Filters.

Abgrenzung zu den Bayes'sche Filtern

Bayes'sche Filter berechnen für jedes Wort des Korpus eine Wahrscheinlichkeit für das Auftreten dieses Wortes in Spam-Mails. Für jede neu eintreffende E-Mail werden diese Wahrscheinlichkeiten nun auf die Wörter dieser E-Mail angewendet und eine feste Anzahl dieser Wörter (15 oder 27) zur Klassifizierung der E-Mail herangezogen. Ausgewählt werden charakteristische Wörter, das heißt Wörter, deren Spam-Wahrscheinlichkeiten sehr hoch oder sehr niedrig sind und die bedingte Wahrscheinlichkeit über deren Auftrittswahrscheinlichkeit berechnet. Derartige Filter können umgangen werden, indem Spam-Mails mit einer großen Menge irrelevanter Wörter aufgefüllt werden. So werden die Auftrittswahrscheinlichkeiten der einzelnen Wörter verfälscht und eine genaue Analyse ist nicht mehr möglich. Im Vergleich zu den hier beschriebenen Texterkennungsverfahren arbeiten Bayes'sche Filter desweiteren nur auf einem sehr kleinen Ausschnitt der E-Mails. Der Kontext der E-Mail wird außer Acht gelassen.

Abgrenzung zur Markov-Diskriminierung

Bei der Markov-Diskriminierung wird geprüft, ob Ketten von Wörtern einer neu eintreffenden E-Mail bereits im Korpus zu finden sind. Hier steht die Reihenfolge der einzelnen Wörter im Vordergrund, so dass eine kontextabhängige Analyse der eintreffenden E-Mails erfolgt. Im Rahmen der Spamfilterung wird geprüft, ob Zeichenketten der eintreffenden E-Mail im Spam- und Ham-Korpus auftreten. Zur Berechnung der Spam-Wahrscheinlichkeit wird geprüft, in welchem Korpus die längere dieser Zeichenketten zu finden ist. Ausschlaggebend ist die Gesamtlänge der auftretenden Zeichenketten, die durch einzelne Wörter oder Zeichen unterbrochen werden können. Ebenso wie bei der Spamfilterung mittels Texterkennungsverfahren erfolgt eine kontextabhängige Analyse der neu eintreffenden E-Mail. Auch diese Filter können umgangen werden, indem eine Menge irrelevanter Ketten von Wörtern in eine Spam-Mail eingefügt werden. Somit werden neben den relevanten Bereichen der E-Mail auch irrelevante Bereiche analysiert und mit dem Korpus

verglichen.

Ein Unterschied zur Spamfilterung mittels Texterkennungsverfahren besteht darin, dass keine vollständige inhaltliche Analyse der E-Mails erfolgt, da die Länge der betrachteten Ketten aus Performanzgründen beschränkt werden muss. Desweiteren steht bei der Markov-Diskriminierung die Reihenfolge der Wörter, das heißt der Kontext im Vordergrund. Betrachtet wird der Informationsgehalt des folgenden Wortes, indem dessen Auftrittswahrscheinlichkeit in Abhängigkeit von den bisherigen Wörtern berechnet wird. Bei der Spamfilterung mittels Texterkennung wird ebenfalls der Informationsgehalt berechnet. Dieser ergibt sich jedoch vorrangig durch Reduktion der Redundanz unter Zuhilfenahme der Auftrittswahrscheinlichkeit von Zeichenketten und nicht aus deren Anordnung innerhalb der E-Mail.

Kapitel 8

Spamfilterung mittels Sequenzklassifizierung

In Kapitel 7 wurde verdeutlicht, dass ein Verfahren zur Spamfilterung dem Vorgehen der Sequenzerkennung folgen muss. Ziel ist somit die Approximation der in Formel 7.2 dargestellten Zielfunktion. In diesem Abschnitt werden verschiedene Verfahren und Methoden zur Annäherung an diese Zielfunktion vorgestellt und verglichen, um schließlich eine abschließende Formel zur Berechnung des Informationsabstands zu erhalten.

Das hier vorgestellte Verfahren zur Texterkennung basiert auf der Informationstheorie, die die in einem Text enthaltene Information als aufgehobene Unsicherheit auffasst und den durschnittlichen Informationsgehalt über den Begriff der Entropie definiert. Die Abschätzung der relativen Entropie zwischen zwei Texten erfolgt unter Zuhilfenahme von Datenkompressionsverfahren. Die relative Entropie gibt die Freiheit bei der Wahl einzelner Symbole an. Je geringer die relative Entropie ist, desto weniger Wahlmöglichkeiten für die einzelnen Symbole gibt es. Zur Klassifizierung einer eintreffenden E-Mail wird deren relative Entropie in Bezug auf die Ordner Spam und Ham berechnet. Eine geringe relative Entropie führt zu einer höheren Redundanz und erhöht somit die Wahrscheinlichkeit, dass eine E-Mail zu dem betroffenen Ordner gehört.

8.1 Berechnung des Informationsabstands

Ziel der Sequenzerkennung ist es, einen unbekannten Text X in Bezug auf eine Menge bekannter Texte A_i zu klassifizieren. Dazu wird für jeden Text der Menge $A = \{spam, ham\}$ der Informationsabstand zu dem zu klassifizierenden Text X berechnet. Mit der in 7.2 dargestellten Funktion ergibt sich

die in 8.1 dargestellte Zielfunktion für die Spamfilterung.

$$\hat{\Phi} : X \times \{spam, ham\} \rightarrow \{false, true\} \tag{8.1}$$

Der Informationsabstand gibt die Menge an Zeichen an, die zusätzlich benötigt werden, wenn Nachrichten einer Quelle X mit der für A optimalen Codierung dargestellt werden. Zur Berechnung des Abstands gibt es verschiedene Möglichkeiten: Die Berechnung des Abstands mittels der Kolmogorov-Komplexität oder die Berechnung mittels der relativen Entropie bzw. der Kullback-Leibler Divergenz [Kal04, BGL+98, Sha01]. Im Folgenden werden zunächst beide Verfahren beschrieben. Anschließend wird dargelegt, wie mittels dieser Verfahren eine Approximierung der Zielfunktion erfolgen und diese konkret umgesetzt werden kann.

8.1.1 Kolmogorov-Komplexität

Eine Möglichkeit der Bestimmung des Informationsabstands zwischen zwei Texten A und X besteht in der Berechnung der bedingten Kolmogorov-Komplexität [Kal04, BGL+98]. Die Kolmogorov-Komplexität oder (algorithmische) Entropie $K(X)$ entspricht der Länge des kürzesten binären Programms, um X zu berechnen. Die bedingte Kolmogorov-Komplexität $K(X|A)$ wird als die Länge des kürzesten Programms definiert, das benötigt wird, um X mit dem für A optimalen Programm zu berechnen. Bei einem derartigen Programm kann es sich zum Beispiel um eine Codierungsvorschrift für die einzelnen Zeichen der Quelle oder um eine Turing-Maschine handeln.

Um X mit dem für A optimalen Programm zu berechnen, also $K(X|A)$, werden maximal so viele Informationen benötigt, wie zur Berechnung von $K(A|X)$. Die (maximale) Entfernung der Quellen A und X berechnet sich mittels der in Funktion 8.2 dargestellten Entfernungsfunktion [Kal04]. Es muss nicht notwendigerweise $K(X|A) = K(A|X)$ gelten, da die Auftrittswahrscheinlichkeiten der einzelnen Nachrichten unterschiedlich sind.

$$E_1(X, A) \triangleq max\{K(X|A), K(A|X)\} \tag{8.2}$$

Oft ist es wünschenswert, einen festen Wertebereich zu erhalten um die spätere Analyse der Ergebnisse zu erleichtern und um eine Vergleichbarkeit der erhaltenen Ergebnisse zu ermöglichen. Der Wertebereich wird idealerweise im Intervall [0;1] festgelegt, da dieser Bereich durch eine Normalisierung erreicht werden kann. Dies führt zu einer zweiten (normalisierten) Entfernungsfunktion $E_2(X, A)$, die in Funktion 8.3 dargestellt ist [Kal04].

$$E_2(X, A) \triangleq \frac{max\{K(X|A), K(A|X)\}}{max\{K(X), K(A)\}} \tag{8.3}$$

Die Kolmogorov-Komplexität $K(X)$ eines Textes X entspricht (idealisiert) dessen Entropie $H(X)$ [Kal04, BGL+98]. Durch Einsetzen der Entropie H in Funktion 8.3 erhalten wir somit eine auf der Entropie basierende Formel für den Informationsabstand der Texte X und A, die wie in Funktion 8.4 dargestellt werden kann [Kal04].

$$E_2(X, A) = \frac{max\{H(X|A), H(A|X)\}}{max\{H(X), H(A)\}} \tag{8.4}$$

Die informationstheoretische Identität $H(X|A)$ berechnet sich wie in Funktion 8.5 dargestellt [Kal04, Sha01].

$$H(X|A) = H(X, A) - H(A) \tag{8.5}$$

Man erhält somit eine auf der Entropie basierende Formel für den Informationsabstand zweier Quellen.

8.1.2 Relative Entropie

Die relative Entropie oder Kullback-Leibler Divergenz beschreibt das Maß der Unterschiedlichkeit zweier Wahrscheinlichkeitsverteilungen bzw. Quellen X und A [Kal04] des gleichen Ereignishorizonts. Die relative Entropie beschreibt die Anzahl der überschüssigen Zeichen die entstehen, wenn eine Nachricht der Quelle X mit der für A optimalen Wahrscheinlichkeitsverteilung codiert wird. Wie bereits in Formel 5.3 dargestellt, berechnet sich die relative Entropie zweier Quellen A und X mit gegebenen Wahrscheinlichkeitsverteilungen p_A und q_B nach Funktion 8.6 [Kal04, PBC+03].

$$D(A||X) = \sum p_A \, log \frac{p_A}{q_X} \tag{8.6}$$

Die in 8.6 dargestellte Formel kann wie in 8.7 dargestellt mittels der Gesetze zur Berechnung von Logarithmen aufgelöst werden.

$$D(A||X) = -\sum p_A \, log \, q_X \; + \; \sum p_A \, log \, p_A \tag{8.7}$$

Die Entropie einer Quelle A berechnet sich wie in Formel 5.2 dargestellt, wenn die Auftrittswahrscheinlichkeiten der einzelnen Symbole voneinander unabhängig sind [Sha01]. Sei $N_A \in \{0,1\}$ die Menge möglicher Nachrichten der Quelle A, so berechnet sich die Entropie E_A nach Funktion 8.8, wenn p_i der Auftrittswahrscheinlichkeit des Symbols i entspricht.

$$\begin{aligned} E_A &= -p\log_2 p - (1-p)\log_2(1-p) \\ &= \sum_{i\in 0,1} p_i \log_2 p_i \end{aligned} \tag{8.8}$$

Sollen nun die Nachrichten $N_X \in \{0,1\}$ einer Quelle X mit der für A optimalen Codierung dargestellt werden, so werden für die Codierung der Nachrichten die für A optimalen Wahrscheinlichkeiten verwendet. Bei zwei möglichen Nachrichten berechnet sich die Entropie E_{Ax} der Quelle X wie in Funktion 8.9 dargetellt, wenn q_i der Aufrittswahrscheinlichkeit des Symbols i entspricht.

$$\begin{aligned} E_X &= -p\log_2 q - (1-p)\log_2(1-q) \\ &= \sum_{i\in 0,1} p_i \log_2 q_i \end{aligned} \tag{8.9}$$

Eine Verallgemeinerung der Formeln 8.8 und 8.9 ergibt die in 8.10 und 8.11 dargestellten Formeln. Die Werte aus Funktion 8.7 können somit mittels der Entropie dargestellt werden.

$$-\sum p_A \; log \; p_A = H(A) \tag{8.10}$$

$$-\sum p_A \; log \; q_X = H(X,A) \tag{8.11}$$

Ein Einsetzen der Werte aus 8.10 und 8.11 in 8.7 ergibt eine Formel zur Berechung der relativen Entropie, dargestellt in Formel 8.12.

$$D(A||X) = H(X,A) - H(A) \tag{8.12}$$

$$= H(X|A) \tag{8.13}$$

Auch in diesem Fall erhält man somit eine auf der Entropie basierende Formel für den Informationsabstand zweier Quellen.

8.2 Berechnung der (relativen) Entropie

Sowohl bei der Berechnung des Informationsabstands mittels der Kolmogorov-Entropie, als auch bei der Berechnung des Abstands mittels der relativen Entropie wird das Problem auf die Berechnung der Entropie einer Informationsquelle $H(A)$ und auf die Berechnung der bedingten Entropie $H(X, A)$ reduziert. Im Folgenden bleibt somit zu klären, auf welche Art und Weise die (bedingte) Entropie effizient berechnet werden kann, um somit die in Funktion 8.1 dargestellte Zielfunktion zu konkretisieren.

Zur Berechnung der relativen Entropie $H(X|A)$ muss die bedingte Entropie $H(X, A)$, sowie die Entropie der Quelle A, $H(A)$ bestimmt werden. Eine Möglichkeit besteht in der Bestimmung der Werte mittels Kompression [Kal04, BGL+98]. Ziel der Datenkompression ist die Minimierung der Datengröße. Dies wird durch eine Reduzierung der Redundanz erreicht, so dass nur der informelle Inhalt verbleibt, der durch die Entropie beschrieben wird. Da der Informationsgehalt bei der (verlustfreien) Kompression der Daten erhalten bleiben muss, liefert die Entropie H die untere Grenze für die Datenkompression.

8.2.1 Abschätzung der Entropie

Mittels der Kompressionsrate kann die Güte eines Kompressionsverfahrens bestimmt werden [BK90]. Sie gibt den Faktor an, um den bei der Kompression eine Reduktion der Dateigröße erfolgt und ermöglicht somit die Vergleichbarkeit der einzelnen Verfahren. Informell kann die Kompressionsrate $\mathcal{K}$ wie in Funktion 8.14 berechnet werden [Ver98, PBC+03].

$$\mathcal{K} = \frac{\textit{Länge der Nachricht nach der Kompression}}{\textit{Länge der ursprünglichen Nachricht}} \tag{8.14}$$

Sei $|N|$ die Länge der Nachricht N vor der Kompression und $|\mathcal{C}(N)|$ die Länge der Nachricht nach der Kompression, so kann die in 8.14 dargestellte Formel wie in 8.15 formalisiert werden.

$$\mathcal{K} = \frac{|\mathcal{C}(N)|}{|N|} \tag{8.15}$$

Diese (informelle) Darstellung findet in Definition 17 eine Konkretisierung.

Definition 17 (Kompressionsrate) *Sei $L(y_i^n)$ die Länge der komprimierten Nachricht (in Bit) und $n\ log_2\alpha$ die Länge der Eingabe, so gilt für die Kompressionsrate $\mathcal{K}$ [LZ78]:*

$$\mathcal{K} = \frac{L(y_i^n)}{n\ log_2\alpha} \tag{8.16}$$

Es muss nicht notwendigerweise $\mathcal{K} \leq 1$ gelten. Bei jeder Kompression wird eine feste Anzahl an Bits zur Codierung von Zustandsinformationen benötigt, die mit dem komprimierten Text mitgeführt werden. Vor allem bei kurzen Texten ist es möglich, dass die Menge der Zustandsinformationen größer ist als die erreichte Reduktion der Redundanz, so dass die Dateigröße durch die Kompression steigt. Idealerweise sinkt die Kompressionsrate mit steigender Dateigröße, so dass die Entropie einer Sequenz A durch Berechnung der in Definition 17 dargestellten Kompressionsrate bestimmt werden kann. Die Berechnung der Entropie kann mittels der in Definition 18 dargestellten Funktion erfolgen [LZ78].

Definition 18 (Bestimmung der Entropie) *Für jede (unendliche) Sequenz A geht die Kompressionsrate gegen deren Entropie. Sei $|A|$ die Länge des zu komprimierenden Textes, so gilt:*

$$\lim_{|A|\to\infty} \mathcal{K}(A) = H(A) \tag{8.17}$$

Mit steigender Länge des zu komprimierenden Textes nähert sich die Kompressionsrate somit der Entropie. Dabei handelt es sich um einen idealisierten Wert der nur dann erreicht werden kann, wenn keine Beschränkung des verwendeten Wörterbuchs und des Vorschaufensters des Kompressionsverfahrens erfolgt.

8.2.2 Berechnung der Entropie H(A)

Im vorhergehenden Abschnitt wurde bereits die Berechnung der Entropie mittels Datenkompressionsverfahren beschrieben. Mit Definition 17 und Funktion 8.15 ergibt sich somit die in 8.18 dargestellte Formel zur Berechnung der Entropie $H(A)$.

$$\begin{aligned} H(A) &\sim \mathcal{K}(A) \\ &= \frac{|\mathcal{C}(A)|}{|A|} \end{aligned} \tag{8.18}$$

Die Entropie $H(A)$ kann somit mittels Datenkompression angenähert werden. Voraussetzung ist gemäß Definition 18, dass die Länge des zu komprimierenden Textes gegen unendlich geht und dass (grundsätzlich) keine Beschränkung des Vorschaufensters oder des Wörterbuchs erfolgt.

8.2.3 Berechnung der bedingten Entropie H(X,A)

Um die bedingte Entropie $H(X, A)$ zu berechnen, muss zunächst eine für A optimale Codierung bestimmt werden. Dies kann durch Kompression der Nachricht A erreicht werden. Anschließend wird X mit der für A optimalen Codierung dargestellt. Dies wird, wie in Funktion 8.19 dargestellt, durch Konkatenation von A und X und anschließende Kompression erreicht [Kal04, BGL+98, Sha01]. Die Normalisierung mittels der originalen Länge der Datei ist notwendig, um den tatsächlichen Abstand und somit die bedingte Entropie zu erhalten.

$$\begin{aligned} H(X, A) &\sim \mathcal{K}(X, A) \\ &= \frac{|\mathcal{C}(A \circ X)|}{|A \circ X|} \end{aligned} \tag{8.19}$$

Bei der Datenkompression wird die Redundanz von Dateien ausgenutzt, um eine Reduktion der Dateigröße vorzunehmen. Dies wird vor allem dadurch erreicht, dass häufig auftretende Zeichenfolgen mit möglichst wenig Bits codiert werden. Die für A optimale Codierung ist nur dann für eine gute Kompression des unbekannten Textes X geeignet, wenn A und X gemeinsame Eigenschaften haben. Je mehr gemeinsame Eigenschaften vorliegen, desto größer ist die Reduktion der Dateigröße und desto mehr ähnelt der unbekannte Text X dem bereits bekannten.

8.3 Ergebnis

Ziel der Spamfilterung ist die Einordnung einer neu eintreffenden E-Mail anhand ihres Abstands zu bestehenden Ordnern. Mittels der Kolmogorov-Komplexität wird allgemein der Abstand zweier Texte voneinander berechnet, während die relative Entropie den Abstand eines bestimmten Textes X von einer bekannten Quelle A berechnet.

Bei der Datenkompression erfolgt eine Reduktion der Dateigröße durch Ausnutzen der in der Datei enthaltenen Redundanz. Wird ein neuer Text X

an eine große Datei A angehängt und komprimiert, so kann sich die Kompressionsrate zunächst verschlechtern, wenn der neue Text X viele neue Merkmale enthält, die in Text A nicht enthalten sind. Sehr deutlich sind diese Unterschiede, wenn A und X in verschiedenen Sprachen verfasst sind. Heutige Kompressionsverfahren arbeiten nicht statisch, sondern passen sich dynamisch an die Gegebenheiten an. Mit steigender Dateigröße wird sich die Kompressionsrate somit wieder verbessern.

Bei der Spamfilterung wird ein sehr kurzer Text, die eintreffende E-Mail X, mit dem (meist sehr großen) Korpus A konkateniert und anschließend komprimiert. Der Text X ist in der Regel so kurz, dass nur dann eine starke Verbesserung der Kompressionsrate erreicht werden kann, wenn der Text X dem jeweiligen Korpus sehr ähnlich ist. Eine Berechnung des Abstands mittels der Kolmogorov-Komplexität ist im Rahmen der Spamfilterung nicht sinnvoll, da der zu klassifizierende Text X im Vergleich zu dem bekannten Korpus A sehr klein ist, so dass die Berechnung H(A|X) keine zuverlässigen Ergebnisse liefern kann. Weiterhin ergeben sich kaum Unterschiede der Dateigrößen, da der Text X bereits nach kurzer Zeit nicht mehr im Vorschaufenster des Kompressionsverfahrens enthalten ist. Die Berechnung des Informationsabstands mittels der Kolmogorov-Komplexität kann somit nur dann eingesetzt werden, wenn A und X ähnliche Dateigrößen aufweisen. Für die Spamfilterung wird der Informationsabstand wie in Sektion 8.1.2 dargestellt mittels der Kullback-Leibler-Divergenz berechnet.

Die Entropie und somit der Informationsabstand zwischen Texten, kann mittels Verfahren zur Datenkompression abgeschätzt werden. Zur Berechnung der relativen Entropie $H(X|A)$ ist es notwendig, Vergleiche der Größen der komprimierten Dateien vorzunehmen. Mit den Ergebnissen aus den Funktionen 8.18 und 8.19 ergibt sich die in Funktion 8.20 dargestellte Formel zur Berechnung des Informationsabstands.

$$\begin{aligned} H(X|A) &= H(X,A) - H(A) \\ &\sim \mathcal{K}(X,A) - \mathcal{K}(A) \\ &= \frac{|\mathcal{C}(A \circ X)|}{|A \circ X|} - \frac{|\mathcal{C}(A)|}{|A|} \end{aligned} \tag{8.20}$$

Zur Klassifizierung von E-Mails wird zunächst ein Korpus (A) angelegt, der aus zwei Dateien besteht: Einem Ordner für Ham-Mails (ham) und einem Ordner für Spam-Mails ($spam$). Für jede eintreffende E-Mail X und jeden Benutzerordner $A \in \{spam, ham\}$, wird nun mittels Funktion 8.20 die relative Entropie berechnet. Die E-Mail wird dem Ordner zugeführt, für den $H(X|A_i)$ minimal ist, da in diesem Fall der Informationsabstand mini-

mal wird. Mit der in 8.20 dargestellten Formel und in 8.1 Funktion ergibt sich nun wie in Funktion 8.21 dargestellt eine konkrete Zielfunktion für die Spamfilterung.

$$\hat{\Phi}_{Filter} : H(X|A) \rightarrow \{false, true\} \tag{8.21}$$

Die Entscheidung des Filters wird anhand bereits getroffener Entscheidungen gefällt. Die Trainingsmenge Ω darf nicht leer sein, da ansonsten keine Vergleiche der sich verändernden Dateigrößen durchgeführt werden können. Es ist zu erwarten, dass sich die Entscheidungsgenauigkeit mit steigender Korpusgröße verbessert, da in diesem Fall eine größere Menge an Eigenschaften zur Entscheidung herangezogen werden können. Der Lernerfolg des Filters ist jedoch durch das Vorschaufenster oder das Wörterbuch des verwendeten Kompressionsverfahren beschränkt, da diese aus Performanzgründen nicht beliebig groß gewählt werden können.

Kapitel 9

Implementierung des Filters

Die Implementierung des Spamfilters erfolgte mittels der Skriptsprache `Perl` unter Verwendung des Zusatzmoduls `Mail::MboxParser`, das über das Comprehensive Perl Archive Network (*CPAN*) zur Verfügung steht [cpa06]. Dieses Modul liefert eine Schnittstelle für den Zugriff auf E-Mails, die im *mbox*-Format gespeichert sind. Dabei erfolgt eine Beschränkung auf das Lesen der E-Mails. Analysiert wurden E-Mails, die dem Mail-Fundus von Trec entnommen wurden [Tre06]. Die E-Mails lagen als einzelne Dateien vor und wurden unter Zuhilfenahme eines Shell-Skriptes gemäß ihrem zeitlichen Eintreffen in eine einzige *mbox*-Datei übertragen. In einer zusätzlichen Textdatei wurde vermerkt, welche E-Mail als Ham- und welche als Spam-Mail einzustufen ist. Im Rahmen dieses Abschnitts erfolgt die Darstellung des Filters unter Zuhilfenahme einzelner Abschnitte des Quellcodes.

Für die dem Filter zugrundeliegenden Mechanismen, das Einordnen von E-Mails und die Kompression der Ordner, wurde auf externe Mechanismen zurückgegriffen. Das `Perl`-Programm dient lediglich dem Zugriff auf die einzelnen E-Mails und stellt einen Mechanismus zur Koordinierung dieser externen Mechanismen dar. Die Speicherung und Einordnung der E-Mails in Benutzerordner wurde mit Hilfe des externen Programms `procmail` durchgeführt. Weiterhin wurde zwischen Header und Body der E-Mail unterschieden, die getrennt voneinander analysiert und bewertet wurden. Die Trennung in Header und Body erfolgte ebenfalls unter Zuhilfenahme von `procmail`. Zur Abschätzung der relativen Entropie wurden die Kompressionsverfahren gzip und rar eingesetzt.

Der Spamfilter besteht grundsätzlich aus drei Teilen. Der erste Teil ist für das Erstellen von `procmail`-Rezepten verantwortlich, um eine Einordnung der E-Mails in Benutzerordner vorzunehmen. Der zweite Teil betrifft die Bestimmung der Dateigröße nach der Kompression des Korpus mit den einzelnen Kompressionsverfahren und der dritte Teil das Empfangen der E-

Mails und Berechnen der notwendigen Werte für die Filterentscheidungen. Implementiert wurde ein Filter mit Feedback unter Verwendung der *TEFT*-Trainingsmethode (siehe Kapitel 4.1 und 4.2), das heißt alle Entscheidungen des Filters wurden überwacht und im Fehlerfall korrigiert. Im Rahmen der Spamfilterung wurde in jedem Test mit jeweils einer E-Mail im Korpus begonnen. Alle weiteren E-Mails wurden als einkommende E-Mails betrachtet, in der Reihenfolge ihres zeitlichen Eintreffens auf dem zugrundeliegenden Mailsystem dem Filter zugeführt und analysiert. Ziel ist es, dass der Filter im Laufe der Zeit "lernt", welche E-Mails für den Benutzer relevant sind und welche nicht.

9.1 Einordnung der E-Mails mittels `procmail`

Im Rahmen der Filterung der E-Mails ist es notwendig, einzelne E-Mails bzw. Teile davon den Ordnern Spam und Ham zuzuführen. Um dies zu erreichen, wurde auf das externe Programm `procmail` zurückgegriffen, das in den meisten Linux-Distributionen enthalten ist [BM02]. Das Speichern von E-Mails erfolgt nach festen Regeln, den so genannten "Rezepten", die vom Benutzer angelegt und von `procmail` nach Vorschriften für den Umgang mit E-Mails durchsucht werden. Um die einzuordnenden E-Mails zu speichern, werden mittels Perl `procmail`-Rezepte erzeugt, die den Body bzw. den Header einer E-Mail extrahieren und ihn dem Corpus zuführen (siehe Listing 9.1).

Listing 9.1: *Perl-Programm für das Erzeugen von `procmail`-Rezepten*

```
setup_procmail {
    # Get the values from the default input-string
    my $type = $_[0];          # Ham/Spam
    my $mailparts = $_[1];   # Header/Body (h/b)

    # Create a file for the procmail-recipe
    my $rcfile = $working_dir . "rc." . $mbox;

    # Open the file the recipe is written to
    open(RC, ">$rcfile");

    # The folder, procmail gets the mails from
    print RC "MAILDIR=$working_dir\n";

    # Location and name of the log-file
    print RC 'LOGFILE=$MAILDIR/log.'.$type."\n";

    # Print the body of the mail to file;
```

```
    print RC ":0 $mailparts\n";
    print RC "corpus_$type\n";

    # Close the file of the recipe
    close RC;

    return $rcfile;
}
```

Ein derartiges Rezept muss für jeden Korpus angelegt werden, in dem E-Mails gespeichert werden sollen (`$type` = Ham/Spam), als auch für jeden zu analysierenden Teil der E-Mail (Header/Body (`Oh/Ob`)). Das Rezept gibt zum einen an, welcher Teil der E-Mail extrahiert werden soll, als auch den Ordner, dem diese E-Mail zuzuführen ist (`corpus_$type`) und wo sich dieser Ordner befindet (`MAILDIR`).

9.2 Berechnung der Korpusgröße

Zur Berechnung des Informationsabstands werden externe Kompressionsverfahren eingesetzt, die aus dem Perl-Programm heraus aufgerufen werden. Im Einzelnen sind dies gzip und rar. Zunächst wird jedem der verwendeten Kompressionsverfahren der Korpus übergeben (`$corpus`). Außerdem werden weitere Parameter für die Kompression benötigt. Im Einzelnen sind dies ein Parameter für die Komressionsrate (`$compression_ratio`) und das Ziel-Archiv (`$compression_target`) für das Verfahren gzip. Der Aufruf des Kompressionsprogramms ist in Listing 9.2 dargestellt.

Listing 9.2: *Ausschnitt des Perl-Programms für die Durchführung der Kompression mittels gzip*

```
    # call the gzip compression-program
    system( "/usr/bin/gzip",
            "$compression_ratio",
            "$compression_target",
            "$corpus") == 0
        or die "Error during compression!\n";
```

Im Anschluss an die Kompression wird die Dateigröße ausgelesen, die als Rückgabewert in der Variablen `$delta` gespeichert wird. Dies erfolgt unter Zuhilfenahme des `Perl`-Moduls `File::stat`, wie in Listing 9.3 dargestellt. Anschließend wird die komprimierte Datei wieder gelöscht um sicherzustellen, dass keine Fehler durch vorhandene Dateifragmente entstehen, die die Dateigrößen verfälschen können.

Listing 9.3: *Ausschnitt des Perl-Programms für das Auslesen der Dateigröße mittels des Moduls* `File::stat`

```
# Get the size of the compressed file
$delta = stat($compressed_file)->size
    or die "Can't stat file: $! \n";

# Delete the compressed file
unlink($compressed_file);

# Return the computed filesize
return $delta;
```

Die Dateigröße (`$delta`) wird an die aufrufende Funktion zurückgegeben und dient der Berechnung der Filterentscheidung.

9.3 Berechnung der Werte für die Filterentscheidungen

Die Berechnung der Werte, die für das Treffen von Filterentscheidungen benötigt werden, erfolgt innerhalb einer eigenen Methode. Zunächst ist es notwendig, die eintreffenden E-Mails zu initialisieren und ein `procmail`-Rezept anzulegen, um auf die einzelnen Ordner zugreifen zu können. Für jeden Korpus als auch für jeden zu analysierenden Bereich der E-Mail muss ein derartiges Rezept erstellt werden. Listing 9.4 zeigt exemplarisch die Initialisierung des Filters für die Analyse der Header der E-Mails. Diese Initialisierung der `procmail`-Rezepte muss analog für den Body der E-Mails erfolgen.

Listing 9.4: *Ausschnitt des Perl-Programms für die Initialisierung des Filters mittels Spam-Mails*

```
# Setup procmail recipe for the spam-mails
my $rc_spam_header =
    setup_procmail( "spam",
                    "header",
                    $working_dir);
my $rc_ham_header =
    setup_procmail( "ham",
                    "header",
                    $working_dir);

# Initialize the incoming E-Mails
my $incoming =
    Mail::MboxParser -> new(
```

```
                    "$working_dir" . "incoming",
                    decode      => 'ALL',
                    parseropts  => $parseropts
    );
```

Anschließend erfolgt die Initialisierung der Korpora des Filters. Um den Lernprozess nachvollziehen zu können, wird der Spam- und Ham-Korpus jeweils mit einer E-Mail initialisiert. Dazu wird die erste E-Mail gemäß dem Procmail-Rezept in den betroffenen Korpus geschrieben. In Listing 9.5 ist exemplarisch die Initialisierung des Filters mittels der Header der E-Mails dargestellt. Das Verfahren für den Body verläuft analog.

Listing 9.5: *Initialisierung des Korpus mittels dem Header der E-Mails*

```
# get the first spam- and the first ham-message
my $spam_message = $incoming_spam->get_message(0);
my $ham_message  = $incoming_ham->get_message(0);

# Write the spam-message to the corpus-file
open(OUT, "|procmail $rc_spam_header");
print OUT $spam_message;
close OUT;

# Write the ham-message to the corpus-file
open(OUT, "|procmail $rc_ham_header");
print OUT $ham_message;
close OUT;
```

Im Rahmen des eigentlichen Filterprozesses werden dem Filter Ham- und Spam-Mails abhängig von ihrem zeitlichen Eintreffen auf dem System übergeben. Für jede Mail werden nun die relevanten Werte berechnet. Im Einzelnen sind dies die Größe des originalen und des um die eintreffende E-Mail erweiterten Korpus vor und nach der Kompression. Diese Werte werden für die weitere Verarbeitung in eine Datei geschrieben und der Prozess wird fortgesetzt, bis keine weiteren E-Mails mehr eintreffen. In Listing 9.6 ist die exemplarische Berechnung der Werte für den Spam-Korpus bei einer eintreffenden Spam-Mail unter ausschließlicher Berücksichtigung des Header dargestellt. Das Verfahren für den Body der Spam-Mails und für eintreffende Ham-Mails verläuft analog.

Listing 9.6: *Exemplarische Berechnung der für die Filterentscheidung relevanten Werte in Bezug auf den Korpus des Body der Spam-Mails*

```
for (my $a = 0; $a<$number_of_mails; $a++) {

    ...

    # Get an incoming message to be classified
    my $message = $incoming_spam->get_message($a);

    # Compute the size of the origin corpus
    $origin_filesize =
        stat($working_dir."spam_header")->size;
        or die "Can't stat file: $!\n";

    # Compute the size of the compressed file
    $origin_compressed = delta(spam_header);

    # Add the incoming message to the corpus
    open(OUT, "|procmail $rc_spam_header");
    print OUT $message;
    close OUT;

    # Compute the size of the new corpus
    $new_filesize =
        stat($working_dir."spam_header")->size;
        or die "Can't stat file: $!\n";

    # Compute the size of the compressed file
    $new_compressed = delta(spam_header);

    # Write the results to a text-file
    open(LOG,">$working_dir"."$log");
    print $origin_filesize, $origin_compressed,
        $new_filesize, $new_compressed;
    close LOG;

    ...

}
```

Da der Filter mit Feedback und der Trainingsmethode *TEFT* implementiert wurde ist es nicht notwendig, die Filterentscheidungen zusätzlich zu berechnen. Statt dessen wurden die erhaltenen Werte in eine Datei geschrie-

ben, damit sie für weitere Berechnungen zur Verfügung stehen. Die eigentlichen Entscheidungen werden mittels eines weiteren Skriptes berechnet (siehe Sektion 9.4). So ist es außerdem möglich, Aussagen über die tatsächliche Berechnungszeit zu machen.

9.4 Berechnung der Filterentscheidung

Die Berechnung der Filterentscheidung erfolgt über ein separates Programm. Die Filterentscheidungen wurden zum einen anhand der relativen Entropie getroffen, die sich wie in Funktion 8.20 dargestellt berechnet. Neben der relativen Entropie wurde eine Filterentscheidung für einen Wert *Delta* berechnet, der die (nicht normierte) Differenz der komprimierten Dateigrößen ausgibt. Ziel ist zum einen die Berechnung eines Vergleichswertes als auch um festzustellen, welchen Einfluss die Normierung mit der Dateigröße der in 8.20 dargestellte Funktion auf die Filterentscheidung hat. Sei X die neu eintreffende E-Mail, A_i der bestehende Korpus, $|C(A_i)|$ die Dateigröße des komprimierten Korpus A_i und $|C(A,X)|$, so berechnet sich Δ_{AX} wie in Funktion 9.1 dargestellt.

$$\begin{aligned} \Delta_{AX} &= \Delta_{A+X} - \Delta_A \\ &= |C(A,X)| - |C(A)| \end{aligned} \tag{9.1}$$

Die Filterentscheidungen für die relative Entropie und *Delta* werden unabhängig voneinander berechnet. Allgemein gilt: Für jede eintreffende E-Mail X und jeden Korpus $A \in \{ham, spam\}$ wird die relative Entropie und der Wert für *Delta* berechnet. Die E-Mail wird dem Ordner zugeführt, für den der Wert minimal wird. Die getroffenen Entscheidungen werden für eine spätere Auswertung in einer Datei protokolliert. Hier kann zwischen korrekten Entscheidungen, falsch-positiv, falsch-negativ und keiner Entscheidung (im Falle der Gleichheit der Werte) unterschieden werden.

9.5 Integration des Filters in ein bestehende Strukturen

Im Rahmen dieser Arbeit werden lediglich die theoretischen Grundlagen zur Spamfilterung mittels Texterkennungsverfahren dargelegt. Diese Ergebnisse werden anhand verschiedener Testreihen verifiziert. Die Integration in ein bestehendes System zur Spamfilterung erfolgt nicht. Dennoch stellt sich die

Frage, wie eine Einbettung des Spamfilters in bestehende Strukturen erfolgen kann. Dies betrifft zum einen die Integration des Filterverfahrens in einen Spamfilter, als auch die Integration in die Einsatzumgebung des Filters selbst.

9.5.1 Integration der Filterroutine in einen bestehenden Spamfilter

Spamfilter verwenden eine Vielzahl unterschiedlicher Verfahren für die Analyse der E-Mails und das Treffen der Filterentscheidungen. Diese Verfahren werden einzeln auf jede eintreffende E-Mail angewendet und abschließend zusammenfassend hinsichtlich der Spamwahrscheinlichkeit dieser E-Mail bewertet, wie in Abbildung 9.1 dargestellt. Grundsätzlich bestehen Spamfilter aus einer Kombination statischer und dynamischer Verfahren. Bei statischen Verfahren finden meist Black- und Whitelists ihren Einsatz, während dynamische Verfahren in der Regel Bayes'sche Filter implementieren. Im Rahmen der Analyse trifft jedes Verfahren eine Aussage über die Spamwahrscheinlichkeit der E-Mail. Abhängig von der Art des Verfahrens ergibt die Gewichtung dieser Aussagen eine Spamwahrscheinlichkeit. Eine als Spam erkannte E-Mail kann in einem speziellen Ordner separiert, gelöscht oder abgelehnt werden, während Ham-Mails den Benutzerordnern zugeführt werden.

Im Rahmen der Spamfilterung wird eine Vielzahl verschiedener Verfahren eingesetzt, die eine neue eintreffende E-Mail unabhängig von vorhergehenden bewerten. Die Art und Menge der verwendeten Verfahren hängt vom verwendeten Spamfilter ab. Ziel ist es, das Umgehen des Spamfilters zu erschweren, indem Verfahren mit unterschiedlichen Charakteristiken eingesetzt werden. Ein Spammer muss diese Charakteristiken kennen und beachten, um den Spamfilter zu umgehen.

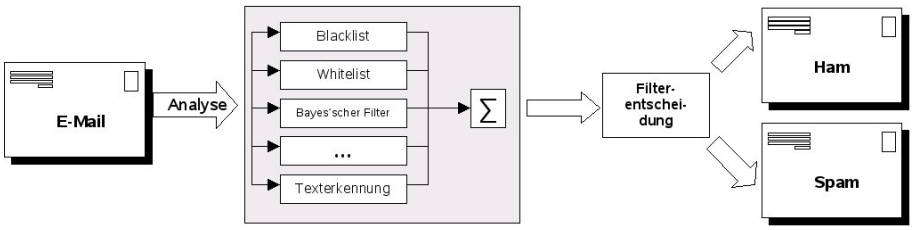

Abbildung 9.1: *Einbettung der Texterkennung in einen Spamfilter. Jede einkommende E-Mail wird analysiert und ein Ergebnis berechnet. Anhand dieses Ergebnisses wird eine Filterentscheidung getroffen*

Funktionsweise von SpamAssassin

SpamAssassin ist eine freie Software zur Filterung von E-Mails, die anhand der Kombination verschiedener Techniken angibt, mit welcher Wahrscheinlichkeit es sich bei der analysierten E-Mail um Spam handelt [Sch05, EW05]. SpamAssassin dient lediglich der Markierung von E-Mails und nicht dem Löschen, Ablehnen oder Aussortieren. Zur Markierung der E-Mail werden zusätzliche Headerzeilen eingefügt, die vom E-Mail Client zur Bewertung der E-Mail verwendet werden können. SpamAssassin beinhaltet verschiedene Verfahren zur Klassifizierung von E-Mails. Mit jedem Test wird ein Punktewert ermittelt. Die Summe aller Punkte entspricht der Spam-Punktzahl der E-Mail. Negative Punkte geben an, dass es sich eher nicht um Spam handelt, während positive Punkte die Spam-Wahrscheinlichkeit erhöhen. Es werden fast ausschließlich Tests auf Spam durchgeführt, so genannte positive Tests. Ab welchem Punktewert eine E-Mail als Spam markiert wird, ist abhängig von den Einstellungen des Mail-Clients.

Ein Großteil der Tests sind statische Tests, die unabhängig von den bisher gefilterten E-Mails sind und in der Regel vom Benutzer angepasst werden müssen (siehe Abschnitt 3.2). Hier handelt es sich meist um Blacklists, mit denen sowohl der Inhalt der E-Mail als auch der Header überprüft wird und Whitelisting, das E-Mails mit Absendern aus dem Adressbuch als Ham aussortiert. Neben den statischen Verfahren beinhaltet SpamAssassin einen Bayes'schen Filter als dynamisches Verfahren. Nach der Verteilung der Punkte und Summierung aus den einzelnen Tests wird geprüft, ob es sich bei der E-Mail um Spam handeln könnte. Dies ist der Fall, wenn das Spamlevel über 0 liegt, das heißt wenn mehr als 0 Punkte erreicht wurden. Nun fügt SpamAssassin drei zusätzliche Kopfzeilen zur E-Mail hinzu. Dies sind im Einzelnen der Spam-Status, das Spamlevel und der Spam-Report. Der Spam-Status beinhaltet das Klassifizierungsergebnis (LOW, MEDIUM und HIGH) und die Angabe des Spamlevel multipliziert mit 10. Das Spamlevel gibt mit "*" (ganzzahliger Teil) und "+" (Dezimalteil) den erreichten Punktewert an, zum Beispiel ***++ bei 3,2 und der Spam-Report gibt die Spam-Kriterien an, die in der E-Mail aufgetreten sind, das heißt die Tests, die zum Ergebnis "Spam" geführt haben.

Integration in SpamAssassin

Denkbar wäre eine Einbettung der textbasierten E-Mailerkennung in das System von SpamAssassin. Dazu ist es notwendig, einen Punktewert für eine mittels der Texterkennung behandelte E-Mail festzulegen. Grundsätzlich vergibt SpamAssassin Punkte auf verschiedene Art und Weise. Im Rahmen der

Analyse mittels einer Blacklist werden positive Punkte vergeben und für das endgültige Ergebnisse addiert. Die Anzahl der Punkte liegt zwischen $0,001$ und 5. Die Menge der für eine mittels Bayes'scher Filter bewerteter E-Mail ist von der errechneten Spamwahrscheinlichkeit abhängig und liegt zwischen 0,001 und 3,5 [SAP07].

Eine Einbindung in SpamAssassin kann die unabhängige Berechnung von Header und Body einer E-Mail ermöglichen. Im Gegensatz zu Bayes'schen Filtern berechnen Filterverfahren mittels Texterkennung keine Spamwahrscheinlichkeit, sondern geben ein konkretes Ergebnis aus. Da der Header grundsätzlich bessere Erkennungsraten liefert als der Body, sollte eine Analyse des Header eine größere Gewichtung erhalten. Aufgrund der hohen Fehlerrate des Body wird eine Punktzahl von $2,5$ für eine als Spam erkannte E-Mail empfohlen, $-2,5$ sofern es sich um eine Ham-Mail handelt, und eine Punktzahl von 3 (bzw. -3) bei der Analyse des Header. Die Vergabe sowohl positiver als auch negativer Punkte soll dazu führen, dass die Menge der falsch-positiven Ergebnisse reduziert wird, da nur E-Mails als Spam klassifiziert werden, die mittels beider Verfahren als solche erkannt wurden. Ansonsten ist der Punktewert sehr gering und die anderen verwendeten Verfahren erhalten ein größeres Gewicht.

Eine weitere Möglichkeit für die Punktevergabe besteht in einer Kombination der Ergebnisse der Texterkennungsverfahren. So ist es zum Beispiel möglich, eine Analyse mittels verschiedener Kompressionsverfahren durchzuführen. Im Rahmen des Trainings des Filters wird zunächst berechnet, wie hoch die Erkennungsraten der einzelnen Kompressionsverfahren für die Analyse des Header und die des Body sind. Jede eintreffende E-Mail wird nun anhand der Verfahren analysiert und bewertet. Eine bedingte Wahrscheinlichkeit über die erreichten Ergebnisse gibt die Spamwahrscheinlichkeit der eintreffenden E-Mail an. Bei der Punktevergabe kann nun eine Orientierung an der Punktevergabe des in SpamAssassin integrierten Bayes'schen Filters erfolgen, so dass nun Punkte zwischen $0,001$ und 5 vergeben werden.

9.5.2 Integration des Filters in ein Firmennetz

Nach der Einbindung des Filterverfahrens in einen Spamfilter ist dessen Integration in bereits bestehende Strukturen notwendig. Im Folgenden wird beispielhaft die Integration des Spamfilters in ein Firmennetzwerk dargestellt. Die Einbettung des Spamfilters kann sowohl server- als auch clientseitig erfolgen. Dies ist abhängig von der geplanten Einsatzumgebung. Vor allem in Firmen erfolgt in der Regel der Einsatz eines zentralen Mailservers, der die Postfächer der Benutzer mittels IMAP zugänglich macht. Der Mailserver befindet sich im internen Firmennetz und wird somit durch eine Firewall ge-

schützt. Der Zugriff auf die E-Mails kann sowohl über das lokale Netzwerk als auch über das Internet erfolgen. Bei einem Zugriff über das Internet wird die Verbindung mittels SSL/TLS abgesichert. Diese Einsatzumgebung ist in Abbildung 9.2 dargestellt [Lau06].

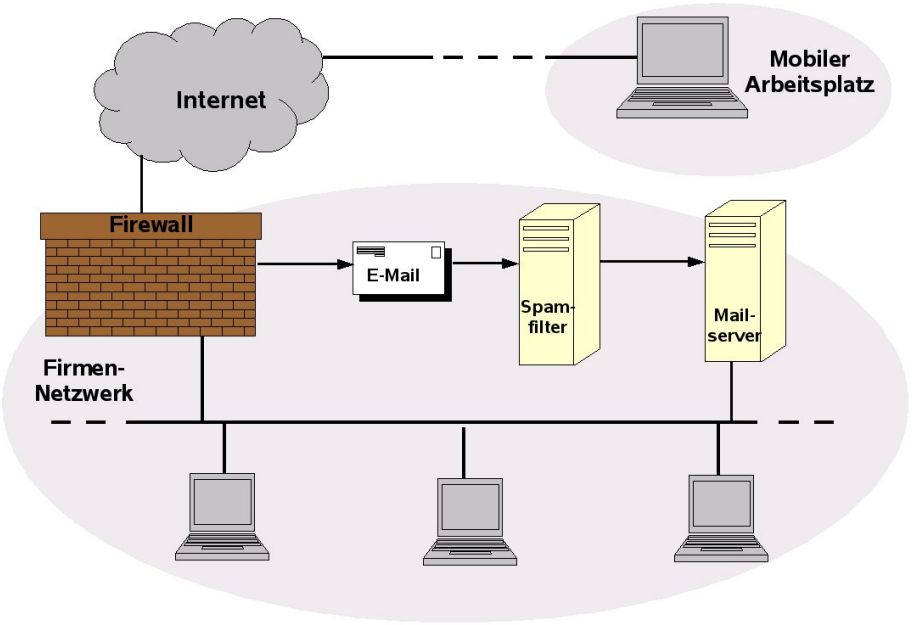

Abbildung 9.2: *Möglicher Aufbau eines Firmennetzwerkes mit einem Mailserver und einem in einem Gateway integrierten Spamfilter*

Der Einsatz des Spamfilters erfolgt wie in Abbildung 9.2 dargestellt auf einem dem Mailserver vorgeschalteten Gateway [Sch05]. Hier muss sichergestellt werden, dass ein Unterscheidung der einzelnen Accounts der Anwender erfolgen kann um sicherzustellen, dass die lernfähigen Filterverfahren ausschließlich in Zusammenhang mit den betroffenen Accounts eingesetzt werden, um die individuelle Lernfähgigkeit zu garantieren. Die eigentliche Behandlung und Klassifizierung der E-Mail erfolgt bei Eintreffen der E-Mail.

Jede eintreffende E-Mail wird zunächst dem Spamfilter, zum Beispiel SpamAssassin, übergeben. Innerhalb des Spamfilters werden nun, wie in Abbildung 9.1 dargestellt, die einzelnen Tests durchgeführt und ein Punktewert für die E-Mail berechnet. Die Ergebnisse der Spamfilterung werden im Header protokolliert. Anschließend wird die E-Mail dem Mailserver übergeben, der diese im Postfach des Anwenders ablegt. Der E-Mail Client liest nun die Informationen des Spamfilters aus dem Header aus. Abhängig vom festgelegten Punktewert wird die E-Mail den Benutzerordnern oder einem eigens für Spam angelegten Ordner übergeben.

Kapitel 10

Analyse

Im Rahmen der Analyse werden die Ergebnisse dargestellt, die mittels des in Kapitel 9 dargestellten Filters erreicht wurden. Die mittels der Kompressionsverfahren erzielten Werte werden getrennt voneinander analysiert. Die Verwendung verschiedener Kompressionsverfahren hat den Zweck, ein für die Spamfilterung geeignetes Verfahren zu bestimmen, da die Güte des Filters vom verwendeten Kompressionsverfahren abhängt. Im Rahmen der Analyse wurden im Einzelnen die Kompressionsverfahren `gzip` und `rar` betrachtet. Header und Body wurden getrennt voneinander bewertet. So ist es möglich, besser auf die Charakteristiken bestimmter Typen von E-Mails einzugehen, die sich oft bereits in kleineren Teilen der E-Mail wiederspiegeln. Desweiteren ist es für die Spammer schwieriger, Einfluss auf die Daten des Header zu nehmen.

Die eigentliche Analyse erfolgt in sechs Schritten. Zunächst erfolgt die Bestimmung eines für die Spamfilterung geeigneten Kompressionsgrades. Jeder der verwendeten Kompressionsalgorithmen bietet die Verwendung verschiedener Kompressionsparameter an, mit denen der Kompressionsgrad beeinflusst werden kann. Im Rahmen der Analyse des Kompressionsgrades wird geprüft, ob ein optimaler Kompressionsgrad bestimmt werden kann. Der optimale Kompressionsgrad hängt sowohl von den korrekten Ergebnissen, als auch von der Art der Fehler seitens des Filters ab.

Anschließend erfolgt ein Vergleich der Ergebnisse, die mittels der Berechnung der relativen Entropie und der Ergebnisse von *Delta* erreicht wurden (siehe Funktionen 8.20 und 9.1). Hier ist ein Vergleich der Ergebnisse der beiden Methoden notwendig. Ziel ist es festzustellen, welche der Funktionen die besseren Werte liefert. Im Anschluss daran erfolgt die Bewertung des Filters in Abhängigkeit von der Korpusgröße. Die Korpora der Ham- und der Spam-Mails sind selten gleich groß. Hier ist es sinnvoll, die Filterergebnisse von den unterschiedlichen Korpusgrößen abhängig zu machen, um so

unter Umständen eine Möglichkeit für eine Optimierung der Ergebnisse zu erhalten.

Im weiteren Verlauf erfolgt nun eine Bewertung der Kompressionsverfahren anhand der Größe des verwendeten Wörterbuchs. Das Wörterbuch beeinflusst die Kompression maßgeblich. Desweiteren beschränkt die Größe des Wörterbuchs die Anzahl der E-Mails, die zur Bewertung einer neu eintreffenden E-Mail herangezogen wird. Hier ist zu prüfen, welchen Einfluss die Größe des Wörterbuchs tatsächlich auf die Spamfilterung hat und wie diese Ergebnisse für eine Optimierung des Filters genutzt werden können.

Anschließend erfolgt eine Bewertung des Lernverhalten des Filters. Innerhalb dieses Analyseschrittes soll geprüft werden, welchen Einfluss die Korpusgröße auf die Lernkurven des Filters hat und ob mittels der durchgeführten Tests endgültige Aussagen bezüglich der Konvergenz des Verfahrens getroffen werden können. Der implementierte Spamfilter trifft Filterentscheidungen in Abhängigkeit von bisher getroffenen Entscheidungen. Die Filterentscheidungen sollten mit steigender Größe des Korpus besser werden, so dass sich die Entscheidungen als Lernkurve darstellen lassen. Abschließend erfolgt die Bewertung des Filters in Abhängigkeit von der Berechnungszeit. Ein Kriterium für die Realisierbarkeit eines Filters ist die für die Klassifizierung einer E-Mail notwendige Zeit, die aufgrund der verwendeten Kompressionsverfahren stark von der Größe des Korpus abhängt. Sind bei steigender Korpusgröße starke Performanzeinbußen zu beobachten, ist unter Umständen eine Optimierung der Rechenzeit notwendig. Hier ist zu prüfen ob es möglich ist, die Größe des zugrundeliegenden Korpus zu verändern, ohne Einbußen in Bezug auf die Klassifikationsergebnisse zu erleiden.

Bei den Ergebnissen der einzelnen Analysereihen handelt es sich um eine Zusammenfassung der Ergebnisse mehrerer Trainigsdurchläufe. Analysiert wurden im Einzelnen fünf verschiedene Korpora mit 1000 E-Mails, die jeweils unterschiedliche Mengen an Ham- und Spam-Mails enthielten, um auch eine Bewertung anhand der Zusammensetzung und Größe der Korpora zu ermöglichen. Alle E-Mails wurden gemäß ihrem zeitlichen Eintreffen klassifiziert. Der Korpus wurde mit jeweils einer Ham- und einer Spam-Mail initialisiert. Zur unterstützenden Darstellung werden die Analyseergebnisse anhand von Tabellen und Diagrammen dargestellt.

Desweiteren wurde eine Langzeitanalyse durchgeführt. Im Rahmen dieser Testreihe wurde eine Menge von 10.000 E-Mails klassifiziert. Die Vorklassifizierung der Korpora erfolgte mit 500 E-Mails. Ziel dieser erweiterten Tests ist zum einen die Verifikation der bereits erzielten Ergebnisse, als auch die Analyse des Filterverhaltens bei großen Mengen eintreffender E-Mails. Dies betrifft zum einen die Bewertung weiterer Lernkurven und die damit verbundene Konvergenz der Werte gegen den maximalen Lernerfolg, als auch

die Bewertung der Performanz des Filters. Auf einem Mailserver einer größeren Firma treffen sehr viele E-Mails ein, die analysiert, bewertet und an den Mail-Client der Anwender weitergeleitet werden müssen. Mittels dieser Langzeitanalyse soll diese Situation simuliert und hinsichtlich der Realisierbarkeit des Filters und dessen Performanz in großen Einsatzumgebungen bewertet werden.

10.1 Bestimmung eines geeigneten Kompressionsgrades

Im Rahmen der Analyse der Kompressionsparameter wird die Qualität des Filters in Abhängigkeit vom gewählten Kompressionsgrad dargestellt. Ziel ist die Auswahl eines möglichst optimalen Kompressionsgrades für die weitergehende Analyse. Ein Kompressionsgrad ist dann optimal, wenn die Anzahl der korrekten Ergebnisse maximiert und die Anzahl der falschen Entscheidungen minimiert werden. Bei Verwendung eines einzelnen Verfahrens ist es in der Regel nicht möglich fehlerhafte Ergebnisse zu vermeiden, da vor allem neue Typen von E-Mails zunächst "gelernt" werden müssen, bis sie vom Filter korrekt erkannt werden. Eine generelle Minimierung fehlerhafter Ergebnisse ist jedoch oft nicht sinnvoll, da eine Unterscheidung zwischen falsch-positiven und falsch-negativen Fehlern erfolgen muss. Vorrangig soll eine Minimierung der falsch-positiven erfolgen, da diese sich kritischer auf den E-Mail Verkehr auswirken, als dies bei falsch-negativen der Fall ist.

Im Rahmen der Analyse wurden die Kompressionsverfahren gzip und rar betrachtet. Bei gzip werden neun Kompressionsparamter unterschieden (siehe Abschnitt 6.6.1). Mittels der Kompressionsparameter wird das *lazy matching* beeinflusst, das heißt die Länge der Zeichenkette, die im Vorschau-Fenster gesucht wird. Je größer der Kompressionsparameter, desto länger die im *lazy matching* verwendete Zeichenkette. Die Größe des Vorschaufensters beträgt bei gzip 32 KByte und wird durch eine Veränderung der Kompressionsparameter nicht beeifnlusst.

Bei rar können fünf verschiedene Kompressionsparameter unterschieden werden (siehe 6.6.2). Diese Kompressionsparameter beeinflussen den Kompressionsgrad durch Verwendung verschiedener Algorithmen und Verfahren. Während im Rahmen von Kompressionsgrad eins und zwei generelle Algorithmen eingesetzt werden, werden bei Angabe der anderen Kompressionsparameter weitere Algorithmen verwendet, die vor allem die Bild- und Audioverarbeitung betreffen. Spezielle Algorithmen für die Kompression von Texten, wie der PPM-Algorithmus (siehe Sektion 6.5.2), werden lediglich bei

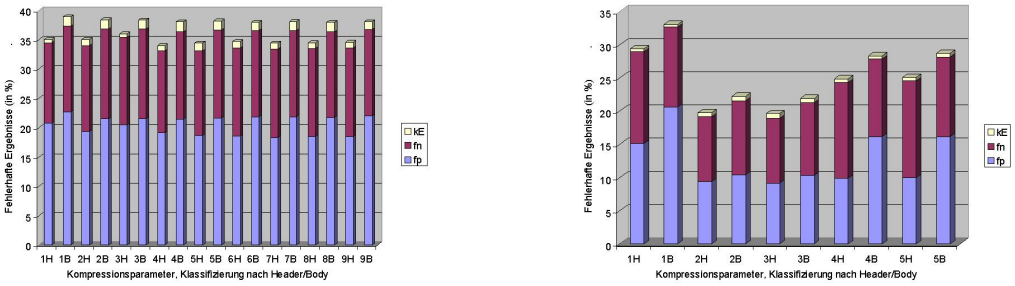

Abbildung 10.1: *Klassifikationsergebnisse für die Kompressionsalgorithmen gzip (links) und rar (rechts) in Abhängigkeit vom Kompressionsparameter und von Header und Body. Dargestellt sind die prozentualen Fehlentscheidungen, aufgeschlüsselt in falsch-positive (fp), falsch-negative (fn) und keine mögliche Entscheidungen (kE)*

Verwendung von Kompressionsgrad vier und fünf eingesetzt.

Die Menge der fehlerhaften Entscheidungen ist in Abbildung 10.1 beispielhaft für die Berechnung der Werte mittels *Delta* dargestellt (siehe Funktion 9.1). Neben der Unterscheidung der Kompressionsverfahren gzip und rar ist es weiterhin notwendig, zwischen der Analyse des Header und des Body der E-Mail zu unterscheiden. Auftretende Fehler sind die falsch-positive oder die falsch-negative Klassifizierung einer E-Mail. Weiterhin ist es möglich, dass anhand der vorliegenden Daten keine Entscheidung bei der Klassifizierung getroffen werden konnte. Dies ist dann der Fall, wenn die für den Ham- und den Spam-Korpus berechneten Werte identisch sind.

Eine Analyse der in Abbildung 10.1 dargestellten Ergebnisse von gzip macht deutlich, dass sich im Rahmen der Klassifizierung von E-Mails kaum Unterschiede in Bezug auf die Kompressionsfaktoren ergeben. Dies ist darauf zurückzuführen, dass gzip ein festes Vorschaufenster von 32KB verwendet und mit dem Kompressionsparameter lediglich eine Variation der Länge der Zeichenketten des *lazy matching* erfolgt. Die Suche nach längeren Treffern zur Verbesserung der Kompressionsrate beeinflusst die Erkennungsrate des Spamfilters kaum. Dies trifft auf Header und Body zu. Sowohl in den Ham- als auch in den Spam-Mails sind eine große Menge für die Klassifizierung irrelevanter Wörter enthalten. Die für den jeweiligen Typ charakteristischen Wörter können oft schon mittels sehr kurzer Zeichensequenzen identifiziert werden, so dass eine Suche nach längeren passenden Zeichenketten keine Verbesserung der Klassifizierungsresultate mit sich bringt. Im Rahmen der weiteren Analyse wird Kompressionsparameter 4 als Referenzparameter verwendet, da dieser im Vergleich zu den anderen Kompressionsgraden im Durchschnitt etwas bessere Werte liefert.

Bei Betrachtung der Darstellung der Ergebnisse von rar wird deutlich, dass sich hier sehr große Unterschiede zwischen den einzelnen Kompressionsparametern ergeben. Dies ist darauf zurückzuführen, dass im Rahmen der verschiedenen Kompressionsparameter verschiedene Algorithmen eingesetzt werden und nicht wie bei gzip lediglich eine Veränderung der für die Kompression relevanten Parameter erfolgt. Zunächst ist festzustellen, dass Kompressionsparameter 1 im Vergleich zu den anderen Parametern sehr schlechte Werte liefert. Mittels Parameter 1 wird die schnellste Kompression durchgeführt [Ros06]. Diese schnelle Kompression liefert jedoch nur geringe Kompressionsraten. Dies könnte der Grund für die schlechten Ergebnisse des Kompressionsparameter 1 sein. Auch Kompressionsparameter 4 und 5 liefern im Vergleich schlechtere Ergebnisse. Bei der Kompression mittels dieser Parameter werden spezielle Algorithmen zur Kompression von Texten eingesetzt. Im Einzelnen betrifft dies das PPM-Schema (siehe Abschnitt 6.5.2). Aufgrund der schlechten Ergebnisse wird deutlich, dass das PPM-Schema für die Spamfilterung nicht geeignet ist. Die besten Ergebnisse liefern Kompressionsparameter 2 und 3. Diese Kompressionsparameter unterscheiden sich dadurch, dass bei Parameter 2 generelle Algorithmen eingesetzt werden, während bei Parameter 3 spezielle Algorithmen ihren Einsatz finden. Dies betrifft vor allem spezielle Algorithmen zur Bild- und Audiobearbeitung, die jedoch keinen Einfluss auf die Kompression von Texten haben. Optimal im Sinne obiger Definition stellt sich Kompressionsparameter 3 dar, da die Anzahl der falsch-positiven sowie die Anzahl aller falschen Entscheidungen hier am geringsten ist.

Weiterhin wird deutlich, dass rar grundsätzlich bessere Ergebnisse als gzip liefert, da der durchschnittliche Anteil der Fehler und der Anteil der falsch-positiven Klassifizierungen wesentlich geringer ist. Dies könnte darauf zurückzuführen sein, dass rar ein Wörterbuch der Größe 4096KB verwendet, während das Vorschaufenster von gzip, das das Wörterbuch darstellt, auf 32KB beschränkt ist. Mittels rar ist es somit möglich, auf weiter zurückliegende Entscheidungen zurückzugreifen. Weiterhin ist festzustellen, dass die mittels der Analyse des Header erzielten Ergebnisse besser sind als die Ergebnisse des Body. Dies ist zunächst darauf zurückzuführen, dass viele Felder des Header nicht direkt vom Anwender beeinflusst werden können und somit charakteristisch für den Typ der E-Mail sind. Hier sind vor allem die Einträge der Gateways zu nennen, die den Weg der E-Mail beim Versand protokollieren, als auch der absendende Server, der im Header vermerkt ist. Die betroffenen Server sind vor allem bei Ham-Mails bekannt, da sie sich nicht verändern. Ein weiteres Charakteristikum ist das Betreff-Feld, das sich bei vielen Spam-Mails sehr ähnlich ist. Bei der Analyse des Body besteht das Problem, dass eine Vielzahl irrelevanter Wörter in der E-Mail enthalten ist. Diese Wörter werden jedoch mit komprimiert und benötigen Platz im Wör-

terbuch. Überschreibt das Wörterbuch die vorgesehene Grenze, so werden einige Ausdrücke nicht mehr betrachtet. Dies könnte der Grund sein, warum die Differenz der Ergebnisse von Header und Body bei rar geringer ist als bei gzip: Das von rar verwendete Wörterbuch ist viel größer, so dass auf eine längere "Geschichte" der empfangenen E-Mails zurückgegriffen werden kann.

10.2 Vergleich der Werte der relativen Entropie und Delta

In Abschnitt 10.1 wurden lediglich die mittels der Berechnung von *Delta* erzielten Ergebnisse betrachtet. Im Rahmen dieses Kapitels soll nun ein Vergleich der mittels *Delta* und mittels der relativen Entropie erzielten Ergebnisse erfolgen. Der Vergleich der Ergebnisse erfolgt anhand des in Abschnitt 10.1 bestimmten Kompressionsparameters. Im Einzelnen ist dies Kompressionsparameter 4 für gzip und Parameter 3 für rar. Die Ergebnisse der Vergleiche sind in Abbildung 10.2 dargestellt. Relevant sind neben den Endergebnissen, die die Gesamtanzahl fehlerhafter Entscheidungen darstellen, auch die jeweiligen Lernkurven. In den Lernkurven wird die prozentuale Anzahl korrekter Entscheidungen in Abhängigkeit von der Korpusgröße dargestellt. Im Laufe der Zeit konvergieren diese Kurven gegen einen festen Wert.

Aus Abbildung 10.2 wird deutlich, dass mittels der Berechnungsmethode der relativen Entropie grundsätzlich schlechtere Werte erreicht werden, da vor allem prozentual mehr falsch-positive bei der Filterung auftreten. Eine Ausnahme bildet die Berechnung der Entscheidungen des Body mittels gzip, bei dem mittels der Berechnung der relativen Entropie bessere Werte erreicht werden als mittels *Delta*. Bei der Berechnung der Ergebnisse für die Filterentscheidung mittels der relativen Entropie erfolgt eine Normierung der Werte mittels der Dateigröße der jeweiligen Korpora (siehe Funktion 8.20). Die Dateigrößen des Body unterliegen starken Schwankungen, so dass die Normierung dieser Dateigrößen eine positive Auswirkung auf die Filterentscheidung hat, während die Dateigrößen der Korpora der Header geringeren Schwankungen unterliegen, wodurch eine Normierung die Ergebnisse nicht positiv beeinflusst.

Sowohl bei der Berechnung der Werte mittels *Delta* als auch bei Berechnung der relativen Entropie sind die Ergebnisse aufgrund des verwendeten Wörterbuchs beschränkt. Ältere auf dem System vorhandene E-Mails werden nur dann für Filterentscheidungen herangezogen, wenn sich deren Einträge noch im Wörterbuch befinden. Da rar ein größeres Wörterbuch verwendet, sind die Ergebnisse von rar (grundsätzlich) besser.

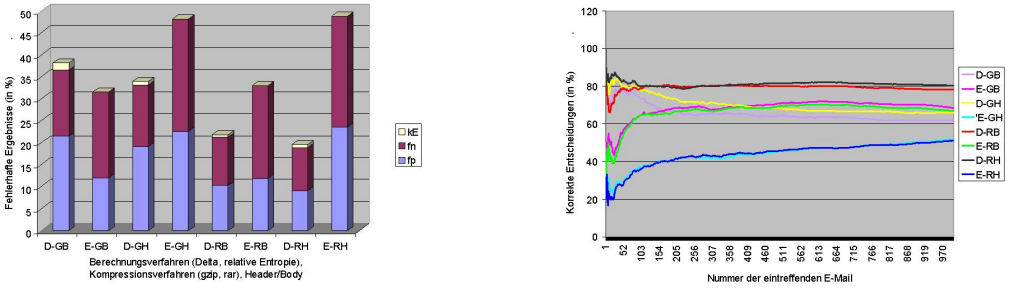

Abbildung 10.2: *Klassifikationsergebnisse (links) und Lernkurven (rechts) für die Filterentscheidungen unter mittels Delta und der relativen Entropie unter Verwendung der Kompressionsalgorithmen gzip und rar in Abhängigkeit vom analysierten Bereich der E-Mail (Header/Body) bei der Analyse von 10000 E-Mails. Dargestellt sind die Klassifikationsergebnisse für falsch-positive (fp) und falsch-negative Entscheidungen (fn), sowie keine Entscheidung (kE)*

Um konkrete Aussagen über die Filterentscheidungen zu treffen, können die in Abbildung 10.2 dargestellten Lernkurven herangezogen werden. Aus diesen Kurven wird deutlich, dass die Anzahl korrekter Entscheidungen, die mittels der Berechnungsmethode *Delta* erreicht werden, spätestens nach 250 E-Mails konvergieren. Dies scheint bei der Berechnung der Entscheidungen mittels der relativen Entropie nicht der Fall zu sein. Der Grund dafür kann sein, dass eine Abschätzung der relativen Entropie mittels Kompression gemäß Definition 18 eine sehr lange Sequenz voraussetzt. Der Wert konvergiert, wenn die Länge der betrachteten Sequenz gegen unendlich geht. Um konkrete Aussagen über die Konvergenz in Bezug auf die relative Entropie treffen zu können ist es somit notwendig, eine größere Menge (eintreffender) E-Mails zu analysieren.

In Abbildung 10.3 sind die Lernkurven für die Analyse von 10.000 E-Mails dargestellt. Analysiert wurden insgesamt 2.419 Ham- und 7.581 Spam-Mails und es wurde ein Korpus mit jeweils 500 vorklassifizierten E-Mails verwendet. Zunächst ist festzustellen, dass auch bei Langzeittests mittels der Berechnungsmethode der relativen Entropie grundsätzlich mehr Fehler entstehen als bei der Analyse mittels der Berechnung von *Delta*. Während sich die Ergebnisse bei der Berechnung der Werte mittels *Delta* nur gering ändern, verschlechtert sich die Gesamtmenge falscher Entscheidungen bei der Berechnung mittels der relativen Entropie deutlich. Eine Unterscheidung in die Art der auftretenden Fehler zeigt jedoch, dass mit steigender Anzahl von E-Mails bei Berechnung der relativen Entropie auch die Menge der falsch-positiven Ergebnisse vor allem im Vergleich zur Berechnungsmethode *Delta*

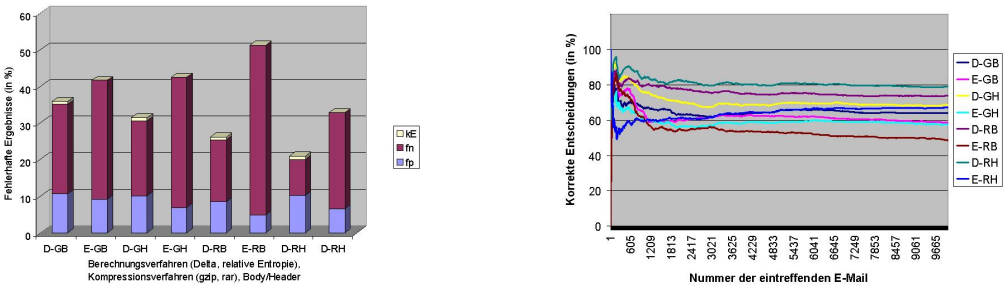

Abbildung 10.3: *Klassifikationsergebnisse (links) und Lernkurven (rechts) für die Filterentscheidungen unter mittels Delta und der relativen Entropie unter Verwendung der Kompressionsalgorithmen gzip und rar in Abhängigkeit vom analysierten Bereich der E-Mail (Header/Body) bei der Analyse von 10000 E-Mails. Dargestellt sind die Klassifikationsergebnisse für falsch-positive (fp) und falsch-negative Entscheidungen (fn), sowie keine Entscheidung (kE)*

sinkt. Dies bestätigt Definition 18: Die Entropie kann durch die Kompression von Sequenzen abgeschätzt werden, wenn die Länge der betrachteten Sequenz gegen unendlich geht.

10.3 Darstellung in Abhängigkeit von der Korpusgröße

In einem realen System treffen selten gleich viele Ham- und Spam-Mails ein, so dass sich die Größen der Ham- und Spam-Korpora immer weiter auseinander divergieren. Im Rahmen dieses Abschnitts ist zu bewerten, ob und wenn ja welche Auswirkungen eine Differenz der Korpusgrößen Auswirkungen auf die Filterentscheidungen hat. Liegen derartige Auswirkungen vor, so sind diese im Rahmen der weiteren Analyse zu berücksichtigen. Weiterhin können sich unter Umständen Ansätze zur Verbesserung der Filterraten ergeben.

In Abbildung 10.4 ist zunächst eine Übersicht über die Trainingsmengen dargestellt. Die Trainingsmengen enthalten eine unterschiedliche Anzahl an Ham- und Spam-Mails gemäß ihrem Eintreffen auf dem System. Dies simuliert zum einen den realistischen E-Mail Verkehr, zum anderen soll so geprüft werden, welche Auswirkung eine Veränderung der Differenz der Korpusgrößen auf die Filterentscheidungen hat. Die Differenzen der Korpusgrößen, die sich erst im Laufe der Zeit ergibt, sind detailliert in Abbildung 10.5 dargestellt.

Die Auswirkung der Differenz der Korpusgrößen auf die Filterentschei-

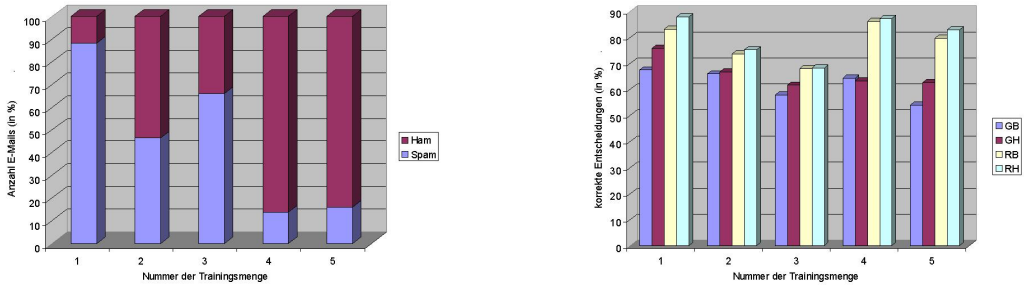

Abbildung 10.4: *Differenz der Korpusgrößen in Bezug auf die Anzahl von Spam- und Ham-Mails (links) und korrekte Entscheidungen in Abhängigkeit vom verwendeten Korpus (rechts)*

dungen ist auf der rechten Seite der Abbildung 10.4 dargestellt. Zunächst ist festzustellen, dass sich die Filterergebnisse nicht durch Auswahl ähnlich großer Korpora verbessern. Die Veränderung der Korpusgröße hat weiterhin eine stärkere Auswirkung auf das Kompressionsverfahren `rar`. Somit stellt sich das Verfahren gzip als robuster gegen Veränderungen des Korpus dar. Dies kann seine Ursache darin haben, dass das Vorschaufenster von rar grundsätzlich größer ist als der Korpus (siehe Abbildung 10.5). Aufgrund des kleinen Vorschau-Fensters von gzip sind nur geringe Veränderungen der Entscheidungen festzustellen.

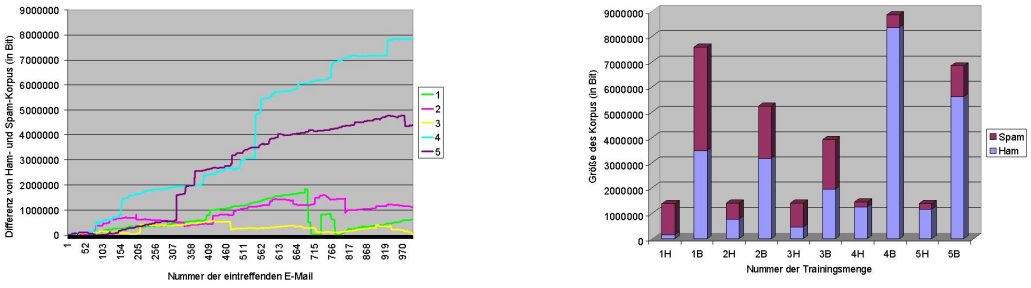

Abbildung 10.5: *Absolute Differenz der Korpusgrößen des Body in Bit in Abhängigkeit vom verwendeten Korpus (links) und Gesamtgrößen der verwendeten Korpora in Bits (rechts)*

In Abbildung 10.5 ist die Differenz der Korpusgrößen in Bit dargestellt, abhängig von der Situation einer gerade eintreffenden E-Mail. Hier wird deutlich, dass der Ham-Korpus grundsätzlich stärker wächst als der Spam-Korpus und somit Korpus 4 und 5 sehr stark auseinander divergieren, während die restlichen Korpora sehr ähnliche Dateigrößen aufweisen. Weiterhin

sind in Abbildung 10.5 die Gesamtgrößen der verwendeten Korpora dargestellt. Grundsätzlich sind die Korpora der Header ähnlich groß. Der Header beinhaltet zwar nur wenige Pflichtfelder (siehe Abschnitt 2.1.2), die meisten besitzen jedoch einen ähnlichen Aufbau, da die einzelnen Headerzeilen von den E-Mail Clients und den Zwischenknoten beim Versand hinzugefügt werden. Desweiteren ist festzustellen, dass die Dateigrößen der Korpora der Header geringer sind als die der Korpora der Body der E-Mail. Die Menge der Zeilen des Body ist sehr variabel. Vor allem E-Mails, die zum Beispiel Geschäftsberichte oder -Protokolle enthalten oder die über eine Newsgroup versendet werden und mehrere Beiträge enthalten, können sehr groß werden.

10.4 Bewertung des Filters in Abhängigkeit von den Lernkurven

Ziel dieses Abschnitts ist die Darstellung der Lernkurven in Abhängigkeit von der Anzahl der eintreffenden Ham- und Spam-Mails und in Abhängigkeit vom verwendeten Korpus. Anhand derartiger Lernkurven können weiterführende Aussagen in Bezug auf die Lernerfolge des Filters getroffen werden. So ist zu erwarten, dass die Anzahl erkannter E-Mails nach einer Zeit t gegen einen festen Wert konvergiert, so dass der Filter anhand dessen bewertet werden kann. Die Lernkurven geben somit an, wann eine Beschränkung der Lernfähigkeit des Filters erfolgt und ermöglichen eine Analyse dieser Beschränkung. Die Lernkurven werden zum einen anhand des verwendeten Kompressionsverfahrens dargestellt (gzip und rar) und zum anderen anhand des bewerteten Bereiches der E-Mail (Header und Body).

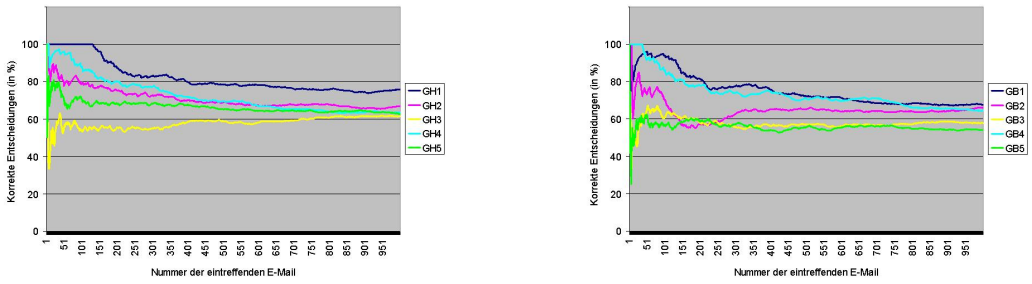

Abbildung 10.6: *Lernkurven unter Verwendung von gzip bei Analyse des Header (links) und des Body (rechts) in Abhängigkeit von unterschiedlicher Zusammensetzung des verwendeten Korpus*

In Abbildung 10.6 sind zunächst die mittels des Kompressionsverfahrens

gzip erreichten Ergebnisse in Form von Lernkurven dargestellt. Grundsätzlich stellt sich die Analyse des Header mittels gzip robuster gegenüber Schwankungen dar als die Analyse des Body. Dies hat seinen Grund vor allem darin, dass die Informationen des Body einfacher durch den Anwender beeinflussbar sind als die des Header. Die Daten des Header können zwar nicht als authentisch vorausgesetzt werden, eine Änderung der Art der Spam-Mails bezieht sich jedoch in der Regel nur auf den Body. Immer dann, wenn eine neue Art von E-Mail auftritt, können sich Schwankungen in den Lernkurven ergeben. Bei neuen Arten von E-Mails kann es sich zum einen um eine neue Art von Spam handeln, aber auch um noch unbekannte Ham-Mails, wie zum Beispiel Beiträge aus einer neuen Art von Mailing-Listen. Weiterhin beträgt das Vorschaufenster, und somit das Wörterbuch von gzip 32KB (siehe Sektion 6.6.1). Mit Ausnahme der Spam-Mails von Korpus 4 wird dieses Fenster von allen Korpora überschritten. Dies ist der Grund, warum bei allen Korpora bereits nach einer geringen Anzahl von E-Mails nur noch geringe Schwankungen der Lernkurven zu beobachten sind.

In Abbildung 10.7 sind die mittels des Kompressionsverfahrens rar erreichten Klassifikationsergebnisse in Form von Lernkurven dargestellt. Sowohl die Analyse des Header als auch die des Body unterliegen nur geringen Schwankungen und die Ergebnisse liegen alle in einem ähnlichen Wertebereich. Im Rahmen der Analyse verwendet rar ein Wörterbuch von 4096KB (siehe Sektion 6.6.2). In Abbildung 10.5 wird deutlich, dass mit Ausnahme der Ham-Mails von Korpus 4 alle Korpora kleiner sind als das Wörterbuch. Dies hat zur Folge, dass bei der Analyse alle Informationen der bereits eingetroffenen E-Mails im Wörterbuch verfügbar sind. Desweiteren kann nicht davon ausgegangen werden, dass die erreichten Ergebnisse gegen einen endgültigen Wert konvergieren, da die Kapazität des Wörterbuches noch nicht überschritten ist. Eine Konvergenz der Werte muss mittels einer größeren Testreihe überprüft werden, deren Korpusgrößen die Größe des verwendeten Wörterbuches übersteigen. Derartige Ergebnisse sind bereits in Abbildung 10.3 dargestellt und im zugehörigen Abschnitt analysiert.

Grundsätzlich ist festzustellen, dass die Ergebnisse von gzip schlechter sind als die mittels des Kompressionsverfahrens rar erreichten Werte. Dies kann seine Ursache in den unterschiedlich großen Wörterbüchern haben. Der Einfluss des Wörterbuchs wird im Rahmen des folgenden Abschnitts geprüft. Weiterhin ist festzustellen, dass sich rar und gzip bei der Analyse einer Testreihe in den erzielten Ergebnissen unterscheiden. Dies ist zum Beispiel bei Korpus 5 der Fall, der im Vergleich zu den anderen Korpora von gzip schlechter bewertet wird als von rar. Somit ist es möglich, dass der Filter durch eine Kombination der Ergebnisse der Kompressionsverfahren bessere Ergebnisse liefert als dies bei Verwendung eines einzelnen Verfahrens der Fall

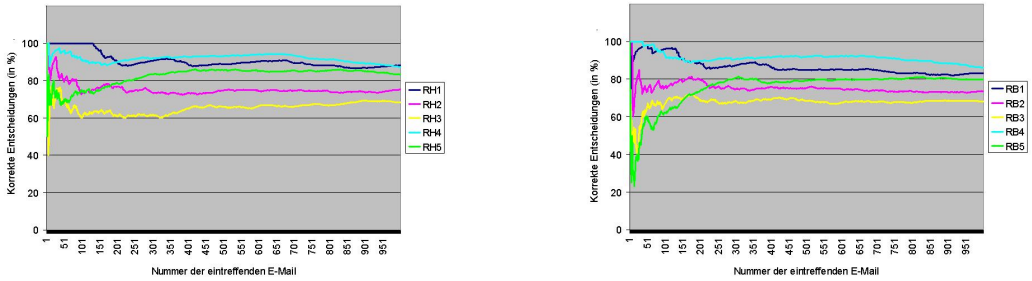

Abbildung 10.7: *Lernkurven unter Verwendung von rar bei Analyse des Header (links) und des Body (rechts) in Abhängigkeit von unterschiedlicher Zusammensetzung des verwendeten Korpus*

ist.

Beide Kompressionsverfahren stellen sich robust gegen Änderungen der Differenz der Dateigrößen dar, obwohl bei der Berechnung der Filterentscheidung mittels *Delta* keine Normierung erfolgt. Dies kann seine Ursache darin haben, dass die Differenz der Dateigrößen nach der Kompression entscheidend ist. Je kleiner diese Differenz ist, desto besser konnte die neu eintreffende E-Mail komprimiert werden und umso mehr Zeichenfolgen waren bereits im Wörterbuch vorhanden.

10.5 Bewertung des Filters in Abhängigkeit vom verwendeten Wörterbuch

Im Rahmen dieses Abschnitts erfolgt die Bewertung des Filters in Abhängigkeit von der Größe des verwendeten Wörterbuchs. Analysiert wurde Korpus 1 mit dem Kompressionsverfahren rar unter Verwendung eines Wörterbuchs in sieben verschiedenen Größen. In Abbildung 10.8 sind die Ergebnisse dieser Analysereihe in Abhängigkeit vom analyiserten Bereich der E-Mail (Header/Body), von dem verwendeten Wörterbuchgröße und von der Art der entstandenen Fehler dargestellt. Ziel ist die Bewertung des Einflusses des Wörterbuches auf die Filterentscheidung, da sich hinsichtlich dieser Faktoren Möglichkeiten zur Optimierung und zur Steigerung der Performanz ergeben können.

Aus Abbildung 10.8 wird zunächst deutlich, dass die Verwendung eines größeren Wörterbuches eine positive Auswirkung auf das Auftreten falschnegativer Ergebnisse hat, da diese reduziert werden. Weiterhin hat eine Veränderung der Größe des Wörterbuches stärkere Auswirkungen auf die Analyse

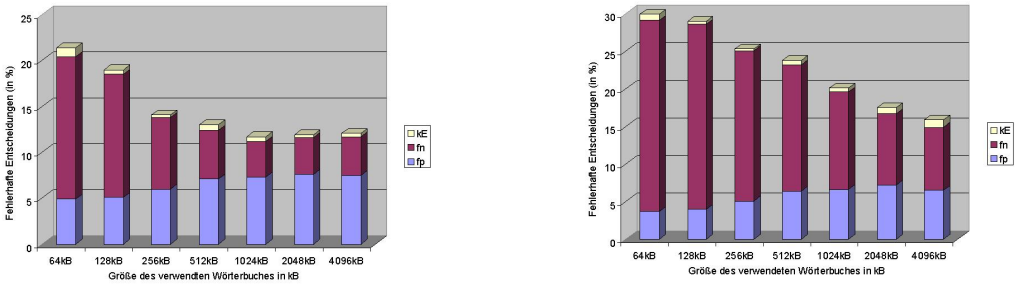

Abbildung 10.8: *Darstellung der fehlerhaften Entscheidungen, die mittels verschiedener Wörterbuchgrößen unter Verwendung des Kompressionsverfahrens rar bei der Analyse des Header (links) und des Body (rechts) erreicht wurden. Dargestellt sind falsch-positive Entscheidungen (fp), falsch-negative Entscheidungen (fn) und keine Entscheidung (kE)*

des Body. Dies kann seine Ursache darin haben, dass der Korpus des Body stärker wächst als der des Header und die Größe des Wörterbuches somit schneller überschritten wird. Dies wird im Einzelnen anhand einer konkreten Analyse der Ergebnisse deutlich. Der Korpus des Header erreicht eine maximale Größe von 128KB für die Spam-Mails und 1024KB für die Ham-Mails (siehe Abbildung 10.5). Somit sind bei Verwendung der Wörterbücher mit 1024KB, 2048KB und 4096KB nur geringfügige Veränderungen zu erkennen. Die Korpusgrößen des Body überschreiten 2056KB, so dass eine Veränderung der Wörterbuchgröße stärkere Auswirkungen auf die Analyseergebnisse hat, da nicht alle Informationen des Korpus im Wörterbuch vorgehalten werden können.

Ein Vergleich mit Wörterbuchgrößen von 64KB erzielten Ergebnisse mit den Ergebnissen, die für Korpus 1 mittels des Kompressionsverfahrens gzip erzielt wurden zeigt weiterhin, dass rar bei Verwendung eines kleinen Wörterbuchs bei der Analyse des gleichen Korpus grundsätzlich ähnliche Ergebnisse liefert wie gzip (siehe Abbildung 10.8 und 10.4). Während gzip bei einer Analyse des Header 75% der E-Mails korrekt klassifizierte, konnten bei der Analyse des Header mittels rar 79% der E-Mails korrekt eingeordnet werden. Im Rahmen der Analyse des Body ordneten sowohl gzip als auch rar 68% der eintreffenden E-Mails korrekt ein. Eine Analyse mittels größerer Vorschaufenster für gzip war nicht möglich, da gzip eine derartige Möglichkeit nicht vorsieht.

Schließlich ist festzustellen, dass mittels eines größeren Wörterbuchs die falsch-negativen Ergebnisse stark reduziert werden, die Menge der falsch-positiven Ergebnisse steigt jedoch. Dies betrifft sowohl die Analyse des Hea-

der als auch die Analyse des Body. Für die Implementierung ist somit abzuwägen, zugunsten der falsch-positiven Ergebnisse auf ein großes Wörterbuch zu verzichten.

10.6 Bewertung in Abhängigkeit von der Berechnungszeit

Im Rahmen dieses Kapitels erfolgt eine Bewertung des Filters in Abhängigkeit von der notwendigen Berechnungszeit. Die Bewertung erfolgt zum einen anhand der Grundeinstellung der verwendeten Kompressionsverfahren mittels des im in Abschnitt 10.1 bestimmten Kompressionsgrades (4 für gzip und 3 für rar) und zum anderen anhand der notwendigen Berechnungszeit bei Einsatz des Kompressionsverfahrens rar mittels der Verwendung verschieden großer Wörterbücher. Ein Kriterium für die Realisierbarkeit eines Filters ist die für die Klassifizierung einer E-Mail notwendige Zeit. Die Berechnungszeit ist vor allem mit steigender Größe des Korpus wichtig, um Aussagen über die Performanz und Realisierbarkeit des Filters treffen zu können.

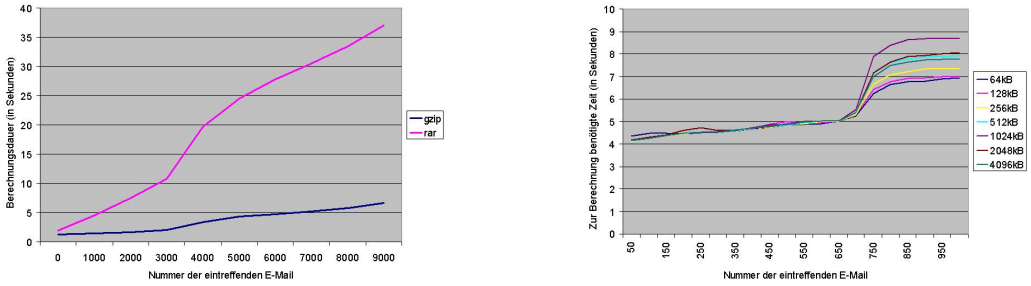

Abbildung 10.9: *Durchschnittliche Berechnungszeit in Abhängigkeit von der Menge der eintreffenden E-Mail und dem verwendeten Kompressionsverfahren für die Berechnung von 10000 E-Mails (links) und in Abhängigkeit von der Größe des verwendeten Wörterbuchs bei dem Kompressionsverfahren rar bei Berechnung von 1000 E-Mails (rechts)*

Wie bereits in den vorangegangen Abschnitten deutlich wurde, liefert rar im Rahmen der Klassifikation von E-Mails die besseren Ergebnisse. Dies geht jedoch auf Kosten der Berechnungszeit, wie in Abbildung 10.9 dargestellt. Die größeree Berechnungszeit für die Klassifizierung ist zum Teil darauf zurückzuführen, dass gzip ein Vorschaufenster von 32KB verwendet, während rar bei der Kompression in den Grundeinstellungen ein Wörterbuch der Größe 4096KB verwendet. Aus Abbildung 10.9 wird weiterhin deutlich, dass die

Berechnungszeit mit sinkender Wörterbuchgröße nur leicht sinkt. Die Größe des Wörterbuches hat somit zwar einen Einfluss auf die Berechnungszeit, ist jedoch nur gering. Einen größeren Einfluss auf die Berechnungszeit haben somit die im Rahmen der Datenkompression verwendeten Algorithmen. In diesem Rahmen stellt sich gzip performanter dar, da weniger komplexe Algorithmen. Im Rahmen der Performanz bleibt nun zu prüfen, ob gzip bei Verwendung eines größeren Vorschaufensters weiterhin performanter arbeitet als rar.

Weiterhin ist in Abbildung 10.9 die notwendige Berechnungszeit in Abhängigkeit von der Wörterbuchgröße dargestellt. Der sprunghafte Anstieg der Berechnungszeit nach rund 670 E-Mails ist auf eine sprunghafte Erhöhung der Dateigröße des Korpus zurückzuführen. Dies wird aus Abbildung 10.5 deutlich. Außerdem wird deutlich, dass eine Veränderung der Wörterbuchgröße zu einer um maximal 2 Sekunden höheren Berechnungszeit führt. Die Geschwindigkeitseinbuße bei Verwendung größerer Wörterbücher des Kompressionsverfahrens rar ist somit sehr gering und zugunsten besserer Ergebnisse zu vernachlässigen. Kritischer stellt sich dagegenüber die starke Differenz der Berechnungszeit im Vergleich zu gzip dar. Aus diesem Grund wird von der Verwendung von rar im Einsatz größerer Umgebungen abgeraten, da zur Berechnung der Filterentscheidung bei einer größeren Menge von E-Mails zu viel Zeit benötigt wird.

Kapitel 11

Fazit

Im Rahmen dieser Arbeit erfolgte die Darstellung der Spamfilterung mittels Texterkennungsverfahren. Dies betrifft zum einen die Darlegung der notwendigen theoretischen Grundlagen als auch die Durchführung verschiedener Trainingsdurchläufe. Im Rahmen der Analyse erfolgte die Bewertung des Filters anhand verschiedener Gesichtspunkte, wie zum Beispiel die Bewertung anhand der Erkennungsraten, anhand der Größe des verwendeten Wörterbuches oder anhand der zur Analyse notwendigen Berechnungszeit.

Grundsätzlich können nur unverschlüsselte E-Mails analysiert werden. Eine Entschlüsselung der E-Mails durch den Spamfilter ist aus Sicherheitsgründen nicht möglich, da die privaten Schlüssel der Anwender weder dem Gateway, auf dem der Spamfilter implementiert ist, noch dem Mailserver bekannt sein sollten. Bisher stellt sich dies nicht als Problem dar, da für das Verschlüsseln einer E-Mail der öffentliche Schlüssel des Empfängers benötigt wird. Wird eine E-Mail verschlüsselt an viele Empfänger verschickt, so müssen alle öffentlichen Schlüssel der jeweiligen Empfänger bekannt sein. Spam-Mails werden an eine Vielzahl von Adressen verschickt von denen die Existenz oft nicht verifiziert werden kann, so dass Spammer ihre E-Mails aufgrund des zur Schlüsselbeschaffung notwendigen Aufwandes aktuell nicht verschlüsseln.

Unabhängig vom verwendeten Kompressionsverfahren liefert die Analyse des Header bessere Ergebnisse als die des Body. Dies ist unter anderem darauf zurückzuführen, dass E-Mails bekannter Absender sich in vielen Abschnitten des Header ähneln. Diese Ähnlichkeiten betreffen vor allem die E-Mail Adresse des Absenders, sowie den sendenden Rechner und die beim Versand passierten Zwischenknoten. Desweiteren beinhalten viele Spam-Mails ähnliche Betreff-Zeilen. Bei der Analyse des Header wird somit zwar auf wenige, aber dafür prägnante Bereiche der E-Mail zurückgegriffen, die eine sehr eindeutige Einordnung der E-Mail erlauben. Weiterhin wächst die Dateigröße des Bo-

dy einer E-Mail stärker als die des Header, so dass sich bei der Analyse des Header Informationen aus einer größeren Menge an E-Mails im Wörterbuch befinden als dies bei der Analyse des Body der Fall ist. Es können somit weiter zurückliegende Entscheidungen berücksichtigt werden.

Die Güte des Spamfilters hängt vor allem vom der Größe des verwendeten Wörterbuches ab, da die Größe des Wörterbuchs des verwendeten Kompressionsverfahren die Lernfähigkeit des Spamfilters begrenzt. Für die Analyse einer neu eintreffenden E-Mail können nur die bereits klassifizierten E-Mails berücksichtigt werden, die sich im Wörterbuch befinden. Ein größeres Wörterbuch führt somit dazu, dass auf eine größere Menge an Informationen zurückgegriffen werden kann. Wie in Abschnitt 10.6 dargestellt, führt die Verwendung eines größeren Wörterbuches nur zu geringen Performanzeinbußen. Demgegenüber führt eine Veränderung des zugrundeliegenden Verfahrens bei ähnlicher Größe des Wörterbuches nur zu geringen Veränderungen der Filterergebnisse. So konnte in Abschnitt 10.5 dargestellt werden, dass rar bei Verwendung eines kleinen Wörterbuchs ähnliche Ergebnisse liefert wie gzip. In weiterführenden Tests ist nun zu prüfen, ob eine Vergrößerung des Vorschaufensters von gzip zu ähnlichen Analyseergebnissen wie rar mit gleich großem Wörterbuch führt. Ist dies der Fall, sollte für den Spamfilter gzip mit einem größeren Vorschaufenster eingesetzt werden, da rar im Rahmen der Spamfilterung eine sehr schlechte Performanz liefert.

Der größte für die Analyse von E-Mails relevante Unterschied der Kompressionsverfahren spiegelt sich in der für die Kompression notwendigen Zeit wieder. So steigt die Berechnungszeit bei Verwendung des Verfahrens rar mit steigender Anzahl analysierter E-Mails stark an, während gzip sich als performanter erweist, was sich in denen im Vergleich sehr geringeren Berechnungszeiten widerspiegelt. Dies hat seinen Grund vor allem in den komplexeren Algorithmen, die von rar zur Datenkompression verwendet werden, da bereits gezeigt werden konnte, dass eine Veränderung der Größe des Wörterbuches sich nicht so stark auf die Berechnungszeiten auswirkt. Da rar bei kleinen Wörterbüchern ähnliche Ergebnisse wie gzip aufweist, ist die Verwendung von gzip mit einem größeren Wörterbuch empfehlenswert. Aufgrund der schlechten Performanz wird vom Einsatz von rar abgeraten.

Es ist nicht sinnvoll, ein einzelnes Verfahren als ausschließliches Verfahren zur Spamfilterung einzusetzen, da ein einzelnes Verfahren leicht umgangen werden kann. Vielmehr ist es sinnvoller, eine Kombination mit anderen Verfahren anzustreben. So ist zum Beispiel eine Kombination mit Black- und Whitelists sinnvoll, um eine Steigerung der Performanz zu erreichen. Mittels Blacklists werden eindeutig als Spam identifizierte E-Mails direkt aussortiert, während E-Mails von bekannten Absendern des Adressbuches gar nicht an die dynamischen Verfahren übergeben werden. Somit kann eine Performanzstei-

gerung erfolgen, da der Einsatz dynamischer Verfahren mit einem größeren Zeitaufwand verbunden ist als der Einsatz statischer Verfahren. Weiterhin könnte das dargestellte Verfahren mit einem Bayes'schen Filter kombiniert werden, um die Effektivität des Filters weiter zu steigern und es Spammern außerdem zu erschweren, den Filter zu umgehen. Um eine Kombination der erreichten Analyseergebnisse zu ermöglichen, bietet sich eine Einbindung in den Spamfilter *SpamAssassin* an.

Eine weitere Steigerung der Erkennungsraten kann mit einer Kombination der Ergebnisse der Texterkennung untereinander erreicht werden. Diese Kombination kann zum Beispiel durch eine Analyse mittels verschiedener Kompressionsverfahren und durch eine Kombination der mittels Header und Body erzielten Ergebnisse erfolgen. Anhand der Analyseergebnisse die im Rahmen des Trainings erreicht wurden, kann für jeden Parameter eine Spamwahrscheinlichkeit ausgerechnet werden. Als Ergebnis wird eine bedingte Wahrscheinlichkeit über alle Werte berechnet. Nun kann der Anwender festlegen, ab welcher Wahrscheinlichkeit eine E-Mail als Spam eingestuft wird. Dies kann zu einer Reduktion der falsch-positiven Ergebnisse und zu einer Erhöhung der Benutzerfreundlichkeit führen.

Die Qualität des Spamfilters wird stark von der Größe des verwendeten Wörterbuches beeinflusst. Die Verwendung eines größeren Wörterbuches kann jedoch zu Performanzeinbußen bei der Analyse führen. Um eine größere Anzahl von E-Mails zu analysieren ohne aufgrund größerer Wörterbücher Leistungseinbußen hinzunehmen, können speziell für die Spamfilterung entwickelte Kompressionsverfahren verwendet werden. Im Rahmen der Analyse wurden verlustfreie Kompressionsverfahren verwendet, deren Ziel die Minimierung der zur Darstellung einer Nachricht notwendigen Zeichen ist. Im Rahmen der Spamfilterung muss nicht notwendigerweise ein verlustfreies Verfahren zur Datenkompression eingesetzt werden, da keine Dekompression des Korpus erfolgt. Bei der Analyse des Header einer E-Mail ist es zum Beispiel möglich, vor der Kompression die Schlüsselwörter der Header zu entfernen, so dass nur der Inhalt der Headereinträge bei der Spamfilterung beachtet wird. Somit wird die Menge der zu analysierenden Daten reduziert und es ist möglich, auch ohne eine Vergrößerung des zugrundeliegenden Wörterbuches weiter zurückliegende Entscheidungen zu berücksichtigen. Bei der Analyse des Body ist es ebenfalls möglich, bei der Kompression auf einen festgelegten Satz an Zeichen zu verzichten. Dies betrifft vor allem die Wörter, die mit einem gleichen Anteil in Spam- und Ham-Mails vorkommen. Hier ist es zum Beispiel möglich, auf die durch einen Bayes'schen Filter angelegten Hash-Tabellen mit den Worthäufigkeiten zurückzugreifen.

Desweiteren ist es möglich, den notwendigen Speicherplatz zu reduzieren, indem die Größe des Korpus auf eine Dateigröße beschränkt wird, die

etwa zweimal der Größe des Vorschaufensters entspricht. E-Mails, die sich außerhalb dieses Vorschau-Fensters befinden werden nicht für die Kompression herangezogen. Dies reduziert zum einen die Menge an Daten, die lokal auf dem System verfügbar sein müssen als auch die für die Kompression notwendige Berechnungszeit.

Um die für die Analyse einer E-Mail notwendige Zeit zu reduzieren ist es möglich, das verwendete Kompressionsverfahren derart zu modizieren, dass ein hinzukomprimieren der Daten möglich ist. Im Rahmen der Analyse war es in jedem Analyseschritt notwendig, den Korpus neu zu komprimieren. Muss der zur Analyse notwendige Korpus nur noch in komprimierter Form vorgehalten werden, kann zum einen der notwendige Speicherplatz reduziert werden. Weiterhin wird durch das Anhängen der E-Mail die Neukompression des bestehenden Korpus vermieden, was sich positiv auf die notwendige Analysezeit auswirken wird. Vor allem in großen Netzwerken, in denen eine Vielzahl an E-Mails in kurzer Zeit analysiert werden muss, ist eine Reduktion der Berechnungszeit anzustreben.

Im Rahmen dieser Arbeit konnte gezeigt werden, dass sich die Spamfilterung mittels Texterkennungsverfahren aufgrund der guten Erkennungsraten und aufgrund der Vorteile gegenüber anderen dynamischen Verfahren als Alternative zu Bayes'schen Filtern und zur Erkennung mittels Markov-Diskriminierung anbietet. Im Gegensatz zu den bisher bekannten dynamischen Verfahren wird der gesamte Inhalt der E-Mails zur Berechnung der Filterentscheidung herangezogen, so dass das Umgehen des Filters erschwert wird. Es konnte gezeigt werden, dass die Größe des verwendeten Wörterbuchs einen wesentlicher Faktor des Spamfilters darstellt, der sich sowohl auf die Entscheidungsrate des Filters als auch auf die Performanz auswirkt. Eine Verbesserung der Analyseergebnisse und eine Steigerung der Performanz kann weiterhin durch die Kombination mit anderen (statischen) Verfahren erreicht werden.

Teil I

Anhang

Abkürzungsverzeichnis

ASCII	American Standard Code for Information Interchange
CPAN	Comprehensive Perl Archive Network
DNS	Domain Name Service
E-Mail	Elektronische Nachricht
GPL	General Public License
HMM	Hidden Markov-Modell (verstecktes Markov-Modell)
HTML	Hypertext Markup Language
IMAP	Internet Message Access Protocol
KB	Kilobyte
LZ-Kompression	Lempel-Ziv Kompression
MIME	Multipurpose Internet Mail Extensions
MX	Mail Exchanger
POP	Post Office Protocol
PPM	Prediction by Partial Matching
S/MIME	Secure/Multipurpose Internet Mail Extension
SMS	Short Message Service
SMTP	Simple Mail Transfer Protocol
SSL	Secure Socet Layer
TEFT	Train-Everything
TLS	Transport Layer Security
TOE	Train-on-Error
TUE	Train-Until-Exhaustion
TUM	Train-Until-Mature
TUNE	Train-Until-No-Errors

Stichwortverzeichnis

A

Arithmetische Codierung 46

B

Bayes'scher Filter 25
Blacklists 23

C

Code 40
  Eindeutig Decodierbar 41
  Präfix-Code 41
  Unterscheidbarkeit 41

D

Datenkompression 40, 61
DNS 14

E

E-Mail
  Body 13
  Header 13
Entfernungsmatrix 53
Entropie 37, 62
  algorithmisch . . . . . *siehe* Kolmogorov-Komplexität
  bedingte 61, 63
  relative . . . . . *siehe* Kullback-Leibler-Divergenz
  von Informationsquellen . . . 38, 61
Ergodische Prozesse 36

F

falsch-negativ 26
falsch-positiv 26

G

Greylisting 23
gzip 49

H

Ham 19
Huffman-Codierung 46

I

IMAP . . . . . 14 f
Information 37
Informationsabstand 58

K

Kolmogorov-Komplexität 58
  bedingte 58
Kommunikationssystem 34
  (diskrete)Informationsquelle . . 34
  Empfänger 35
  Kanal 35
  Sender 35
  Ziel 35
Kompression
  dynamische Verfahren 43
  Hybride Techniken 43
  statische Verfahren 43
  Verlustbehaftet 42

Verlustfrei 42
Kompressionsrate 61
Korpus . . . . . 32, 51 ff
Kullback-Leibler-Divergenz . . . 38, 59

L

lazy matching 49, 79
Lernkurven 33
LZ77 47

M

Markov-Diskriminierung 25
Markov-Modell . . . . . 36, 44 f
der Ordnung D 45
verstecktes 36
Markov-Prozess *siehe* Markov-Modell
mbox . . . . . 14 f
MIME 16
MX 14

N

Nachricht 40

P

POP . . . . . 14 f
PPM 48

R

rar 50
Redundanz 38, 41

S

SMTP 14
SMTP-Envelope 12
Spam . . . . . 18 f
Kategorien 20
Primäre Merkmale 19
Sekundäre Merkmale 20
Spamfilter
mit Feedback 31
ohne Feedback 31
Simulation 32
Spamfilterung
Dynamische Verfahren 24
Statische Verfahren 22
Stochastischer Prozess 35

T

Texterkennung 26, 51
Klassifizierung von E-Mails . . . 53
Sequenzerkennung . . . . . 52 f
Sequenzklassifizierung 53
Training
TEFT 32
TOE 32
TUE 32
TUM 32
TUNE 32

V

Verteilte Erkennung 24

W

Whitelists 23
Wörterbuchverfahren
Dynamische Verfahren 44
Statische Verfahren 44

Z

Zeichencodierung 40
Zielfunktion 51

Literaturverzeichnis

[BA06] K. C. Barr and K. Asanovic. Energy Aware Lossless Data Compression. *ACM Trans. Comput. Syst.*, 24(3):250–291, 2006. http://doi.acm.org/10.1145/1151690.151692.

[BCL03] D. Benedetto, E. Caglioti, and V. Loretto. Zipping out relevant Information. *Computing in Science and Engg.*, 5(1):80–85, 2003. http://dx.doi.org/10.1109/MCISE.2003.1166556.

[BGL+98] C.H. Bennett, P. Gacs, M. Li, P.M.B. Vitanyi, and W.H. Zurek. Information Distance. *IEEE Transactions on Information Theory*, 44(4):1407–1423, July 1998. http://ieeexplore.ieee.org.

[BK90] A. B. and S. T. Klein. Compression, information theory, and grammars: a unified approach. *ACM Trans. Inf. Syst.*, 8(1):27–49, 1990. http://doi.acm.org/10.1145/78915.78917.

[BM02] R. Berlich and H. Milz. Mail-Filterung mit Procmail – Die Nadel im Heuhaufen. *Linux-Magazin*, 01, 2002. http://www.linux-magazin.de/Artikel/ausgabe/2002/01/procmail/procmail.html.

[BW02] C. Busch and S. D. Wolthusen. *Netzwerksicherheit.* Spektrum – Akademischer Verlag, 2002.

[cpa06] Cpan – comprehensive perl archive network, November 2006. http://www.cpan.org/.

[Cri03] M. Crispin. INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1. RFC 3501 (Proposed Standard), March 2003. http://www.ietf.org/rfc/rfc3501.txt.

[CW84] J. G. Cleary and I. H. Witten. Data Compression Using Adaptive Coding and Partial String Matching. *IEEE Transactions on Communications*, Com-32(4):396–402, April 1984. http://ieeexplore.ieee.org.

[Deu96] P. Deutsch. DEFLATE Compressed Data Format Specification version 1.3. RFC 1951 (Informational), May 1996. http://www.ietf.org/rfc/rfc1951.txt.

[DEV02] D. Benedetto, E. Caglioti, and V. Loretto. Language Trees and Zipping. *Physical Review Letters*, 88(4), January 2002.

[Eck04] C. Eckert. *IT-Sicherheit: Konzepte – Verfahren – Protokolle*. Oldenbourg, 3rd edition, 2004.

[EW05] P. Eisentrauth and A. Wirth. *Mit Open Source-Tools Spam und Viren bekaempfen*. O'Reilly, 1st edition, 2005.

[FB96a] N. Freed and N. Borenstein. Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies. RFC 2045 (Draft Standard), November 1996. http://www.ietf.org/rfc/rfc2045.txt.

[FB96b] N. Freed and N. Borenstein. Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types. RFC 2046 (Draft Standard), November 1996. http://www.ietf.org/rfc/rfc2046.txt.

[fWuT06] Bundesministerium fuer Wirtschaft und Technologie. e-f@acts – informationen zum e-business. 17, February 2006. http://www.bmwi.de/BMWi/Navigation/Service/Bestellservice/Publikationen/publikationen-efacts.html.

[GA06] J. Gailly and M. Adler. The gzip home page. October 2006. http://www.gzip.org/.

[GC06] Dr. J. Graham-Cumming. The spammers' compendium. Spam Conference, 2006. http://www.jgc.org/tsc/.

[Gra02] P. Graham. A plan for spam. August 2002. http://www.paulgraham.com/spam.html.

[Gra03] P. Graham. Better bayesian filering. January 2003. http://www.paulgraham.com/better.html.

[Hal05] E. Hall. The application/mbox Media Type. RFC 4155 (Informational), September 2005. http://www.ietf.org/rfc/rfc4155.txt.

[Kal04] A. Kaltchenko. Algorithms for Estimating Information Distance with Application to Bioinformatics and Linguistics. *ArXiv Computer Science e-prints*, April 2004. http://adsabs.harvard.edu/cgi-bin/nph-bib_query?bibcode=2004cs........4039K&db_key=PRE.

[Kle01] J. Klensin. Simple Mail Transfer Protocol. RFC 2821 (Proposed Standard), April 2001. http://www.ietf.org/rfc/rfc2821.txt.

[Lau06] O. Lau. Mailserver mit spamfilter. *c't-spezial Linux*, July 2006. http://www.heise.de/open/artikel/74991.

[Lev98] E. Levinson. The MIME Multipart/Related Content-type. RFC 2387 (Proposed Standard), August 1998. http://www.ietf.org/rfc/rfc2387.txt.

[LH87] D. A. Lelewer and D. S. Hirschberg. Data compression. *ACM Comput. Surv.*, 19(3):261–296, 1987. http://doi.acm.org/10.1145/45072.45074.

[LZ77] A. Lempel and J. Ziv. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, 23(3):337–343, 1977. citeseer.ist.psu.edu/ziv77universal.html.

[LZ78] A. Lempel and J. Ziv. Compression of individual sequences via variable-rate coding. *IEEE Transactions on Information Theory*, 24(5):530–536, 1978. http://citeseer.ist.psu.edu/ziv78compression.html.

[Moc87] P.V. Mockapetris. Domain names - concepts and facilities. RFC 1034 (Standard), November 1987. http://www.ietf.org/rfc/rfc1034.txt.

[Moo96] K. Moore. MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text. RFC 2047 (Draft Standard), November 1996. http://www.ietf.org/rfc/rfc2047.txt.

[MR96] J. Myers and M. Rose. Post Office Protocol - Version 3. RFC 1939 (Standard), May 1996. http://www.ietf.org/rfc/rfc1939.txt.

[OEC04] OECD. Background paper for the oecd workshop on spam. *The OECD Workshop On Spam*, 2004. http://www.olis.oecd.org/olis/2003doc.nsf/LinkTo/dsti-iccp(2003)10-final.

[Pal97] J. Palme. Common Internet Message Headers. RFC 2076 (Informational), February 1997. http://www.ietf.org/rfc/rfc2076.txt.

[PBC+03] A. Puglisi, D. Benedetto, E. Caglioti, V. Loreto, and A. Vulpiani. Data Compression and Learning in Time Sequences Analysis. *Physica D.*, January 2003.

[Ram99] B. Ramsdell. S/MIME Version 3 Message Specification. RFC 2633 (Proposed Standard), June 1999. http://www.ietf.org/rfc/rfc2633.txt.

[rar06] The official rar home page. November 2006. http://www.rarsoft.com/.

[Res01] P. Resnick. Internet Message Format. RFC 2822 (Proposed Standard), April 2001. http://www.ietf.org/rfc/rfc2822.txt.

[RKB06] P. Ratanaworabhan, J. Ke, and M. Burtscher. Fast lossless compression of scientific floating-point data. In *DCC '06: Proceedings of the Data Compression Conference (DCC'06)*, pages 133–142, Washington, DC, USA, 2006. IEEE Computer Society. http://dx.doi.org/10.1109/DCC.2006.35.

[RN95] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ, 1st edition, 1995.

[Ros06] A. Roshal. rar user guide. 2006. http://joy.debian.net/rar/Manual.html.

[SAP07] The apache spamassassin project. January 2007. http://spamassassin.apache.org/tests_3_1_x.html.

[Sch04] Dr. G. Schryen. Approaches addressing spam. *Proceedings of the IPSI*, 2004. http://winfor.rwth-aachen.de.

[Sch05] A. Schwartz. *SpamAssassin*. O'Reilly, 1st edition, 2005.

[Seb02] F. Sebastiani. Machine learning in automated text categorization. *ACM Comput. Surv.*, 34(1):1–47, 2002. http://doi.acm.org/10.1145/505282.505283.

[Sha53] C. E. Shannon. General treatment of the problem of coding. *IEEE Transactions on Information Theory*, 1(1), February 1953.

[Sha01] C. E. Shannon. A mathematical theory of communication. *SIGMOBILE Mob. Comput. Commun. Rev.*, 5(1):3–55, 2001. http://doi.acm.org/10.1145/584091.584093.

[Shk02] D. Shkarin. Ppm: One step to practicality. In *DCC '02: Proceedings of the Data Compression Conference (DCC '02)*, pages 202–211, Washington, DC, USA, 2002. IEEE Computer Society.

[Sop06] Sophos. Field guide to spam. October 2006. http://www.sophos.com/spaminfo/explained/fieldguide.html.

[Spa06] Monty python's flying circus – bbc-folgen. 25, November 2006. http://www.pythonsite.de/55.htm.

[Tan03] A. S. Tannenbaum. *Computer Networks*, volume 4. Prentice Hall, Inc., 2003.

[Tre06] 2005 trec public spam corpus. November 2006. http://plg.uwaterloo.ca/~gvcormac/treccorpus/.

[TSo07] The state of spam - a monthly report. January 2007. http://www.symantec.com/avcenter/reference/Symantec_Spam_Report_-_January_2007.pdf.

[Ver98] S. Verdu. Fifty Years of Shannon Theory. *IEEE Transactions on Information Theory*, 44(6), October 1998. http://ieeexplore.ieee.org.

[Wel84] T. A. Welch. A Technique for High-Performance Data Compression. *Computer*, 17(6):8–19, June 1984. http://ieeexplore.ieee.org.

[Win06] The official winrar home page. November 2006. www.winrar.com.

[WS02] For bulk e-mailer, pestering millions offers path to profit. *Wall-Street Journal*, November 2002. http://users1.wsj.com/lmda/do/checkLogin?mg=wsj-users1&url=http%3A%2F%2Fonline.wsj.com%2Farticle%2FSB1037138679220447148.html%3Femailf%3Dyes,.

[Yer05] W. S. Yerazunis. *The CRM114 Discriminator Revealed! or How I Learned to Stop Worrying and Love My Automatic Monitoring Systems*. 2005. http://crm114.sourceforge.net/CRM114_Revealed_20051207.pdf.

[Zdz05] J. A. Zdziarski. *Ending Spam – Bayesian content filtering and the art of statistical language classification*. No Starch Press, 1st edition, 2005.

Erklärung zur Diplomarbeit gemäß §19 Abs. 6 DPO/AT

Hiermit versichere ich, die vorliegende Diplomarbeit ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus den Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, 19. Februar 2007