Technische Universität Graz

Institut für Informationssysteme und Computer Medien

Diplomarbeit aus Telematik

Datenaspekte des deutschsprachigen Usenet

vorgelegt von

Martin Pirker

Begutachter:
o.Univ.-Prof. Dr. Dr.h.c. Hermann Maurer

Betreuer:
Univ.-Ass. Dipl.-Ing. Dr.techn. Harald Krottmaier

September 2005

Kurzfassung

Beständig dringt das Internet in den Alltag der Menschen vor, immer mehr Tätigkeiten werden per Computer und Netzwerk erledigt. Dies beschränkt sich nicht nur auf berufliche Anliegen, sondern beeinflußt auch immer mehr die Freizeitgestaltung.

Ein immer präsentes Bedürfnis der Menschen ist die Kommunikation mit anderen Menschen. Fortschritte in der Softwaretechnologie und Rechnerleistung ermöglichen im Internet experimentelle neue Formen des Informationsaustausches und der Diskussion. Die Verlockung ist groß, sich von Effekten einlullen zu lassen und die Lehren der Vergangenheit zu ignorieren.

Diese Diplomarbeit beschäftigt sich mit einem der ältesten Kommunikationsdienste des Internets, dem Usenet. Das Studium von realen Daten von 2 Jahren deutschsprachigem Usenet dient als Anschauungsbeispiel einiger Punkte, die sich bei der Umsetzung eines neuen Kommunikationsdienstes in großem Maßstab als Stolpersteine erweisen könnten.

Graz University of Technology

Institute for Information Systems and Computer Media

Master's Thesis in Telematics

Data Aspects of German Speaking Usenet

submitted by

Martin Pirker

Assessor:
o.Univ.-Prof. Dr. Dr.h.c. Hermann Maurer

Supervisor:
Univ.-Ass. Dipl.-Ing. Dr.techn. Harald Krottmaier

September 2005

Abstract

Progressivly the internet pervades the daily life of the population, more and more things can be done by a networked computer. This is not just limited to work issues, but also affects the way people spend their free time.

There is an ever present drive in people to communicate with others. Advances in software technology and processing power enable new experimental internet information exchange and chat services. It is all too easy to be blinded by the flashy new world, ignoring the lessons of the past.

This masters's thesis examines one of the oldest internet communication services, the Usenet. The study of 2 years of real data of the German speaking Usenet serves as an illustration of some issues arising, which may prove to be tripping stones on the implementation of a large scale communication service.

Ich versichere hiermit, diese Arbeit selbstständig verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und mich auch sonst keiner unerlaubten Hilfsmittel bedient zu haben.

Unterschrift Autor:

Inhaltsverzeichnis

1 Einführung

„Die Welt ist ein Dorf“

Ein einfacher Satz.

Unklar, wann man ihn das erste Mal gehört hat, unklar, wer diese Erkenntnis zuallererst formuliert hat, doch als heutigem Angehöriger einer Industrienation wird einem täglich vorgeführt, wie klein die Welt durch die Fortschritte in der Kommunikationstechnik geworden ist.

Für den modernen Menschen des 21. Jahrhunderts ist die Computertechnik ein fixer Bestandteil des Alltages geworden. Das „Netz der Netze“, das Internet, hat eine Kommunikationsdynamik ermöglicht, von der man vor wenigen Jahren noch nur träumen konnte.

Täglich werden Millionen, wenn nicht Milliarden Botschaften versandt – E-Mail – der Nachrichtenaustausch für jedermann mit jedermann. Einfach, billig, schnell, zielgerichtet.

Sucht man Informationen über eine Firma oder deren Produkte, sieht man auf ihrer „Homepage“ nach, dem neuen virtuellen Repräsentationspfeiler gegenüber potentiellen Kunden. Für Firmen quasi schon Pflicht um im Geschäft zu bleiben, ist eine Homepage für Privatpersonen jedoch öfters wohl eher ein Experiment der Selbstdarstellung – vor einem weltweiten Publikum.

„Wo chattest du?“. Chat, eine einfache Netzverbindung als Bindeglied, ein paar Zeichen wandern in eine Richtung, ein paar Zeichen in die andere. Gedanken, Emotionen, Ereignisse, . . . in ein paar Satzfetzen gepresst, selten schöner Schriftsprache folgend.

Für viele Teilnehmer besteht das Internet aus „allem was in meinem Internetbrowser passiert“. Gibt es noch andere interessante Kommunikationsdienste? Ja.

Diese Diplomarbeit beschäftigt sich mit den Inhalten eines Dienstes des Internets, der zwar auf mehr als 2 Jahrzehnte Existenz zurückblicken kann, aber heutzutage durch moderne Attraktionen im Begriff ist, in den Hintergrund gedrängt und vergessen zu werden, dem „Usenet“ .

1.1 Motivation

An der TU Graz erfüllt der zentrale „Newsserver“[1] – quasi ein Usenet im Kleinen – bis heute eine wichtige Rolle als Informations- und Kommunikationssystem. Viele Teilnehmer nehmen ihn nur als „selbstverständlichen“ Dienst wahr. Der tägliche Umgang: Man interagiert mit wenigen Gruppen, die dem eigenen Interesse entsprechen, man liest immer nur die aktuellen Artikel auf dem Server – und irgendwann schließt man die Verbindung und widmet sich Wichtigerem.

Für einen Studenten einer Informatik orientierten Studienrichtung – Telematik – ist jedoch die praktische Durchführung, die Technik hinter den Kulissen, die einen reibungslosen Kommunikationsablauf ermöglicht, von Interesse.

Für jedes Informationssystem muß in mehreren Punkten eine Designentscheidung getroffen werden. Wie organisiert man den Prozeß des Datenaustausches mehrerer Datenbestände? Welche Strategien bringen eine gute Informationsverteilung? Wie speichert man die Daten und welche Codierungen werden verwendet? Lassen sich Metainformationen – Informationen über die Informationen im System selbst – gewinnen?

Im Jahre 2002 unternahm der Autor dieser Diplomarbeit selbstständig erste Experimente mit der Archivierung von Usenet Artikeln, inspiriert durch parallel laufende Bemühungen des TU-Graz Newsserver Archivierungsprojektes[2]. Die Entwicklung der Rechnerleistungen in letzter Zeit ermöglichten eine Verarbeitung von zig Gigabyte großen Datenmengen auf einem Standard PC und somit ein Archivierungsprojekt über den Maßstab des lokalen Servers hinausgehend. Für diese Diplomarbeit wurden über einen Zeitraum von 2 Jahren alle deutschsprachigen Usenet-Artikel gespeichert und danach nach verschiedenen Kriterien untersucht.

Dem Leser dieser Diplomarbeit werden hoffentlich, so wie dem Autor dieser Arbeit, mehrere „Aha!“ Momente passieren. Das Studium eines „uralten“ Internetdienstes, mit dessen Limitationen, Problemen und Designentscheidungen sollte ein Lernbeispiel für die Visionäre und Implementoren zukünftiger Kommunikationsdienste sein – vielleicht sogar mit Beteiligung von Telematikern.

1.2 Kapitelübersicht

Als Anfang bietet Kapitel 2 eine kurze allgemeine Einführung in das Themengebiet Usenet und offeriert einen Grundstock historischer Entwicklungen und Anforderungen, eine Diskussion der Merkmale des Systems und eine Begriffsbildung für das Verständnis der folgenden Kapitel.

[1] `http://www.zid.tugraz.at/ki/internet/news/tug.html`
[2] `http://newsarchiv.tugraz.at/`

Kapitel 3 beschäftigt sich mit dem Datenfluß im Usenet System. Es wird auf die (Protokoll-)Kommunikation mit Newsservern eingegangen. Der Weg eines Artikels von seiner Entstehung zu seiner Verbreitung, bis hin zu möglicher Wegen das System wieder zu verlassen, nachvollzogen. Weiters wird die Logistik von lokaler Sammlung großer Artikelmengen beleuchtet.

Kapitel 4 nimmt Maß am gesammelten Datenbestand. Im ersten Abschnitt wird eine Quantifizierung des Datenumsatzes vorgenommen. Der zweite Abschnitt des Kapitels beschäftigt sich mit Eigenheiten der Steuerinformationen der Artikel, wenn man die Chance hat, die Qualität dieser einmal über eine große Datenmenge hinweg zu bestimmen.

In Kapitel 5 wird versucht, anhand einzelner Aspekte durch bewußte Verknüpfung lokaler und globaler Datenmerkmale neue Datenpunkte zu extrahieren. Dies ermöglicht die Konstruktion von Gruppenprofilen, mit Hilfe der Abhängigkeitsstrukturen eine Zitatdetektion, oder aber auch Auffindung von Artikelflutungen und weiteren Anwendungen.

Kapitel 6 bietet eine abschließende Diskussion der positiven und negativen Erkenntnisse und Spekulationen über etwaige zukünftige Entwicklungen.

So nicht explizit darauf verwiesen wird, sind sämtliche im Text genannten Daten mit Hilfe von selbsterstellten Programmen und Implementierungen der angeführten Algorithmen aus dem Datenarchiv extrahiert worden. Anhang A bietet eine Übersicht und Diskussion der praktischen Durchführung dieser Diplomarbeit. Sämtliche Entwicklungen sind auf einer CD dieser Diplomarbeit beigefügt.

Anhang B bietet noch ergänzende Graphen und Tabellen, die für manche Details noch von Interesse sind.

Textkonventionen

Fachbegriffe, die erstmals im Text verwendet werden, sind anfangs mittels „xxx“ hervorgehoben. Nach mehrmaliger Verwendung wird auf eine ständige Hervorhebung im weiteren Verlauf jedoch verzichtet.

Werden im Text (Fach-)Begriffe aus Spezifikationen, Protokollen oder (selbst generierten) Daten verwendet, werden diese in `Schreibmaschinenschrift` ausgezeichnet.

Links

Der Autor dieser Arbeit hat keinerlei Kontrolle über den Inhalt der im Text verlinkten Websites von Dritten und übernimmt daher keinerlei Haftung für deren Inhalt.

2 Usenet

2.1 Geschichte

In den Anfangsjahren der Computertechnik waren die ersten Rechner noch ausschließlich Spielzeug jener, die sie sich schlichtweg leisten konnten. Großunternehmen, Militär oder Universitäten erwarteten sich Fortschritte bei der Lösung ihrer Probleme, die mit viel Rechenaufwand verbunden waren. Mit dem kontinuierlichen technischen Fortschritt der Informationstechnologien erkannte man bald, dass ein Computer alleine produktives leisten kann. Vernetzt man jedoch mehrere befreundete Einrichtungen, konnte man unter Umständen große Probleme aufteilen und/oder gemeinsam lösen.

Der reine Austausch von Forschungsdaten nutzte aber nur einen Teil des Potentials der entstehenden Rechnernetzwerke. Inspiriert durch den „Nachricht des Tages“[1] Dienst damaliger Unix Systeme – bei jedem neuen Einstieg sah man eben solch eine Nachricht – begann man sich Gedanken zum automatischen Austausch von Nachrichten netzwerkweit zu machen.

Um ca. 1980 entstand die erste primitive Software zu diesem Zweck, [A News], eine einfache Möglichkeit mehrere Ankündingungen pro Tag innerhalb eines Verbundes von einer Handvoll Rechner zu verbreiten.

Eine Nachricht hatte einen Rechnernamen als Absender, eine direkte Antwortmöglichkeit war nicht vorgesehen und alle Nachrichten landeten bei einer einzigen lokalen Sammelstelle. Etwaige Zusammenhänge zwischen mehreren Nachrichten wurden nicht dokumentiert. Entweder man las die neuen Nachrichten, streng eine nach der anderen, oder man verschickte eine Neue. Zusammengefasst, handelt es sich bei „A News“ um eine Minimalkonstruktion und aus heutiger Sicht war das Prinzip sehr primitiv, aber als erster Versuch war es „gut genug“ und ein Anfang war getan.

Wie bei vielen anderen Netzwerkdiensten „passierte“ dem Newsdienst eine kontinuierliche Evolution. Im Laufe der Zeit wanderte der Sourcecode durch viele Hände und Stück für Stück wurden neue Fähigkeiten hinzugefügt – die gleichzeitig fortschreitende Rechnervernetzung tat ihr übriges.

[1] Auch bekannt unter MOTD – message of the day

Aus „A News“ wurde [B News], schließlich [C News] und ca. 1992 „InterNetNews“[2] (INN), jenes Programm, welches in einer modernen Variante heute noch auf vielen Servern (auch auf dem der TU-Graz) für den „News“-Dienst verantwortlich ist.

2.2 „Usenet“ oder „News“?

Die beiden Begriffe sind manchmal austauschbar und ihre Benutzung hängt mehr vom aktuellen Kontext und Wissensstand der jeweiligen Personen ab.

Das System wurde als „Usenet“ bezeichnet, in Anlehnung an die USENIX[3] Organisation, in der Hoffnung, diese Organisation nähme eine aktivere Rolle in der zukünftigen Entwicklung ein.

Die Bezeichnung „News“ entstand auf Grund der Ursprünge in der Neuigkeitenverbreitung. Der jeweils zuständige Server für den Dienst wird meist als „Newsserver“ bezeichnet, was sich auch meist im Internetrechnernamen wiederspiegelt, zum Beispiel `news.tugraz.at`.

Die Gesamtheit des weltumspannenden Verbundes wird fast immer als „Usenet“ bezeichnet. Lokal und unter Freunden wird meist der Begriff „News“ verwendet, wenn aus dem Kontext keine Verwechslungsgefahr mit anderen Neuigkeiten- bzw. Nachrichtendiensten besteht.

2.3 Artikel

Wie bereits im vorhergehenden Kapitel angedeutet, wurde Usenet als Neuigkeiten- bzw. Nachrichtenaustauschsystem entworfen. Mit der Zeit entwickelte es sich aber zum Diskussionsmedium. Die Beiträge bezeichnet man als „Artikel“ oder „Postings“.

Was sind die wesentlichen Merkmale eines Artikels?

- Autor

  Der Absender des Artikels. Diese Information besteht üblicherweise aus zwei Teilen, dem Namen des Absenders und einer E-mail Adresse als direkte Kontaktmöglichkeit.

[2] Heute ist INN downloadbar von `http://www.isc.org/`, der Seite von Internet Systems Consortium, Inc., die es sich zur Aufgabe gemacht hat, Internet Infrakstruktur Bausteine in Referenzimplementierungen zur Verfügung zu stellen. Der Fortschritt der Einführung von INN wird in [Salz92] diskutiert.

[3] Gegründet 1975 als „Unix Users Group“, später umbenannt zu Usenix. Bis heute Organisator von Konferenzen in den Gebieten Unix, System Administration, Betriebssystemforschung etc.

- Titel

  Ein Artikeltitel sollte in wenigen Worten knapp und informativ den wesentlichen Inhalt eines Artikels zusammenfassen.

- Datum

  Nichts hat ein schnelleres Verfallsdatum als eine Information. Ein „Zeitstempel“ ist manchmal ein wichtiges Instrument zur groben Abschätzung des Informationsgehaltes eines Artikels.

- Themengebiet

  Eine grobe Einteilung des Zielpublikums eines Artikels. Jemand der den Sportteil einer Zeitung liest, wird nicht unbedingt Interesse am Wirtschaftsteil haben. Den einen Leser interessieren mehr die Fußballergebnisse vom Sonntag, den anderen mehr der aktuelle Aktienkurs seiner letzten Investition.

- Referenzen

  Falls sich ein Artikel auf einen vorhergehenden bezieht, sei es um diesen abzulösen, zu ergänzen, oder um in einer Diskussion den „Faden“ nicht zu verlieren, ist es wichtig, Zusammenhänge mit älteren Nachrichten zu vermerken.

Sämtliche dieser Merkmale (und weitere) sind im modernen Usenet System implementiert worden. In einem Usenet Artikel befinden sich diese speziellen Informationen üblicherweise in einem Artikelvorspann – („head“ bzw. „header“) – eincodiert. Ein Artikel könnte etwa so beginnen:

- Autor
  `From: Martin Pirker <crf@sbox.tugraz.at>`
- Titel
  `Subject: Einführung in die Funktionsweise des Usenet`
- Datum
  `Date: Fri, 18 Mar 2005 12:20:12 +0200`
- Themengebiet
  `Newsgroups: tu-graz.diverses`

Jeder Artikel wird auch mit einer Kennzeichnung versehen. Diese hat die spezielle Eigenschaft, weltweit eindeutig zu sein, um die automatische Verarbeitung zu ermöglichen. Referenzen sind einfach Verweise auf Kennzeichnungen vorhergehender Artikel und ermöglichen eine Rekonstruktion eines Diskussionsverlaufes („Thread“).

- Kennzeichnung
  `Message-ID: <d5qjjsk$po$1@rechername.irgendwo.at>`
- Referenzen
  `References: <hfg8kdf$1@woanders.dort.at>`

2.4 Gruppen und Hierachien

Jedes Themengebiet wird in einer bestimmten Gruppe (oder „Newsgroup“) diskutiert. Diese Gruppen sollten mit einer möglichst eindeutigen Bezeichnung als Gruppennamen versehen sein, um die Leser und Schreiber immer leicht zu „ihrer“ Gruppe zu lenken. Kann man mehrere Gruppen zu einem größeren Gebiet zusammenfassen, wird eine gleiche Kennung vorangestellt und man bezeichnet diesen Gruppenverbund als Hierachie.

Lokal orientierte Gruppen, wie zum Beispiel die der `tu-graz.*` Hierachie[4], werden üblicherweise nur auf einem bestimmten lokalen Server angeboten. Internationale Gruppen erfahren eine (bis zu) weltweite Verbreitung. Ein lokaler Server bietet üblicherweise nur einen Auszug der global verfügbaren Gruppen, den Vorlieben der Kundschaft und der Ausrichtung des Providers entsprechend.

An weltweiten Hierachien sind die bedeutensten 9 größten[5]:

- `alt.*`
  Die „alternative“ Hierachie mit kaum vorhandenen Regulierungen, quasi der „Spielplatz“ im Usenet.
- `comp.*`
  Diskussionen über alles was mit Computer zu tun hat.
- `misc.*`
  Vermischtes oder was sonst nirgends so richtig dazupasst.
- `news.*`
  Themen rund um News (dem Medium News, nicht Neuigkeiten)
- `rec.*`
  Unterhaltung und Freizeitthemen.

[4] Eine Überblicksliste mit Kurzbeschreibungen findet sich unter
`http://www.zid.tugraz.at/ki/internet/news/tu-graz.html`

[5] Da die „wild“ gewachsenen Einteilungen immer schwerer zu administrieren waren, entschloß man sich 1987 zu einem scharfen Einschnitt, „The Great Renaming“, aus dem das bis heute nahezu unveränderte Layout der weltweiten Unterteilungen hervorging.
siehe [Great Renaming] und [Spaf86]

- `sci.*`
  Wissenschaftliche und Spezialwissen Themen.
- `soc.*`
  Gesellschaftliche Themen.
- `talk.*`
  „Stammtisch“gespräche über kontroversielle Themen
- `humanities.*`
  Geistige Themen: Kunst, Literatur, Philosphie, ...

In internationalen Hierachien wird üblicherweise in Englisch kommuniziert. Als Zwischenstufe zwischen lokalen und weltweiten gibt es noch die Länderhierachien, in denen üblicherweise in der jeweiligen Landessprache kommuniziert wird:

- `de.*`
  Deutschsprachige Hierachie.[6]
- `at.*`
  Themen mit Österreich Bezug.[7]
- `fr.*, it.*, ...`
  Französisches, Italienische Themen, ...

Diese Diplomarbeit beschäftigt sich primär mit dem Datenbestand der `at.*` und `de.*` Hierachien, in denen Deutsch die übliche Verkehrssprache darstellt.

2.5 Nutzung

Viele Internetprovider bieten einen lokalen Newsserver `news.providername.at` als Einstiegspunkt in das Usenet an. Sollte man keinen lokalen Server zur Verfügung haben, oder bewegt man sich viel durch verschiedene Netze, bietet es sich an, einen Zugang bei einem spezialisierten Provider[8] für geringes Entgelt zu abonnieren.

Als Software benötigt man einen „Newsreader“. Diesen gibt es für jedes Betriebsystem und er stellt quasi das Navigationswerkzeug für den menschlichen Benutzer im Datensee des Newsservers dar. Die Varianten reichen von einer relativ schlichten und funktionellen Erscheinung (Abbildung 2.1) bis hin zum „Kommunikationszentrum“, das neben News auch E-Mail und andere Dienste vereint.

[6] `http://www.dana.de/mod/`
[7] `http://www.usenet.at/`
[8] Im deutschsprachigen Raum ist beispielsweise der Dienst von
`http://news.individual.de/`
einer der bekanntesten Einstiegspunkte.

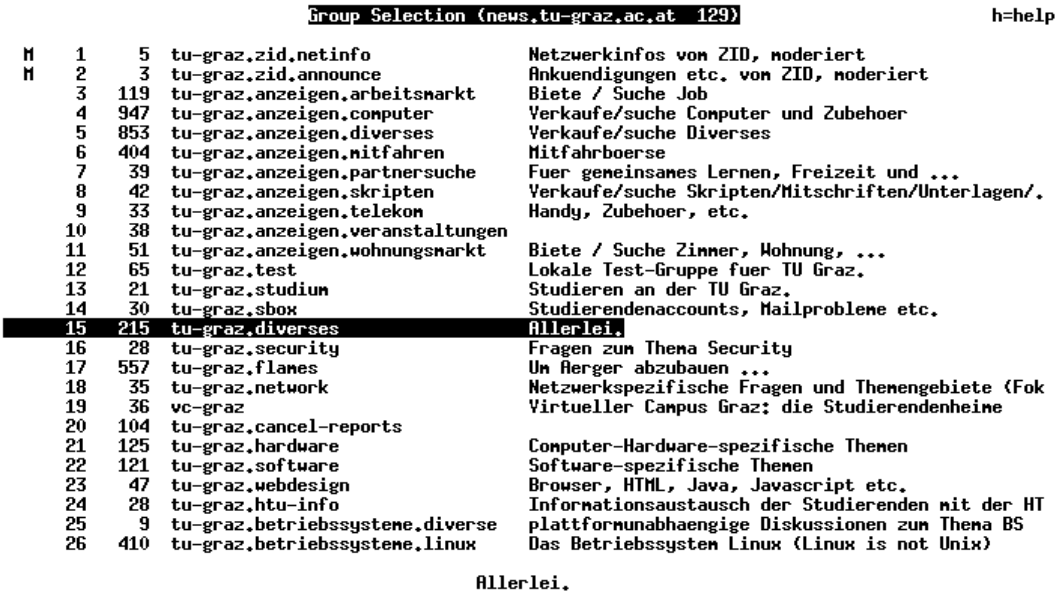

Abbildung 2.1: Ein schlichter Newsreaders (Beispiel Gruppenübersicht)

Nach der Erstinstallation konfiguriert man seine persönlichen Zugangsdaten – ein ähnlicher Vorgang wie bei einem E-Mail Programm – um danach eine Verbindung zum Newsserver aufzubauen. Man kann den eigenen Vorlieben entsprechend interessante Newsgruppen abonnieren, neue Artikel in den gewählten Themengebieten vom Server holen, auf bereits bestehende Artikel antworten oder mit einem Artikel eine gänzlich neue Diskussion starten.

3 Daten

Ein wesentliches Konstruktionsmerkmal eines Informationssystems ist der Umgang mit dem ihm anvertrauten Daten. Dieses Kapitel beschäftigt sich mit dem Artikelkreislauf im Usenet, von der Einspeisung über die Verbreitung bis hin zur „Endlagerung“ von Artikeln und den dabei auftretenden logistischen Herausforderungen.

3.1 Kommunikation mit dem Newsserver

Obwohl Usenet seine ersten Datenaustausche per UUCP[1] Protokoll unternahm, besteht das heutige Usenet, also der globale Verbund der Newsserver, fast nur mehr aus per Internet verbundenen Rechnern. Diese Rechner sprechen, sowohl Server untereinander als auch Newsreader gegenüber Servern, das NNTP[2] Protokoll.

Der NNTP Standard wurde vor ca. 20 Jahren in [RFC977] formal niedergeschrieben und beschreibt einen schlanken Befehlssatz, der alle Funktionen für mögliche Datenanfragen und -transfers abdeckt. Im Laufe der Zeit implementierten populäre Newsserver und -clients eigene Erweiterungen zum NNTP Protokoll, da die ursprünglichen Befehle mit dem wachsendem Datenaufkommen nicht immer die gewünschte Effizienz ermöglichten. Die „üblichen“ Erweiterungen wurden vor ca. 5 Jahren in [RFC2980] zusammenfassend dokumentiert.

Heute gibt es noch laufende Standardisierungsarbeitsgruppen, siehe [nntpext], um das NNTP Protokoll für die Anforderungen des Internets im 21. Jahrhundert zu adaptieren. Dazu gehören unter anderem Erweiterungen in den Bereichen Authentifikation, verschlüsselte Datenverbindungen zum Server, optimierter Artikelaustausch und Anpassungen für erweiterte Zeichensatzbereiche (in Protokoll und Artikeldaten).

Die Verbindung mit einem Newsserver ist vom Typ TCP. Der Server bietet seine Dienste üblicherweise auf Portnummer 119 an. Das Protokoll ist ASCII basierend, Befehle und Statusmeldungen werden zeilenweise hin und zurück übertragen.

Eine Verbindung zu einem Newsserver kann durch Benutzung eines `telnet` Programmes einfach simuliert werden:

[1] UUCP = Unix to Unix Copy Protocol
`http://en.wikipedia.org/wiki/Uucp`
[2] NNTP = Network News Transport Protocol

```
$ telnet news.tugraz.at 119
Trying 129.27.3.20...
Connected to fstgss00.tu-graz.ac.at.
Escape character is '^]'.

200 news.tugraz.at InterNetNews NNRP server INN 2.4.1 ready (posting ok).
```

Der Vorteil von den (meist) ASCII basierenden Internetprotokollen sind die leicht nachvollziehbaren Kommunikationsabläufe. Ein `HELP` Kommando, an einen Newsserver geschickt, offeriert alle angebotenen Funktionen – für eine detaillierte Erklärung jedes einzelnen Befehls sei auf die Standarddokumente im Literaturverzeichnis verwiesen.

```
HELP
100 Legal commands
  authinfo user Name|pass Password|generic <prog> <args>
  article [MessageID|Number]
  body [MessageID|Number]
  date
  group newsgroup
  head [MessageID|Number]
  help
  ihave MessageID
  last
  list
[active|active.times|extensions|newsgroups|distributions|distrib.pats|
overview.fmt|subscriptions|motd]
  listgroup newsgroup
  mode reader
  newgroups [YY]yymmdd hhmmss ["GMT"]
  newnews newsgroups [YY]yymmdd hhmmss ["GMT"]
  next
  post
  slave
  stat [MessageID|Number]
  xgtitle [group_pattern]
  xhdr header [range|MessageID]
  xover [range]
  xpat header range|MessageID pat [morepat...]
  xpath MessageID
Report problems to <usenet@fstgss00.tu-graz.ac.at>
.
```

3.2 Lebenslauf eines Artikels

Der Lebensweg eines Artikels beginnt mit der Erschaffung eines neues Textes durch einen Netzteilnehmer. Vom lokalen PC aus wird dieser auf einen Newsserver transferiert, wird dort Teil des Artikelumlaufes und somit auf weitere Server verbreitet. Nach Erfüllung von Terminierungskriterien, wie zum Beispiel maximaler Haltezeit, wird der Artikel von der Serversoftware wieder aus dem Kreislauf Kopie für Kopie entfernt. Die einzelnen Stationen im Detail:

3.2.1 Einspeisung

Ein Benutzer wählt in seinem Newsreader eine Gruppe aus, für die er einen neuen Artikel schreiben möchte. Nach Fertigstellung des Artikeltextes („article body“), fügt der Newsreader dem Artikel einen Kopf („article head“) mit den interessanten Kenndaten (siehe Kapitel 2.3) und weiteren Steuerdaten bei.

Die Übergabe an einen Newsserver erfolgt von der Newsreader Software mit dem NNTP `POST` Befehl, worauf der Newsserver eine eindeutige Artikelkennzeichnung vorschlägt:

```
POST
340 Ok, recommended ID <d89r35$qnu$1@fstgss00.tu-graz.ac.at>
```

Wenn man Schreibrechte auf dem Newssserver besitzt, übernimmt dieser den Artikel, überprüft diesen ob er dem Newsartikel Standardformat entspricht, weist dem Artikel eine weltweit eindeutige `Message-ID` zu (sofern dieser noch keine `Message-ID` besitzt) und speichert ihn in seiner Datenbank, sofort verfügbar für potentielle Leser.

Manche Gruppen besitzen einen Moderationsstatus, der Artikel wird zuerst in einem Zwischenspeicher abgelegt und ein Moderator – entweder menschlich oder ein Bot[3] – muß diesen erst explizit freigeben, um ihn für alle Leser abrufbar zu machen.

3.2.2 Verbreitung

Bei internationalen Gruppen tauschen „befreundete“[4] Newsserver regelmäßig ihre neuen Artikel aus. Man nennt diese Beziehung einen „Newsfeed“. Server, die sich vertrauen, „füttern“ sich gegenseitig neue Artikel.

[3] Bot = Kurzform von Ro*bot*er
Ein Programm, welches darauf spezialisiert ist, automatisch weitere Operationen vorzunehmen. Zum Beispiel die Überprüfung des Artikels auf komplexere Kriterien: ASCII versus HTML Text, Kontrolle beigefügter Daten auf „unerlaubte“ Inhalt, etc.

[4] Eine wartungsarme und schnelle Überprüfung funktioniert zum Beispiel per IP Adresse.

Technisch gesehen verbindet sich ein Server mit seinem Freund und bietet ihm per `IHAVE <Message-ID>` seine neuen Artikel an. Dieser kann das Angebot annehmen, weil er diesen Artikel laut seiner `Message-ID` noch nie gesehen hat, oder ablehnen, weil dieser Artikel ihm eventuell schon von einem anderen Newsserver geliefert wurde.

Man beachte den Unterschied zwischen `POST` und `IHAVE`:

- `IHAVE` ist ausschließlich für die Serverkommunikation vorgesehen, weil
- bei `IHAVE` führt der Newsserver keine strikten Überprüfungen des Inhaltes durch, der Artikel wird unmodifiziert übernommen. Die Annahme ist, dies wurde bereits bei der allerersten Artikelannahme gemacht und ist somit Rechnerzeitverschwendung. Andererseits ermöglicht dieses Vertrauensverhältnis ein beliebiges „unterjubeln“ von Artikeln mit „kreativem“ – mitunter bösartigem – Inhalt.

Verteilungsstrategie als Merkmal

Diese Art des Artikelaustausches ist eine signifikante Eigenschaft des Usenet Systems. Es gibt keine zentrale Institution, die die Vernetzung der teilnehmenden Rechner und den Artikelaustausch koordiniert. Es ist eine Art P2P[5] Vernetzung, mit dem speziellen Detail, dass der Urheber die Verteilung des neuen Datenpaketes anstößt („push“ Architektur), und nicht eine Abfrage nach neuen Daten eine Verteilung bewirkt.

Das Funktionieren des Datenaustausches beruht auf dem kooperativen Verhalten der Beteiligten. Jeder ist Herr über den eigenen Server, aber das gemeinsame Interesse an neuen Artikeln führt zu einem regen Austausch.

Andererseits hat dieses System auch zur Folge, dass jeder Server seine eigene Sicht des Usenets besitzt. Es gibt keinen vollständigen Newsfeed in dem garantiert keine Artikel fehlen. Ein temporärer Netzausfall verhindert unter Umständen an einer wichtigen Verbindung die Verbreitung von ein paar Artikeln und ein schlecht programmierter Newsserver vergißt dann diese bei wiederhergestellter Verbindung erneut anzubieten.

Im positiven Sinne verleiht diese Form des Datenaustausches dem Netz aber auch eine Form der Robustheit. Ist die Vermaschung der Rechner dicht genug, findet ein Artikel früher oder später seinen Weg auf jeden angebundenen Server. Wird punktuell versucht Spam einzuschleusen, wird anhand des Laufpfades des Artikels schnell der Einspeisepunkt festgestellt und der Rechner koordiniert abgekoppelt beziehungsweise der Spam in der Nähe bekämpft, so dass der Rest des Usenets nichts mehr davon mitbekommt.

[5] P2P = peer-to-peer
Eine Vernetzung von Rechnern die untereinander quasi gleichgestellt sind, die zentralen Verbindungskomponenten verlieren an Bedeutung und die Teilnehmerendpunkte sind selbst die wichtigen Datenträger.

3.2.3 Entfernung

Die Lebensdauer eines Artikels auf einem Newsserver ist begrenzt. Kein Server besitzt unendliche Speicherkapazität um alle jemals empfangenen Artikel aktiv verfügbar zu halten. Für jede Hierachie wird eine `expire` Zeit festgelegt, eine maximale Haltezeit. In Zeiten schwacher Auslastung, also üblicherweise in den Nachtstunden, durchsucht der Newsserver seine Artikeldatenbank und entfernt veraltete Artikel aus dem aktiven Angebot.

Es gibt auch die Möglichtkeit, einen Artikel vorzeitig durch einen expliziten Löschbefehl vom Server zu entfernen, durch

- eine `cancel` Kontrollnachricht, die eine Löschung bewirkt, oder durch
- einen `supersede` gekennzeichneten Artikel, der einen bestimmten Artikel durch einen neueren ersetzt.

Beide besitzen das Problem des Missbrauchs, da das Usenetsystem derzeit keine standardisierte Möglichkeit zur eindeutigen Bestimmung des Urhebers eines Artikels kennt. Das bedeutet, Personen können sich unter Umständen als andere ausgeben und „unliebsame“ Meinungen in anderen Artikeln löschen beziehungsweise verändern.

Dem entgegen wirkt die verteilte Konstruktion des Usenets, unübliche Löschbefehle bleiben anhand der Artikelunterschiede zwischen populären Servern nicht lange unentdeckt. Manche Administratoren gehen aber auch soweit, die Löschfunktion ob ihres Mißbrauchspotentials auf ihrem Server gänzlich zu deaktivieren, was aber wieder Nachteile bei der Spambekämpfung birgt.

3.2.4 Artikel leben theoretisch ewig

Ein Nebenprodukt digitaler Speicherung von Daten ist deren theoretische Ewigkeit. Jahr für Jahr wird digitale Speicherkapazität immer billiger und dieser Trend wird auch noch die nächsten Jahre anhalten.

Verglichen mit heutigen Speichergrößen sind Usenet Textartikel verschwindend klein. Irgendjemand irgendwo auf der Welt archiviert alle Usenet Artikel[6] und irgendwann in der Zukunft werden sie vielleicht auch wieder veröffentlicht. Jede Information, die im Usenet veröffentlicht wird, ist Teil der Internetgeschichte.

[6] Der bekannteste Dienst ist derzeit Google Groups: `http://groups.google.com/`
Google offeriert zwar gratis Zugang zu den Usenet Archiven, die Netzgemeinschaft ist jedoch kritisch, dass *die* größte Sammlung des Usenets in privater Hand ist, kein öffentlicher Plan für die Zukunft besteht und niemand Zugriff auf die Rohdaten hat.

Es bietet sich die Möglichkeit, im Artikelkopf die Zeile
`X-No-Archive: yes`
einzufügen, theoretisch ist [X-No-Archive] ein Standard, dass dieser Artikel nicht archiviert werden sollte.

Praktisch muß sich niemand daran halten und Archive speichern meist intern trotzdem alle Artikel ab, da sonst Abhängigkeiten in Diskussionsverläufen schwerer aufgelöst werden können. Auch im Rahmen der Datensammlung dieser Diplomarbeit wurde auf die Beachtung dieser Kennzeichnung verzichtet und jeder Artikel archiviert.

Ob es Politik eines Archives sein soll, alle gekennzeichneten Artikel zu unterdrücken, oder dies zu ignorieren, da ein fehlender Artikel oft aus Vorgänger und Nachfolgeartikel eine Diskussion weitgehend rekonstruieren kann, wird wohl weiterhin für Diskussionen zwischen Datenschützern und Archivaren von Internetinhalten sorgen.

3.3 Artikelabgleich

Im vorherigen Kapitel wurde der Nachrichtentransfer aus Serversicht erläutert. Betreibt man keinen eigenen Server – was wohl den Regelfall darstellt – bekommt man natürlich die gewünschten Artikel nicht automatisch angeboten, sondern versucht als Bittsteller – also protokolltechnisch gesehen als Newsreader – sie möglichst vollständig zusammenzusuchen.

Abhängig von den Prioritäten

- Serverlast
- Leitungslast
- Vollständigkeit

kann man verschiedene Methoden zum Artikelabgleich wählen. Nachfolgend werden mehrere mögliche Methoden des Artikelabgleichs, sowie die Erfahrungen aus deren praktischer Umsetzung diskutiert.

3.3.1 „Lazy“ Synchronisation mittels NEWNEWS

Der einfachste Weg für einen Newsclient um festzustellen, ob auf einem Newsserver neue Artikel verfügbar sind, ist der Befehl `NEWNEWS`:

```
NEWNEWS newsgroups [YY]yymmdd hhmmss ["GMT"]
```

`NEWNEWS` benötigt als Parameter eine Newsgruppenspezifikation und einen Zeitstempel. Als Ergebnis liefert der Server eine Liste von `Message-ID`s

<Message-ID>
<Message-ID>
<Message-ID>
<Message-ID>
...

aller neuen Artikel ab diesem Zeitpunkt.

Vorteilhaft ist der einfache Aufbau dieses Befehls und bei Anwendung mit „*“ als Gruppenspezifikation die vollständige Lieferung aller neuen Artikel am Server.

Nachteile gibt es mehrere:

- Es ist unbekannt, wieviele Artikel der Server als Antwort liefert.
- Die Antwortliste entspricht nicht der zeitlichen Abfolge.
- Die Serverlast, um alle Artikel auf großen Servern „zusammenzusuchen“, ist nicht zu unterschätzen.
- Man bekommt keine Informationen über Artikellöschungen.

Praktische Erfahrungen

Es wurde zu Versuchszwecken eine Art „Newsticker“ für den TU Newsserver implementiert, ein Programm welches nur zeilenweise „Zeit - Gruppe - Subject - From“ neuer Artikel auflistet - siehe Abbildung 3.1. Es ermöglicht, klein am Rande des Bildschirmes platziert, ein konstantes Verfolgen der Aktivität am Server.

Auf dem INN Server der TU wurden 2 Probleme mit diesem Verfahren festgestellt:

- Ganz selten aber doch wurden manchmal neue Artikel mit der `NEWNEWS` Methode nicht sofort gelistet.
- Ließ man eine Verbindung zum Newsserver ständig offen, um periodisch nach neuen Artikeln zu fragen, wurde der INN um 03:00 in der Nacht immer „unkooperativ“. Der aktiven Verbindung wurden alle neuen Artikel zwar gelistet, aber der effektive Inhalt einfach verweigert. Man musste die aktive Verbindung beenden und neu verbinden.

Ein Software Update des INN behob letzteres Problem, womit ein Programmfehler als verifiziert angenommen werden kann.

Abbildung 3.1: TU Newsserver Ticker

Bezüglich ersterem Problem kann ohne genauere Untersuchung der Serversoftware selbst nur spekuliert werden. Nach Untersuchung der betroffenen Artikel und deren Kontext in den Diskussionen wird vermutet, dass diese einen unerwünschten Zwischenhalt in einem Buffer am Server absolvierten. Der Artikel wurde geschrieben und auf den Server transferiert, verbleibt dort eine Zeit x und wird erst verspätet aktiv verfügbar gemacht. In der abgelaufenen Zeitspanne ist jedoch das Zeitfenster des Newstickerprogrammes schon weitergelaufen und somit wird der verspätete Artikel nicht mehr bemerkt (trotz bewußt gewählter überlappender Zeitbereiche der einzelnen Abfragen).

Auf Grund der überwiegenden Nachteile deaktivieren viele Serveradministatoren den `NEWNEWS` Befehl.

3.3.2 „Brute Force“ Synchronisation mittels LISTGROUP

Ein Newsserver führt zu Verwaltungszwecken eine lokale Liste der Artikel in einer Gruppe. Die Artikel werden einfach von 1 beginnend durchnummeriert. Immer wenn ein neuer Artikel der Gruppe hinzugefügt wird, wird diesem (letzte Nummer)+1 zugewiesen. Also

```
1 : <Message-ID>
2 : <Message-ID>
3 : <Message-ID>
...
```

Der `LISTGROUP newsgroup` Befehl listet alle aktiven Artikelnummern in der spezifizierten Gruppe auf.

Der Vorteil: Man bekommt eine Übersicht aller aktiven Artikel in einer Gruppe. Vergleicht man das Ergebnis mit einem vorherigen Lauf, können durch Differenzbildung leicht neue und entfernte Artikel herausgefiltert werden.

Nachteile:

- Es ist unbekannt, wieviele Artikel der Server als Antwort liefert. Auf modernen Servern mit Haltezeiten bis zu einem halben Jahr (180 Tagen) kann die Antwort bei aktiveren Gruppen *sehr* groß werden.
- Die benötigte eindeutige Identifikation der einzelnen Artikel, also die Message-ID, muß in einem seperaten Schritt ermittelt werden.
- Wenn Artikel nicht mehr aufscheinen, ist nicht klar wodurch. Fehlen Artikel am Anfang der Liste, war es wahrscheinlich durch einen `expire` Vorgang, fehlen sie „mittendrin“, war es meist ein `cancel` oder `supersede` (siehe Kapitel 3.2.3). Eine genauere Feststellung ist jedoch nicht möglich.

Der `LISTGROUP` Befehl war nicht von Anfang an Teil des NNTP Protokolls und wurde erst in [RFC2980] dokumentiert, er ist aber schon so lange Standardausstattung eingesetzter Serversoftware, dass er nicht als Erweiterung empfunden wird.

Praktische Erfahrungen

In der Implementation ist diese Methode relativ anspruchslos, ein einfaches Shellskript mit ein paar Zeilen Code kann den Abgleich durchführen. Im Jahre 2002 wurde so ein Skript implementiert und dann (fast) jede Nacht einmal über die `at.*` und `de.*` Hierachie laufen gelassen – die gesammelten Artikel bilden die Datengrundlage für diese Diplomarbeit.

Effizienz ist nicht gegeben, aber mitten in der Nacht durchgeführt, stört die Leitungs- und Serverbelastung meist weder Serverbetreiber noch andere Netzteilnehmer.

3.3.3 „Smarte“ Synchronisation mittels XHDR und control.cancel

Sind die Anforderungen an Effizienz und Vollständigkeit höher, läßt sich dies durch Kombination mehrerer NNTP Befehle erreichen:

`LIST ACTIVE newsgroups`
liefert eine Liste von
`group lastnum firstnum p`
(Gruppenname, höchste verfügbare Artikelnummer, niedrigste verfügbare Artikelnummer, Rechtestatus)

`LIST ACTIVE` liefert somit anhand der Artikelnummern die Information, ob Artikel `expire`d oder neu hinzugekommen sind.

Die Nachrichten `Message-ID` etwaiger neuer Artikel kann mittels
`XHDR header range`
bestimmt werden.

Weiß man von obigem `LIST`, dass es neue Nachrichten gibt, wählt man die gewünschte Gruppe zuerst via
`GROUP newsgroup`
aus, und holt sich die `Message-ID` zum Beispiel via
`XHDR Message-ID 12345-12347`

Der Artikel selbst – so man ihn noch nicht kennt – kann normal per
`ARTICLE <Message-ID>`
bezogen werden.

Der `XHDR` Befehl war zuerst nur Teil der populären Serversoftwareimplementationen, in Zukunft (siehe [nntpext]) wird er (bzw. ist er manchmal schon) als `HDR` verfügbar sein.

Das verbleibende Problem der Artikellöschungen in der Mitte des aktuellen Gruppenartikelbestandes läßt sich durch Überwachung von `control.cancel` lösen.
`control.cancel` ist keine „echte“ Gruppe, sondern enthält generierte Statusnachrichten, die Auskunft über Artikellöschungen („cancels“) geben. Für jeden Löschvorgang wird eine Nachricht generiert.

Diese spezielle Gruppe wird jedoch nicht von jedem Serveradministrator zum Lesen durch normale Benutzer freigegen. Auch zu beachten ist, dass sich in dieser Gruppe die Löschnachrichten aus allen Gruppen des Servers sammeln – es ist unter Umständen besser, gezielt ein paar einzelne Gruppen komplett neu zu synchronisieren, als die Löschnachrichten von 10000 anderen Gruppen mit zu analysieren.

Praktische Erfahrungen

Diese Synchronisation wurde 2004 vom Autor im Projekt „TUGnews/CancelbotV2“[7] über die `tu-graz.*` Gruppen erfolgreich implementiert. Im lokalen Netzwerk und mit einer kleinen Gruppenhierachie fällt selbst ein oftmaliger Abgleich (alle 20-30 Sekunden) aus Last Perspektive nicht mehr ins Gewicht.

Eine gute Programmierung[8] und ein lokaler Artikelcache verhindern unnötige Doppelabfragen. Die Beobachtung der Löschvorgänge in `cancel.control` und die Beachtung etwaiger `supersede` Hinweise in den Artikelheadern gewährleistet einen quasi 100%ig exakten Artikelabgleich.

[7] `http://www.zid.tugraz.at/ki/internet/news/charta/cancelbot.html`
[8] Eine in allen Situationen effiziente Abfrage benötigt ca. 200 Zeilen Skript-Programmcode.

Eine Anpassung dieser Methode, sodass ein effizienter Abgleich von `at.*` und `de.*` auch auf einem großen Server möglich wäre, ist Raum für zukünftige Experimente.

3.4 Artikelspeicherung

Newsserver wurden zum Nachrichtenaustausch entworfen, die reinen Nachrichtentexte bewegen sich meist im x kb Größenbereich. Die Programmierung eines Newssystems muß sich somit primär mit der Speicherung von sehr vielen kleinen Datenbrocken beschäftigen. Für die logistische Umsetzung gibt es mehrere Ansätze die Daten zugriffs- und speichereffizient zu organisieren.

3.4.1 Traditionelles Filelayout

Anfangs war der Artikeldurchsatz auf Newsservern gering und die Rechnerleistung bewegte sich auch in sehr beschränktem Rahmen. Die Programmierer benutzten ein möglichst einfaches Dateilayout zur Abspeicherung der Artikel, Gruppen wurde an ihren „.“ Trennstellen jeweils in Unterverzeichnisse aufgeteilt. Zum Beispiel:

```
.../tu-graz/algorithmen
.../tu-graz/anzeigen/computer
.../tu-graz/anzeigen/diverses
.../tu-graz/betriebsysteme/linux
.../tu-graz/diverses
.../tu-graz/flames
.../tu-graz/telematik
.../vc-graz
```

Die Artikel selbst sind reine Textdateien, liegen im entsprechenden Gruppenverzeichnis und ihr Filename ist die interne Artikelnummer in der Gruppe auf diesem Server.

Dieses Speicherlayout ist einfach zu programmieren, leicht zu sichern und die implizite Speicherung der Struktur im Filesystem des Betriebsystems erspart zusätzliche Verwaltungsstrukturen.

Mit zunehmendem Artikelaufkommen bildet sich jedoch eine signifikante Abhängigkeit des Newsserverdurchsatzes von der Effizienz des Filesystems. Von Interesse sind im speziellen die Verwaltung von vielen Einträgen pro Unterverzeichnis, als auch die Allokation des Speicherplatzes selbst.

Auf der Linux Plattform hat sich die Verwendung von ReiserFS[9] als signifikante Performancesteigerung erwiesen. Die Liste der Verzeichniseinträge wird nicht mehr als lineare Liste, sondern als Baumstruktur gespeichert. Die Allokation des Speicherplatzes erfolgt nicht in fixen Blöcken, was insbesondere auf Newsservern mit vielen kleinen Artikeln bis zu 50% Speicherplatzverschwendung bedeutet, sondern die Daten werden ohne Zwischenraum auf der Platte aneinandergereiht.

3.4.2 Zyklische Buffer

Hat man kein Interesse auf die Artikelfiles des Newsservers direkt zuzugreifen[10], bieten moderne Newsserver die Speicherung als zyklischer Buffer: Man schätzt den ungefähr benötigten Speicherplatz, den alle Artikel einer Subhierachie zusammen im (selbst) festgelegten `expire` Zeitraum benötigen, addiert noch einen Sicherheitsrahmen dazu und legt ein File dieser Größe an.

Der Newsserver speichert die ankommenden Artikel einen neben den anderen in das File. Er merkt sich jede Position und Größe eines Artikels und Zeiger auf Anfang und Ende des im File schon belegten Bereiches. Die Zeiger bewegen sich immer in die gleiche Richtung.

Neue Artikel werden an einem Ende angefügt, Artikel aus Altersgründen am anderen Ende entfernt. Zyklisch, weil, hat ein Zeiger das Ende des Files erreicht, fängt er wieder bei Position 0 an. Der extra einkalkulierte zusätzliche Speicherbereich sollte ein aufeinandertreffen von Anfang- und Endezeiger verhindern. Tritt dies ungünstigerweise doch einmal auf, werden eben Artikel vor ihrem offiziellen `expire` bereits entfernt.

Der Vorteil dieses Buffer-Systems liegt in der Einsparung der für jeden Zugriff nötigen Abarbeitung der Verwaltungsfunktionen des Filesystems. Mehr noch, kann der Newsserver per „mmap“[11] direkt auf die Daten in der Datei zugreifen, entfällt jeglicher Verwaltungsaufwand, außer einmaligem öffnen, mmap und schließen.

Der Nachteil ist, die Daten sind im Käfig des Newsservers eingesperrt. Ein Backup des großen Bufferfiles ist nur als Ganzes möglich und alle Lesezugriffe nur via offizieller Servicefunktionen.

[9] `http://en.wikipedia.org/wiki/ReiserFS`
Es gibt auf der Linux Plattform mittlerweile noch weitere fortgeschrittene Dateisysteme mit interner Baumstruktur und anderen leistungssteigernden Merkmalen. ReiserFS wurde vom Autor im Vertrauen auf dessen bewährte Stabilität hin gewählt.

[10] Als jeder zusätzliche Rechner noch eine enorme Investition war, war der Newsserverrechner manchmal auch gleichzeitig der Rechner, auf dem die Benutzer arbeiteten. Ein Newsreader konnte zum reinen lesen somit eine Abkürzung nehmen, einfach direkt auf die Artikelfiles zugreifen.

[11] Ein File wird vom Betriebsystem direkt in den Hauptspeicher eingeblendet – „memory mapped“. Zugriffe und Änderungen benötigen keine Fileoperationen und werden transparent vom Betriebsystem durchgereicht, die Speicherverwaltung ist effizient direkt an das Virtual Memory gekoppelt. `http://en.wikipedia.org/wiki/Mmap`

3.4.3 Zeitliche Sortierung plus Hashes

Bei der zu erwartenden Datenmenge für diese Diplomarbeit, mehrere Millionen Artikel, war eine günstige Speicherung der Daten ein zu berücksichtigender Faktor. Die Speicherung von allen Artikeln einer Gruppe in einem Unterverzeichnis, wie in Kapitel 3.4.1 beschrieben, ist selbst auf modernen Filesystemen bei mehr als 200000 Einträgen performant problematisch. Die Artikelnummern selbst sind für zukünftige Untersuchungen (oder als Filenamen) nicht mehr von großer Bedeutung, sie geben ja nur den zeitlichen Eingangsverlauf auf einem speziellen Server wieder.

Interessant ist die zeitliche Abfolge der Artikel. Diese kann durch Auswertung der „`Date:`“ Stempelung, generiert beim Abschicken eines Artikels durch den Newsreader, gewonnen werden. Jeder Entstehungszeitpunkt eines Artikels kann durch einen String der Form `YYYYMMTTSSMMSS`, also 14 Zeichen, dargestellt werden.

Ein weiteres wichtiges Merkmal ist die eindeutige `Message-ID` jedes Artikels. Diese besitzt eine merkliche Schwankungsbreite in ihrer Länge. Um eine automatisierte Verarbeitung effizienter zu gestalten, bietet sich die Umrechnung der `Message-ID` in ein einheitlicheres Format an.

Eine Anwendung des MD5 Hash[12] Algorithmus auf den `<..@..>` String einer `Message-ID` liefert einen 128-bit Hashwert, der üblicherweise mittels 32 Hexadezimalzeichen dargestellt wird. Generiert man von jedem Artikel den Zeitstring plus seinen MID Hashwert, kann man eine Artikelkennung der Form
`YYYYMMTTSSMMSShhhhhhhhhhhhhhhh[...]`
gewinnen.

Eine Datenhaltung von Artikeln mit dieser Kennung als Filename bzw. als Indizierung bietet

- Ein „natürliche“ Sortierung der Artikel nach ihrer effektiven Entstehungszeit (leider mit einem gewissen Fehlerfaktor behaftet, nicht jeder PC Besitzer kümmert sich gewissenhaft um die genaue Einstellung seiner Systemzeit).
- Eine einfache Möglichkeit, neu ankommende Duplikate schnell zu entdecken, die Berechnung aus einem Artikelkopf ist eindeutig und einfach.[13]
- Eine Suchoperation vom Datenanfang bis zu deren Ende ist trivial. Operationen die in einem lokalen Bereich operieren (z.B. auflösen der Artikelabhängigkeiten) sind einfach zu implementieren.

[12] Ein Hash Algorithmus liefert eine Art Prüfsumme über einen Datenblock. Ein bestimmter Datenblock liefert genau einen bestimmten Hashwert. Es ist rechentechnisch sehr aufwändig bis unmöglich einen Datenblock zu konstruieren, so dass dieser einen bestimmten Hashwert liefert.
Siehe [RFC1321] und `http://en.wikipedia.org/wiki/Md5`

[13] MD5 ist quasi in jeder Programmmierumgebung als Bibliotheksfunktion verfügbar.

- Es bietet sich an, alle Artikel eines Tages in jeweils ein Unterverzeichnis der Form `YYYYMMTT` zu speichern, welches ein einfaches Archivierungssystem ermöglicht – zum Beispiel werden immer nur die letzten 365 Tage (also Verzeichnisse) online gehalten und jeweils am Ende eines Tages immer der älteste Tag archiviert.
- Die Länge des Hashteils kann nach Bedarf und Anwendung gewählt werden. In der Praxis sind 32 Zeichen ein viel zu großer Adressraum. Selbst bei der Wahl von nur der Hälfte oder einem Drittel der 32 Zeichen wird es nur selten zu Kollisionen der reinen Hashwerte kommen.

3.4.4 Datenbank

Die Abspeicherung von einem Artikel (im Klartext) pro File bot für diese Diplomarbeit einen entscheidenden Vorteil: Sämtliche üblichen Textbearbeitungstools die bei einem modernen Linuxsystem mitgeliefert werden, konnten zur Bearbeitung und/oder Analyse der Daten eingesetzt werden. Das Textformat ist in der Unixkultur tief verwurzelt, noch heute sind Konfigurationsfiles meist normale Textfiles, entsprechend leistungsfähige Tools zu deren Manipulation stehen zur Verfügung.

Eine alternative Einspeisung in eine industriestarke Datenbank bietet sicherlich Verbesserungen im Zugriff bzw. ein besseres Cacheverhalten bei wiederholten Anfragen als der normale Diskcache des Betriebsystems. Erkauft würde dies jedoch mit dem Verlust von Flexibilität, jede Zugriffsoperation auf die Datenbank müsste explizit in einer höheren Programmiersprache implementiert werden.

Im Laufe der Diplomarbeit wurde zusätzlich zur Einzelspeicherung ein Kompromiss implementiert: Alle Artikel wurden in ein großes File zusammengefasst und zusätzlich eine seperate Indexdatei in ASCII Klartext angelegt. Der Vorteil von einem großen File (siehe auch 3.4.2) ist gegeben, durch den simplen Index sind jedoch einfache Zugriffsmöglichkeiten durch Skripte weiterhin möglich.

4 Statistik

Es gibt kaum öffentlich downloadbare Archive von Usenet Gruppen (oder ganzen Hierachien). Dienste wie Google Groups speichern intern komplette Archive, bieten aber keinen Zugriff auf die Rohdaten an, sondern nur verschiedene Lesefunktionen durch ein Webinterface. Private Archive gibt es sicher von mehreren Teilnehmern mit guter Netzanbindung, Einblick gewähren diese jedoch nur manchmal und nur in Teilen.

Im Rahmen dieser Diplomarbeit wurde eine genauere Betrachtung des Datenbestandes der `de.*` und `at.*` Hierachien vorgenommen. Die große Menge an Artikeln in einem Archiv vereint, ermöglicht im folgenden Kapitel nicht nur rein quantitative Messungen, sondern bietet auch Einblick, inwieweit einzelne Eigenschaften wirklich in der Praxis ein- bzw. umgesetzt werden.

4.1 Datenmenge in Zahlen

Zu Beginn der Datensammlung war noch nicht klar, dass die gesammelten Daten einmal in einer Diplomarbeit Verwendung finden würden, deswegen wurde ein relativ simples, wie in Kapitel 3.3.2 beschriebenes Skript, zur Artikelarchivierung verwendet. Insgesamt wurden folgende Datenmengen archiviert.

4.1.1 Rohdaten

Die Artikel Rohdaten entstammen aus einfach aneinandergefügten, periodischen nächtlichen Abgleichungen – Löschungen, Ersetzungen oder andere nach Artikelabsendung verändernde Operationen sind hiermit nicht berücksichtigt. Die Daten wurden vom Server `aconews.univie.ac.at` bezogen, dem zentralen Usenetserver für das ACOnet[1].

Als Zeitraum für diese Diplomarbeit wurden die Jahre 2003 und 2004 ausgewählt. In diesem Zeitraum gab es sowohl von `aconews` Seite, als auch lokal keine größeren Netzprobleme und die gewonnen Daten sollten somit so vollständig wie möglich sein. Die Artikel einer Gruppe wurden zuerst jeweils in ein Unterverzeichnis gespeichert, es ergibt sich eine Rohdatengröße von 8330223 `de.*` und 458164 `at.*` Artikeln.

[1] ACOnet ist das österreichische Datennetz für Wissenschaft, Forschung und Lehre. `http://www.aco.net/`

4.1.2 Aufbereitung

Bei der Speicherung in Unterverzeichnissen nach Gruppen nehmen Artikel, die in mehreren Gruppen gleichzeitig aufscheinen, sogenannte „crosspostings“, durch die mehrfache Speicherung unnötig Platz ein. Duplikate werden erst durch die Konvertierung in ein Speicherformat wie in Kapitel 3.4.3 beschrieben eliminiert.

Zur Abschätzung einer sinnvollen Länge für den Hashteil wurden alle Hashes der `Message-ID` der `de.*` Artikel einmal berechnet. Ab einer bestimmten Länge für den Hashteil ist die eindeutige Zuordnung Hashwert - Artikel (fast) gegeben.

Zeichen	Hashduplikate	% Wertebereich Ausnutzung	% von 8330223 (`de.*` gesamt)
4	65536	100,00	
5	1046148	99,77	
6	1613042	9,61	19,36
7	138122	0,00..	1,66
8	8768	...	0,10
9	531	...	0,00...
10	44	...	
11	15	...	
12	13	...	
13	13	...	

Tabelle 4.1: Hashlänge versus Eindeutigkeit

Bei 4 (Hexadezimal)zeichen ist der mögliche Hash Wertebereich (16^4) natürlich viel zu klein, 100% aller möglich bildbaren Hashstrings sind mehrfach vergeben. Länge 5 und 6 sind auch noch signifikant übersättigt, wenn auch nicht mehr zu 100% ausgelastet. Erst bei Länge 7 bietet ein exzessiv großer Hash Wertebereich (16^7) genug Freiraum. Es ist besser ab hier die Anzahl der Duplikate im Vergleich zur Gesamtzahl der Artikel zu betrachten.

Es entsteht ab einer gewissen Länge ein Sättigungseffekt. Man könnte die maximal mögliche Länge von 32 Zeichen nehmen und hätte trotzdem noch Duplikate. Das bedeutet, die `Message-ID` zweier Artikel in einem Zeitraum von 2 Jahren ist wirklich ident, ein Fall welcher laut RFC Spezifikation der Eindeutigkeitsbedingung von `Message-ID`s nicht möglich sein dürfte.

Es gibt 2 Möglichkeiten solcher `Message-ID` Kollisionen:

- Ein Roboterprogram postet periodisch immer den gleichen Artikel, zum Beispiel eine FAQ Liste, versieht diesen aber fehlerhafterweise immer mit der gleichen `Message-ID`.
- Manche Newsreader sind in ihrer Erstkonfiguration auf `@localhost.localdomain` als rechter `Message-ID` Teil eingestellt. Selbst wenn der linke Teil per Zufall generiert wird, gibt es doch bei entsprechend großer Artikelmenge irgendwann Kollisionen.

Abhilfe in beiden Fällen ist es, die `Message-ID` immer serverseitig zu generieren (beziehungsweise überschreiben zu lassen), egal ob ein neu eingelieferter Artikel schon eine besitzt oder nicht.

Für den Datenpool wurde eine Hashlänge von 10 gewählt. Zusätzliche 14 Zeichen kommen vom Zeitstring und ein extra „x“ in der Mitte zur optisch besseren Lesbarkeit. Dies ergibt eine 25 Zeichen lange Kennung pro Artikel.

Die letzten Artikelkennungen für die `de.*` Hierachie lauten nach Konvertierung somit:

```
...
20041231235403xc12363e883
20041231235657x4540b3e478
20041231235817x22c4df90bd
20041231235857x791ba7c50d
20041231235939xa89223117c
```

4.1.3 Überblick

Nach erfolgter Konvertierung gibt es 8075520 `de.*` und 452007 `at.*` Artikel. Die Differenz von ca. 254000 `de.*` und ca. 6000 `at.*` Artikeln zu der Zahl in Kapitel 4.1.1 sind jene Artikel, die in mehr als einer Gruppe aufscheinen und zuvor mehrfach gezählt wurden.

Die Jahre 2003 und 2004 haben zusammen 731 Tage[2]. Graphisch aufbereitet, wobei ein Pixel horizontal einen Tag bedeutet und vertikal der Artikelumsatz aufgetragen ist, ergibt sich ein Graph wie in Abbildung 4.1.

Deutlich sichtbar in beiden Graphen sind die einzelnen Wochen, ein Wochenende bringt immer einen signifikanten Einschnitt beim Artikelumsatz. Manche Menschen haben am Wochenende kein Internet bei sich zu Hause und viele gehen einfach vor die Tür – jedenfalls weit weg von einem Computer.

[2] 2004 war ein Schaltjahr

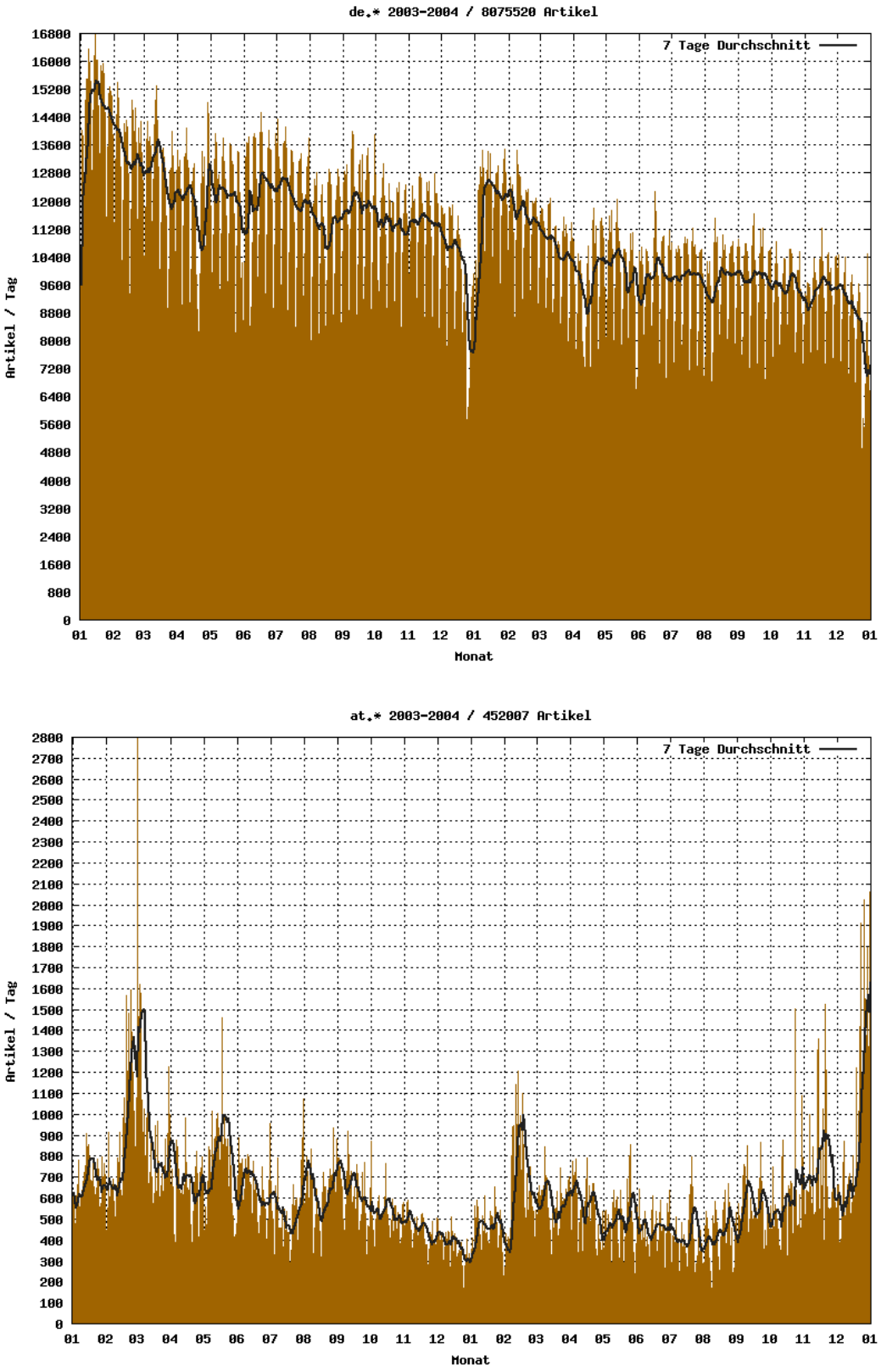

Abbildung 4.1: Täglicher Artikelumsatz je Hierachie

Im Falle von `de.*` läßt der Graph nur einen Schluß zu, die Anzahl der Artikel nimmt ab. Dies deckt sich mit anderen veröffentlichten Statistiken[3]:

Jahr	Postings	DA Zählung Postings
2004	3758579	3681779
2003	4472961	4393741
2002	5338231	
2001	5886271	

Tabelle 4.2: Artikel Jahresumsatz `de.*`

`de.*` bietet täglich durchschnittlich 11047 neue Artikel an, `at.*` 618. Bei dieser kleinen Artikelmenge in `at.*` läßt sich kein eindeutiger Trend festmachen. Die Anzahl der Gruppen in `at.*` ist seit langer Zeit konstant, die Teilnehmerzahl stagniert und wenn es mal in einer Gruppe einen „Aufreger“ gibt, schlägt sich dies schnell in der Gesamtstatistik als Spitze nieder.

4.1.4 Gruppenzahl

Die `de.*` Hierachie zählt über den gemessenen Zeitraum von 2 Jahren insgesamt 548 Gruppen, `at.*` 74 Gruppen. Nicht alle Gruppen waren jedoch über die gesamte Laufzeit existent, es gibt eine kontinuierliche Weiterentwicklung des Angebotes. Gruppen mit wenig bis keinem Umsatz werden gelöscht, neue bei Bedarf angelegt und zum Beispiel die populäre Gruppe `de.alt.etc.auktionshaeuser` wurde Ende 2004 nach `de.etc.handel.auktionshaeuser` transferiert und geht somit in obigen 548 doppelt ein.

Sortiert man die Gruppen nach Artikelanzahl, ergibt sich Tabelle 4.3, die Gruppen in `de.*` und `at.*` mit den größten Artikelumsätzen.

Die Gruppe `at.freizeit.nonsens` nimmt hierbei eine Sonderstellung ein. Eine einzige Gruppe erbringt 32,6% des Artikelumsatzes der gesamten `at.*` Hierachie. Der Inhalt entspricht der Namensgebung – Nonsens. Wenige Personen nutzen diese Gruppe für zielloses „rumquatschen“. Diese „blabla“ Gruppen sind in `de.*` mit `.dummschwatz`, `.talk.bizarre` und `.gruppenkaspar` auch spitzenmäßig vertreten, prozentuell gesamt gesehen jedoch unbedeutender (zum Beispiel `.dummschwatz` auf Platz 6 erbringt „nur“ <1,5% der Gesamthierachie).

[3] `<359u7tF4kejckU1@individual.net>` in de.admin.news.groups
„Postings in de.ALL - Jahresueberblick 2004“
`http://groups-beta.google.com/group/de.admin.news.groups/msg/0e707322a7fb8306?dmode=source`

217586	de.soc.politik.misc	149151	at.freizeit.nonsens
195848	de.rec.fotografie	36359	at.linux
153659	de.talk.tagesgeschehen	33145	at.freizeit.motorrad
132245	de.alt.rec.digitalfotografie	26333	at.gesellschaft.recht
130389	de.soc.recht.misc	23394	at.gesellschaft.politik
121211	de.alt.dummschwatz	23034	at.freizeit.auto
118759	de.etc.sprache.deutsch	16887	at.verkehr.bahn
115792	de.comp.sys.mac.misc	16761	at.freizeit.sonstiges
115503	de.rec.fahrrad	15706	at.telekomm.mobil
110368	de.comp.os.unix.linux.misc	10300	at.internet.provider
110355	de.etc.fahrzeug.auto	9799	at.test
104407	de.alt.etc.auktionshaeuser	8367	at.region.noe
98466	de.talk.bizarre	8337	at.anzeigen.computer.pc
97601	de.alt.gruppenkasper	6539	at.region.graz

Tabelle 4.3: Gruppenumsatzspitzenreiter

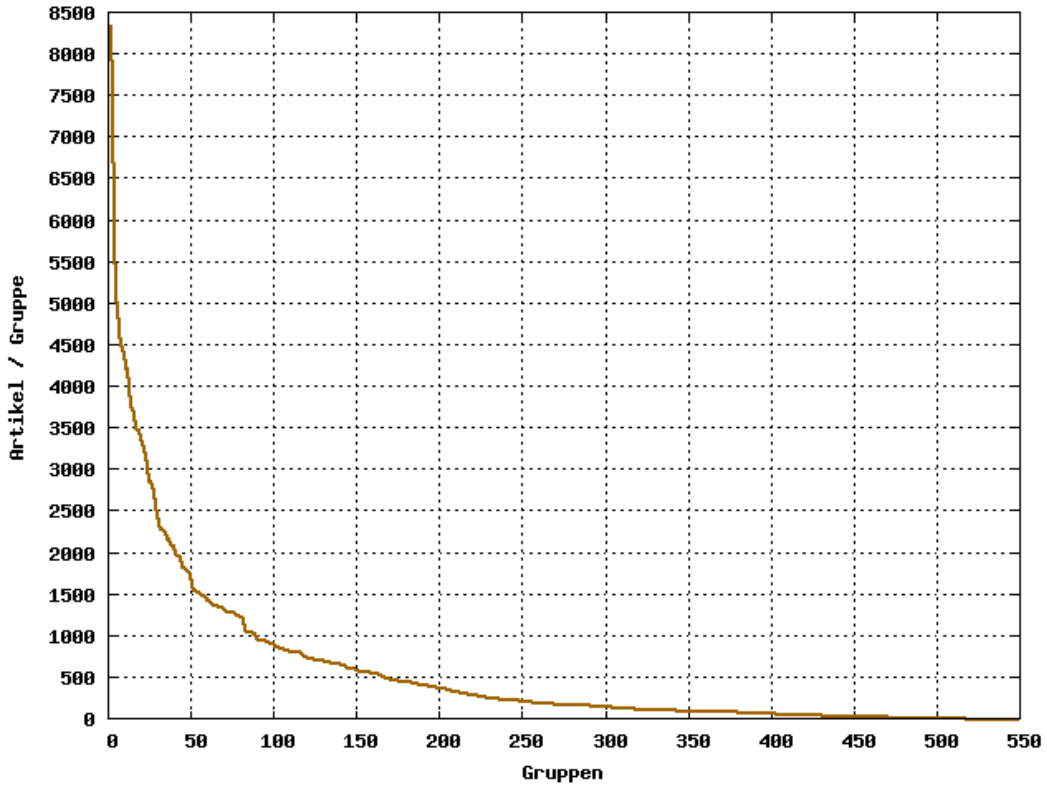

Abbildung 4.2: `de.*` – durchschnittlicher Artikelumsatz pro Gruppe in 28 Tagen

In weiterer Folge dieser Diplomarbeit wird `at.freizeit.nonsens`, kurz „afn“, manchmal aus Berechnungen ausgenommen, die „Monokultur“ dieser Gruppe wirkt sonst zu stark auf das Ergebnis ein.

Sortiert man die 548 `de.*` Gruppen nach ihrem Artikelumsatz, ergibt sich eine Verteilungskurve wie in Abbildung 4.2. Oder in Zahlen ausgedrückt, die Top 10 Gruppen erbringen 16,9% der gesamten Artikel. Top 20: 28,4%, Top 30: 37,2%, Top 40: 44,0%. Die Top 50 Gruppen, also run 10% der Gesamtgruppenanzahl erbringen 49,8%, also rund die Hälfte aller Artikel.

Dies bedeutet nicht, die anderen Gruppen seien nicht so wichtig. Man beachte die Einheiten des Graphen, dargestellt wurde der Artikelumsatz von rund einem Monat. Wieviele Artikel kann jemand pro Tag lesen, wieviel Zeit will man investieren? Auch wenn statistisch gesehen ein großer Teil der Hierachie weniger aktiv ist, sind in diesen Themengebieten noch genug Artikel, um vielen Teilnehmern mehr als genug Lese- beziehungsweise Diskussionsstoff pro Tag zu liefern.

Eine Komplettlistung aller Gruppen, ihrer wichtigsten Kenndaten und zusätzliche Graphen können in Anhang B nachgeschlagen werden.

4.2 Header

Länderhierachien erfahren im Allgemeinen eine Verbreitung auch über Landesgrenzen hinaus. Das Internet ist weltweit. Wer sich für eine Sprache, Land oder bestimmte Kultur interessiert, kann einfach eine Weile die entsprechende Usenet Abteilung mitlesen und somit einen Einstieg in fremde Kulturen erfahren.

Bedingt durch die globale Leserschaft durchlaufen die Artikel auch vielerlei Newssoftware in verschiedensten Konfigurationen, welche Spuren in den Steuerinformationen des Artikels – im sogenannten Header – hinterlassen.

Die Headerinformationen, die sinnvollerweise jeder Artikel besitzen muß, wurden bereits in Kapitel 2.3 erläutert. Bei der Aufarbeitung des Datenpools wurden folgende Beobachtungen bei ausgewählten Headern gemacht.

4.2.1 Nur Header, kein Haupttext

Der Fall, dass ein Artikel nur aus einem Kopf besteht, aber keinen Haupttext enthält, mag auf den ersten Blick nicht sehr sinnvoll erscheinen, bei näherer Untersuchung der 50 Kandidaten in `at.*` wo dieser Fall auftrat, bieten sich jedoch mehrere Erklärungsmöglichkeiten an.

Die 50 gefundenen Artikel verteilen sich auf 4 Gruppen:
`at.anzeigen.computer.mac`(1)
`at.anzeigen.computer.pc`(2)
`at.linux`(1)
`at.test`(46)

Eine Möglichtkeit ist die des Testartikels. Es gibt keinen Haupttext weil nur eine Funktion der Newssoftware getestet wurde. Dieser Fall ist überwältigend vertreten. Weiters ist es manchmal sinnvoll in einer Anzeige keinen Haupttext zu verwenden, weil bereits alles wichtige im Titel angegeben wurde, zum Beispiel „S: CPU Pentium 200MMX“.

Zum Vergleich, in `de.*` finden sich 46 Artikel verteilt auf 18 Gruppen. Hier konnte noch ein weiterer Typus gefunden werden: Spam. Im Artikeltitel ist in diesem Fall nur eine Werbe http://... Adresse angegeben, ein zusätzlicher Artikeltext wird nicht benötigt.

Technisch gesehen gibt es noch einen Fall, der aber während des Herunterladens des Artikels auftrat: Der Newsserver liefert einfach keinen Haupttext aus, weil er ihn nicht mehr hat. Der Artikelkopf ist noch in einem internen Cache vorhanden und wird auf Abfrage geliefert, bei Nachfrage nach dem gesamten Artikel gibt es jedoch eine Fehlermeldung. Dies ist zwar protokollmäßig ein Standard konformes Verhalten, stellt aber Anforderungen an die Robustheit der Newsreadersoftware ob der unerwarteten Serverantworten. Dieser Fall wurde nach Auffindung im Abgleichskript relativ früh abgefragt und korrigiert, im Datenpool wurden solche Artikel somit nicht gespeichert.

Da die Anzahl der Artikel mit keinem Haupttext relativ klein war, wurden diese einfach aus dem Datenpool entfernt und finden in dieser Diplomarbeit keine weitere Verwendung.

4.2.2 Kein Subject

Ein Artikel ohne Titel dürfte eigentlich nicht existieren, in `at.*` wurden jedoch 12 Exemplare aufgefunden. Alle stammen vom gleichen Absender und wurden laut Header vom Programm „Microsoft Outlook Express Macintosh Edition 4.5“ generiert. Das Problem besteht darin, dass der `Subject` Teil zwar generiert wurde, aber nicht ordnungsgemäß in einer eigenen Zeile codiert, sondern einfach ohne Trennzeichen an die vorhergehende angefügt wurde.

Dies legt ein Problem mit den verschiedenen [Zeilenende]codierungen[4] nahe. Mac Systeme waren ursprünglich CR basierend, Unix verwendet nur LF und Windows Systeme und das NNTP Protokoll erwarten ein CR-LF. Wird ein Artikel zwischen diesen Systemem konvertiert und es passiert ein Fehler, kann es durch nicht robust programmierte Software zu unerwarteten Resultaten kommen.

[4] CR = 13; LF = 10; CR-LF = 13 10

Wo der Fehler genau lag, kann im Nachhinein nicht festgestellt werden. Die Daten wurden für diese Diplomarbeit jeweils vor dem lokalen Speichern auf „Unix (nur LF)“ konvertiert. Ob der Konvertierungsvorgang fehlerhaft ist/war, oder ob der INN Server fehlerhaft ausgeliefert hat, konnte nicht festgestellt werden. Es ist jedoch anzunehmen, dass dieser Fehler auch andere (seltenere) Header beschädigt hat, nur eben nicht immer so auffällig ist, wie bei `Subject`.

Artikel mit fehlerhaften `Subject` wurden für die weitere Untersuchung verworfen.

4.2.3 Falsches Datum

Jeder Artikel bekommt bei seiner Erstellung einen Zeitstempel wann dies geschehen ist. Dieser Zeitstempel sollte möglichst genau sein und das Format für diese `Date` Angabe ist genau spezifiziert[5].

Gute Newsserver sollten grundsätzlich eine Annahme von Artikeln aus der Zukunft verweigern. Dies kann mit einer gewissen Toleranz implementiert werden, zum Beispiel, um die immer wieder auftretenden Fehler bei der Umstellung Winter- zu Sommerzeit (und umgekehrt) zu kompensieren. Unter den gesammelten Daten fanden sich aber Kuriositäten, die offensichtlich Unfug darstellen:

```
Tue, 25 Sep 2040 16:06:52 -0000
Wed, 26 Sep 2040 19:05:40 -0000
Sun, 16 Dec 2046 01:00:57 +0100
Sun, 16 Dec 2046 22:03:28 +0100
Fri, 15 Nov 2080 08:13:57 +0100
Mon,  4 Nov 2080 07:42:50 +0100
Wed, 18 Mar 2099 07:39:51 +0100
Wed, 18 Mar 2099 07:43:21 +0100
Wed, 18 Mar 2099 07:50:39 +0100
Thu, 22 Jan 2150 18:06:01 +0100
```

Diese falschen Angaben sollten theoretisch das Austauschprotokoll nicht beeinträchtigen, da gruppenintern Newsserver ja strikt nach Einlauf durchnummerieren, praktisch verwenden jedoch manche Newsreader und Archive den Artikelzeitstempel, um die zeitliche Abfolge bzw. Aufteilung festzustellen.

Ein Problem ergibt sich auf Systemen in denen Zeitpunkte mit 32bit intern gespeichert werden, auch bekannt unter [Unix time] (oder Posix time). Diese Art der Speicherung hat mit Zeiten ab 19.Jänner 2038 ein Überlaufproblem. Theoretisch wird dieses Problem erst in mehr als 30 Jahren aktuell, Software die jedoch unerwarteter Weise absurde Daten aus dem Jahre 21xx bekommt, erbringt mitunter „beliebige“ Ergebnisse.

[5] [RFC2822] Kapitel 3.3

Bei der in Kapitel 3.4.3 und 4.1.2 bereits beschriebenen Konvertierung trat genau dieses „Jahr 2038“ Problem bei der für die praktische Implementierungen dieser Diplomarbeit verwendeten Datumsbibliothek auf. Artikel mit `Date` Markierungen, die aus irgendeinem Grund nicht parsebar waren oder nicht in 2003 und 2004 lagen, wurden entfernt.

4.2.4 Message-ID

In Kapitel 3.4.3 und 4.1.2 wurde die schwankende Länge der `Message-ID` (MID) der Artikel bereits angedeutet (welches ein Hash Indexverfahren attraktiv machte). Abbildung 4.3 zeigt die Verteilung der MID Längen über die Artikelpopulation. Die Prozentangabe gibt an, wieviele Prozent der Artikel man verarbeiten könnte, wenn man für die MID die jeweilige Länge als maximalen Wert annimmt.

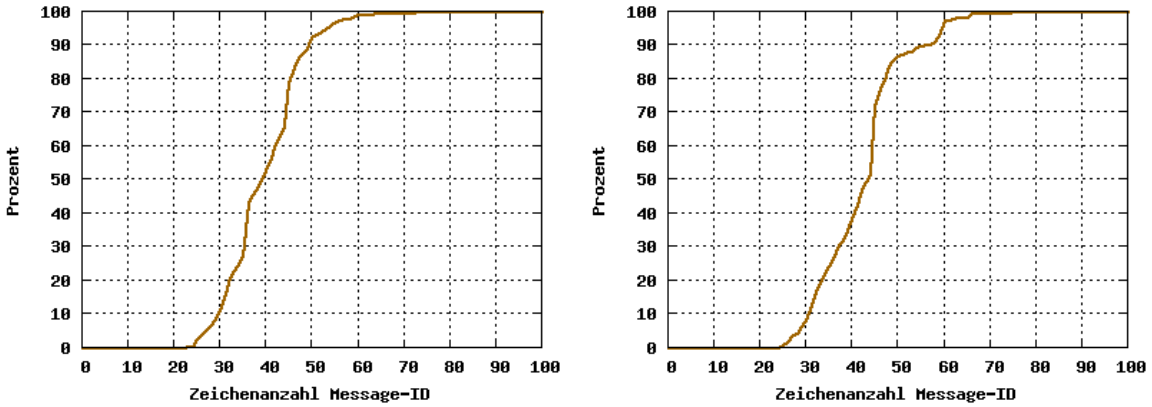

Abbildung 4.3: MID Längenverteilung, `de.*` (links), `at.*` (rechts)

Wie man sieht, die automatisch generierten[6] MID decken den Bereich bis ca. 60 Zeichen Länge großteils ab, längere MID werden eher bewußt manuell erstellt. Von der Softwareperspektive interessant sind die Extremfälle, in `de.*` gibt es knapp 2000 Artikel mit einer MID länger als 85 Zeichen – die Verarbeitung dieser 0,02% Sonderfälle macht die effiziente MID Zwischenspeicherung für zum Beispiel die Berechnung der Artikelabhängigkeiten zu einem Skalierungsproblem.

Ein paar Beispiele aus gefundenen, händisch erstellten überlangen MID:

```
<1.only_netterrorists_are_putting_those_long_strings_into_their_message
-ids.cpegm9.3vvpucp.1@ufh.invalid.de>
<slrnba6ev5.lvq.usenet-from_expires-@wer-das-hier-quoted-ist-doof-und-
hat-verloren.meine-domain-hier-ist-die-aller-aller-laengste-domain-
von-allen.de>
```

[6] Meist ein Hashwert aus Datum und Zeit der Einspeisung, plus einer Seriennummer.

```
<thisisjustapunkrocksongwrittenforthepeoplewhocanseesomethingswronglike
antsinacolonywdooursharebutthereresomanyfuckininsectsoutthere@news.
bmussler.de>
```

4.2.5 Content-Type

Anfangs wurden im Usenet System nur einfachste Nachrichten verschickt. Diese Nachrichten wurden in der am weitesten verbreiteten aller Codierungen, ASCII[7], geschrieben. ASCII hat einen Zeichenvorrat von 128 Zeichen[8]. Diese reichen aus, um die englische Sprache abzudecken, aber mit zunehmender Internationalisierung war es notwendig, erweiterte Zeichencodierungen zu verwenden, die zum Beispiel auch Umlaute erlauben. Diese Erweiterungen wurden standardisiert, wobei in den USA und westeuropäischen Gebieten hauptsächlich ISO-8859-1 und verwandte[9] Codierungen zur Anwendung kommen.

Ein über lange Zeit herrschendes Problem war jedoch die fehlende Deklaration in den Artikeln, welche Codierung denn nun verwendet wurde. Der ASCII Teil ist klarerweise immer gleich, aber alle Zeichen, die nicht ASCII sind, könnten theoretisch „irgendetwas“ sein. Innerhalb eines Sprachraumes war das Problem nicht so offensichtlich. Ein nicht Standard konformer Newsreader erzeugt einen Artikel mit Umlauten nach ISO xy, ein anderer liest diesen, findet keine Deklaration – und verwendet einfach seine gerade lokal verwendete Codierung. Die meist verbreitete war ISO-8859-1, die Artikel waren zwar nicht deklariert, aber es fiel auch niemandem auf.

Mit der Einführung des Euro Anfang 2002 in den täglichen Gebrauch wurde dieses Problem jedoch offensichtlich[10]. Die ISO-8859-1 Codierung besitzt kein €-Symbol, sondern (z.B.) die neue ISO-8859-15 muß hierfür verwendet werden. Newsreader, die strikt nach Standards operierten, zeigten immer schon einen Fehler – meist ein „?“ Zeichen – bei nicht eindeutiger Codierung. Mit der Euroeinführung begann eine langsame Umstellung (siehe Abbildung 4.4) von -1 nach -15 Codierung und viele Newsreader zeigten nun irgendwelche Zeichen bei einem Euro – nur eben keinen €.

Das Bewußtsein für eine ordnungsgemäße Codierung ist jedoch gestiegen. In der `de.*` Hierachie deklarierten Anfang 2003 82,3% der Artikel einen `Content-Type`, Ende 2004 schon 88,0%. Davon deklarieren sich 3,8% der Artikel minimalst als „us-ascii“, 94,0% als eine ISO-8859.. Variante, 1,2% als „windows-12xx“ und erst 0,9% besitzen eine moderne [Unicode] UTF[11] Codierung.

[7] ASCII = American Standard Code for Information Interchange, siehe [ASCII]
[8] 0-127, also die „unteren“ 7 bit eines Bytes.
[9] Die ISO-8859 Familie hat (im Moment) 15 verschiedene Codierungstabellen, siehe [ISO 8859]
[10] `http://einklich.net/anleitung/eurokodier.htm` auf [Volker]
[11] UTF = Unicode Transformation Format
Ein spezielle Codierungsform die den Übergang von ASCII auf Unicode aus programmiertechnischer Sicht erleichtern soll.

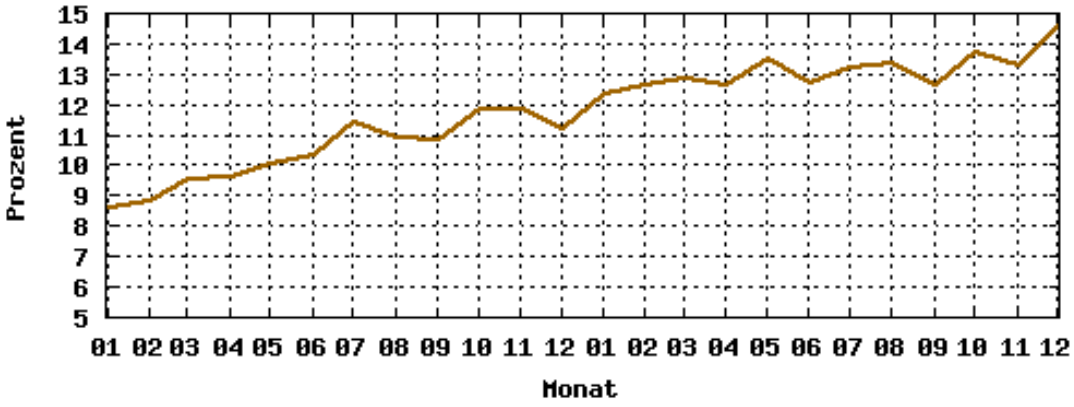

Abbildung 4.4: Marktanteil der ISO-8859-15 Verwendung in ISO-8859-x Deklarationen

4.2.6 Newsreader

Einen wesentlichen Einfluß auf die (technische) Qualität eines Artikels hat die eingesetzte Software, der Newsreader. Ein Newsreader erfüllt mehrere Aufgaben

- Die Abwicklung der protokolltechnischen Aufgaben, also die Kommunikation mit dem Newsserver (siehe Kapitel 3.1).
- Die Aufbereitung der Gruppen und Artikel in eine lokale angenehme Präsentationsform (siehe Abbildung 2.1).
- Eine leicht verständliche und angenehme Benutzerführung zum Navigieren und Editieren neuer Artikel.
- Die ordnungsgemäße Codierung neuer Artikel (siehe Kapitel 4.2.5).

Die verwendete Software verewigt sich hauptsächlich in den Headern `User-Agent`, `X-Mailer` und `X-Newsreader` mit Namen und Versionsnummer. Der Autor dieser Zeilen verwendet beispielsweise derzeit
`User-Agent: tin/1.6.2-20030910 ('Pabbay') (UNIX) (Linux/2.4.21 (i686))`

In Abbildung 4.5 findet sich eine Statistik der aktuell im deutschsprachigen Raum benutzten Newsreader. Als Favouriten können die Produkte der Outlook und Mozilla Familie interpretiert werden.

Als Einstieg dient meist ein Produkt der Outlook Familie, da diese bei vielen Rechnern zum Standardlieferumfang gehört und recht einfach bedient werden kann – was die Spitzenposition in der Statistik bewirkt. Leider ist mit Outlook ein erheblicher Konfigurationsaufwand vonnöten um „schöne“ und spezifikationskonforme Artikel zu

produzieren. Fortgeschrittene Benutzer suchen sich bessere Programme[12] nach ihren individuellen Anforderungen aus.

Bedingt durch die Aufmerksamtkeit, die das Webbrowserprodukt Firefox auf die Mozilla Foundation[13] lenkt, finden sich auch immer mehr Freunde des Mozilla Newsclients. In der Abbildung ist ein schleichender, aber stetiger Abwanderungstrend Outlook -> Mozilla erkennbar.

In den Randgruppen Produkten mit wenigen Prozent Marktanteil ist kein langfristiger Trend erkennbar, die Auf- und Abbewegungen sind statistisch nicht signifikant. Anders betrachtet, hat ein motivierter Usenet Teilnehmer *das* Lieblingsprogramm seiner Wahl gefunden, bleibt er auch bei diesem.

4.2.7 X-No-Archive

In Kapitel 3.2.4 wurde bereits auf die Möglichtkeit der Angabe von
`X-No-Archive: yes`
im Artikelkopf hingewiesen.

Rund 7,2% aller Artikel in `de.*` besitzen eine solche Markierung und sollten von „freundlichen“ Archiven dadurch ignoriert werden.

[12] `http://www.thomas-huehn.de/usenet/newsreaderFAQ.txt`
[13] `http://www.mozilla.org/`

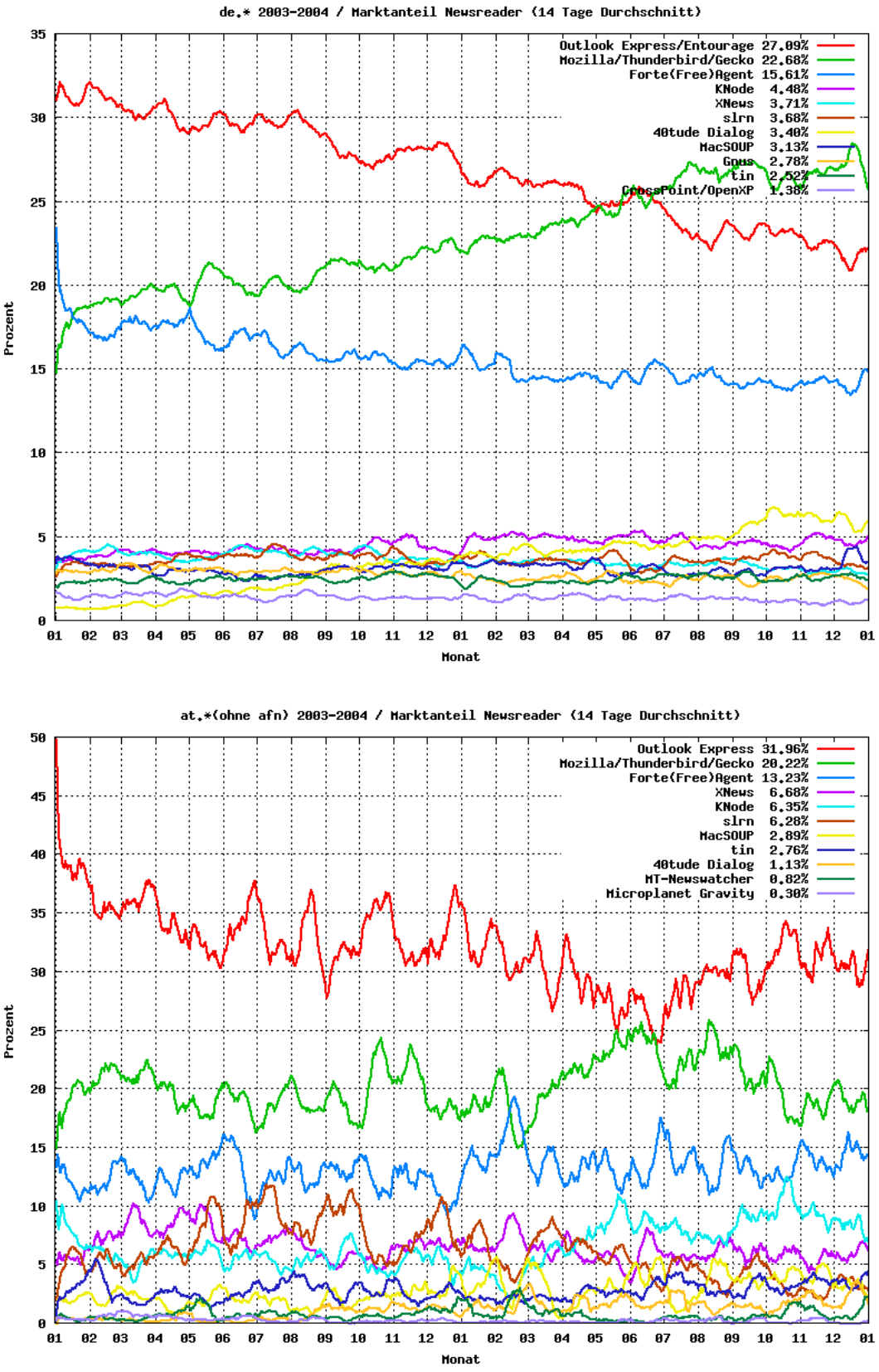

Abbildung 4.5: Statistik Marktanteil Newsreader

5 Charakteristika von Usenet Artikeln

Durch das Vorhandensein eines großen Datenpools lassen sich erste „Eckdaten“ über dessen Inhalt mit einfachen statistischen Methoden ermitteln. Im letzten Kapitel wurden die Daten meist nur einem einfachen Testkriterium gestellt und entsprechend dem Ergebnis in verschiedene Kategorien eingeteilt. Nach fehlerfreiem Durchlauf[1] wurden dann die Daten graphisch veranschaulicht beziehungsweise interpretiert.

In diesem Kapitel werden einzelne Aspekte genauer untersucht. Um ein bestimmtes Charakteristikum – ein „Feature“ – eines Artikels zu extrahieren, erfordert es jeweils den Entwurf eines Algorithmus beziehungsweise einer Implementation.

5.1 Artikelursprung

Für die Verarbeitung eines Artikels durch einen Newsserver stellt der Einspeisepunkt und Routingpfad eines Artikels ein entscheidendes Kriterium dar, er entscheidet über Annahme oder Ignorierung des Artikels mit. Weiters können diese Informationen als ein zusätzliches Informationsstück in komplexeren Analysen dienen. Die Extraktion der gewünschten Informationen ist leider nicht eindeutig, weil sie nicht als strikt definierte Information in einem eigenen Header (wie zum Beispiel die Zeit in `Date`) vorkommt – die Zerlegung des Pfades benötigt einen Schätzalgorithmus.

5.1.1 Path

Bei der Einspeisung eines Artikels (siehe auch Kapitel 3.2.1) bekommt jeder Artikel einen `Path` Eintrag im Header hinzugefügt. Dieser symbolisiert quasi die Reiseroute eines Artikels. Ein Eintrag hat die Form
`Path: irgendwo.server.bla!dorthin.server.at!.....!xyz.at!not-for-mail`
und wird von rechts nach links gelesen und erstellt. Die einzelnen Stationen sind die Server(namen), die durchlaufen worden sind, getrennt durch „!“ Zeichen.

Zu den Anfangszeiten des Usenet, noch vor der Einführung von NNTP und der Vernetzung per Internet, waren `Path` Einträge noch „echte“ Absenderadressen. Um Ver-

[1] Bedingt durch die nicht immer Standard konformen Artikel ist es oft notwendig, mehrmals zu probieren, bis ein Durchlauf wie geplant funktioniert. Bei einer Archivgröße von mehr als 20Gb Text kann auch bei heutigen Rechnerleistungen ein einzelner Durchlauf mehrere Tage dauern.

wechslungen auszuschließen, beginnen Pfade historisch üblicherweise ganz rechts mit einem „not-for-mail“. Links davon verewigt sich als erste Station der Einspeiseserver. Jeder weitere Server trägt sich jeweils links dazu ein und somit ergibt sich ein Pfad vom Ursprung zum aktuellen Server von dem man den Artikel bezogen hat.

5.1.2 Anwendungen

Ein funktionierender `Path` bringt natürlich einmal die offensichtliche statistische Komponente. Bei Auswertung der Pfade läßt sich ein Spinnennetz der Serververbindungen untereinander konstruieren. Dieses dient zum Auffinden von für den Ablauf kritischen Punkten im eigenen Zuliefernetz und kann mit etwas Recherche zum Aufbau neuer (Server)freundschaften und somit mehr Versorgungssicherheit hilfreich sein.

Viel wichtiger jedoch ist der `Path` für die Serversoftware, um festzustellen ob der Artikel schon einmal verarbeitet wurde. Durch die Usenet typische Verteilungsstrategie (siehe Kapitel 3.2.2) sind Schleifen alltäglich, sprich, ein Artikel könnte einem Server von mehreren Partnern angeboten werden.

Bevor ein Server einen Artikel einem Nachbarserver anbietet, prüft er, ob dieser nicht schon im `Path` Eintrag aufscheint. Tut er dies, hat der Artikel diesen Server schon einmal besucht und es wäre sinnlos ihm den Artikel nocheinmal anzubieten. Insbesondere der Fall der direkten Schleife, also wenn man einen Artikel empfangen hat, diesen sofort beim nächsten Exportvorgang dem gleichen Server auch anzubieten, läßt sich somit effizient vermeiden.

Die Anwesenheit eines Servers im `Path` ermöglicht auch lokale Filterungen nach Maß des Administrators.

Eine Variante ist die passive Filterung. Artikel, die aus einer bestimmten Ecke des Netzes kommen, werden einfach ignoriert und/oder nicht weitergereicht. Dies kann zum Beispiel bei bekannten „Spamlöchern“ angewandt werden oder wenn man der Meinung ist, aus dieser Ecke kommen hauptsächlich unnütze Diskussionsbeiträge[2].

Bei der aktiven Filterung geht man noch einen Schritt weiter. Jeder Artikel, ausgehend aus einem bestimmten Netzwerk, wird explizit gelöscht[3].

Man muß jedoch immer bedenken, man ist immer nur ein Teilnehmer eines großen Netzes. Globale Effekte, im positiven wie im negativen Sinn, benötigen die Kooperation mehrerer Knotenpunkte.

[2] Ein Beispiel hierzu ist das Webinterface von Google Groups.
Durch den offenen und anonymen Zugang zum Usenet sammelt sich bei GG ein beträchtlicher Anteil an destruktiven Teilnehmern und technisch ist die produzierte Artikelqualität auch verbesserungswürdig. Manche Usenet Teilnehmer filtern deswegen alles was aus Richtung GG kommt.
`http://www.fh-flensburg.de/wt/usenet/Hinweise_fuer_Google-Poster.txt`

[3] Auch bekannt unter „active [UDP]“ - Usenet Death Penalty

5.1.3 Suche und Reduktion

In der Praxis ist der Fall „Pfadeintrag = echter voller Servername“ leider nur teilweise gegeben. Für die Effizienz des Artikelweiterleitens wäre dies natürlich optimal, aber da der `Path` niemals auf Richtigkeit überprüft wird, kann man noch „beliebige“ Zusatzinformationen in den `Path` einfügen. Ob ein Newsserver seinen eigenen Namen oder den der Artikel einliefernden Verbindung ganz rechts anführt, ist auch nicht definiert.

Zur Auffindung der Einspeiseinformation aus dem `Path` gilt es einen Reduktionsalgorithmus zu konstruieren, der uninteressante Informationen in mehreren Schritten aussiebt, bis nur mehr ein sehr wahrscheinlicher (Server-)Kandidat überbleibt. Der `Path` wird von rechts nach links zu durchlaufen und jeder Eintrag iterativ mit mehreren Regeln durchgetestet.

Folgende Bewertungsregeln wurden angewandt:

- `not-for-mail`
  Dieser Eintrag kann verworfen werden da rein formal verlangt (Kapitel 5.1.1).
- `.POSTED`
  Manche Server markieren bewußt den Einspeisepunkt indem sie ihrem Namenseintrag noch `.POSTED` anhängen. Dies wäre eine schöne Lösung, um immer schnell den richtigen Ursprung zu finden, leider tragen jedoch nur 1,5% der Pfade so eine Markierung. Die beste Behandlung ist somit die simple Entfernung dieses Subteiles und somit Rückfall auf die allgemeine Detektion.
- `.MISMATCH`
  Diese Endung zeigt an, dass die IP Adresse links davon nicht mit dem Eintrag rechts davon verifiziert werden konnte, was meist eine DNS[4] Unstimmigkeit bedeutet, die IP->Name Auflösung stimmt nicht überein. 3,9% der Pfade beinhalten eine solche Markierung und können hier ignoriert werden, reine IP Zahlenadressen sind nicht von Interesse.
- `192.222.212.123`
  Reine IP Adressen bringen keine Information und werden ignoriert.
- `abcxyz`
  Manche Server fügen noch eigene Kennungen aus Buchstaben und Zahlen ein, zum Beispiel Kundennummern. Dies sind keine Reisestationen, alles was nicht einem Servernamen der Form aaa.bbb.ccc entspricht, wird ignoriert.
- `.123.`
  Reine Zahlenkomponenten in einem Servernamen können ignoriert werden. Sie

[4] Das DNS – Domain Name System – ist im Internet zuständig für die Umsetzung von Benutzer freundlichen abc.def.ghi Rechnernamen zu der Rechner freundlichen 192.123.111.222 numerischen Schreibweise.

dienen meist nur zur Unterscheidung von einzelnen Rechnern in einem Rechnerverbund.

- `xxx. ......`
  Sollte der verbliebene Eintrag noch mindestens 3 Teile besitzen, also die Form `xxx.yyy.zzz`, wird der linkeste Teil auf `xxx` gesetzt. Der genaue Rechner ist nicht interessant, sondern das Netzwerk.

5.1.4 Information

Dieser Schätzalgorithmus des Einspeisepunktes verhilft trotz seiner Ungenauigkeit zu brauchbaren Daten woher ein Artikel ungefähr kommt, denn große Usenet Einstiegspunktanbieter setzen auch leicht parsebare Namen. Alle Artikel aus einer bestimmten Netzecke bekommen somit die gleiche Kennung. Ein paar (willkürlich gewählte) Positivbeispiele:

```
xxx.t-online.com, xxx.t-online.de, xxx.t-online.fr
xxx.individual.net, xxx.math.fu-berlin.de, xxx.math.lsa.umich.edu,
xxx.math.tu-berlin.de, xxx.math.uni-bremen.de,
xxx.mathematik.hu-berlin.de, xxx.mathematik.tu-darmstadt.de,
xxx.residenz.uni-wuerzburg.de, xxx.rz.uni-wuerzburg.de,

xxx.adsl-dynamic.inode.at, xxx.adsl.inode.at
xxx.c-gmitte.xdsl-line.inode.at, xxx.c-gragnitz.xdsl-line.inode.at,
xxx.c-gstpeter.xdsl-line.inode.at, xxx.c-whebra.xdsl-line.inode.at
xxx.dialup.wien.inode.at, xxx.dynamic-athome.inode.at,
xxx.dynamic.adsl-line.inode.at, xxx.dynamic.home.xdsl-line.inode.at,
xxx.dynamic.xdsl-line.inode.at, xxx.graz.inode.at,
xxx.inode.at, xxx.sdsl-line.inode.at, xxx.static.adsl-line.inode.at
xxx.static.home.xdsl-line.inode.at, xxx.vie-mc.inode.at,
xxx.xdsl-line.inode.at
```

Die Aufschlüsselung des inode.at Netzes veranschaulicht den erwünschten Effekt der Reduktion. Ein genauer Einspeisepunkt wird nicht ermittelt, die Zahlenkomponenten des Eintrages wurden entweder gefiltert oder sind durch den Löschvorgang des linken Teiles verschwunden. Es bleibt nur die Information aus welchem Teilnetz von inode.at der Artikel stammt – bei Bedarf kann man die Einträge noch weiter reduzieren beziehungsweise zusammenfassen.

Die `Path` Einspeispunkt Information könnte nun mit anderen Daten verknüpft werden, zum Beispiel dem Absender (`From`) eines Artikels und der verwendeten Software (siehe Kapitel 4.2.6) und somit ergäbe sich ein möglicher Abschätzungsmechanismus ob mehrere Artikel wirklich vom selben Autor stammen.

Leider gibt es auch einige Folgen, die ohne kompliziertere Reduktionsalgorithmen nicht erkannt werden:

```
xxx.tnt1.mad3.esp.da.uu.net          xxx.speedway15.dip105.dokom.de
xxx.tnt1.salisbury.md.da.uu.net      xxx.speedway15.dip112.dokom.de
xxx.tnt1.stafford.tx.da.uu.net       xxx.speedway15.dip114.dokom.de
xxx.tnt1.str2.deu.da.uu.net          xxx.speedway15.dip123.dokom.de
xxx.tnt10.ber2.deu.da.uu.net         xxx.speedway15.dip126.dokom.de
```

Diese Serien würde man als Mensch natürlich sofort als zusammengehörig erkennen, ein Algorithmus sieht jedoch keinen Unterschied zu den positiven Beispielen der Form Fachgebiet.Uni.Land Unterteilungen.

5.2 Autor

Wie bereits in Kapitel 2.3 erwähnt, besitzt jeder Artikel einen Absender der Form
`From: Martin Pirker <crf@sbox.tugraz.at>`
Der Inhalt der `From` Angabe ist „beliebig“, der Benutzer kann in der Konfiguration seines Newsreaders die Absenderangabe meist[5] ausfüllen wie er will. Es findet keine Überprüfung statt ob die Angaben irgendwie mit realen Werten übereinstimmen. Dennoch ermöglichen die Angaben im Absender Gruppenverhaltensprofilauswertungen.

5.2.1 E-mail Teil

Bedingt durch die Spamproblematik ist der E-Mail Adressteil kaum mehr verwertbar, denn es ist ein Leichtes, automatisiert alle Artikel auf einem Newsserver zu durchsuchen und an alle Einträge im `From` Feld Werbemüll abzuladen.

Im Laufe der Zeit sah man an den wechselnden Absendern der regulären Usenet Teilnehmer wie verschiedene Strategien ersonnen wurden, um dem Problem zu begegnen:

- Kompletter Verzicht
  Die ursprüngliche Funktion des E-Mail Absenders, die schnelle Kontaktaufnahme mit dem Autor eines Artikels, wird verworfen, als Absender wird eine falsche E-Mail Adresse angegeben. Idealerweise ist dies eine Standard konforme, die mit `.invalid` endet und von allen Programmen als nicht zustellbar erkannt wird. Manchmal sind es aber auch komplette Fantasieadressen – was eine Einstellung von Ignoranz dokumentiert, wo der Spam schlußendlich landet...

[5] In seltenen Fällen gibt es Arbeitsumgebungen, wo der Absender durch den Sysadministrator festgelegt wird und der Newsreader konfiguriert ist, keine Änderungen durch den Benutzer zu erlauben. Dieser Fall ist aber die Ausnahme.

- Verschlüsselte Adressen
  Bei dieser Taktik ist die richtige E-Mail Adresse zwar angegeben, nur immer mit einem Trick leicht verunstaltet. Sie muß also vor Verwendung wieder korrigiert werden. Dies kann ein einfaches Entfernen von Komponenten wie `.nospam.` oder `.loeschmich.` sein, oder teilweise Entschlüsselung mit „ROT13“[6] oder ähnlichen Transformationen

- Müllpostfach
  Ein E-Mail Postfach auf einem Freemail Dienst dient als Müllkippe. Ab und zu überprüft man den Inhalt, wird die Spam Flut zu groß, holt man sich einfach ein neues Postfach.

- Periodisch wechselnde Adressen
  Betreibt man einen eigenen Server, kann man sich leichter spezielle Empfangsadressen anlegen. Man verwendet als Adresse z.B. `news-2005-06@mydomain.at`. Es gibt eine Jahres- und Monatskomponente, die sich natürlich monatlich ändert. Der Mailserver nimmt immer nur Mails an, die relativ „jung“ sind, alte Adressen werden gesperrt. Die Adressen bleiben relativ frisch und man bekommt weniger Spam.

Die E-Mail Adresse eignet sich somit nicht zur Identifikation einer Person. Über einen Zeitraum von 2 Jahren ist ein Adresswechsel sehr wahrscheinlich, welcher mit automatisierten Mitteln nur aufwendig verfolgt werden kann – womit aber das ursprüngliche Ziel, den Spammern die Arbeit zu erschweren, aber erreicht ist.

5.2.2 Namensteil

Bei der `From` Namenskomponente kann eine höhere Langzeitstabilität angenommen werden. Es liegt in der Natur des Menschen der Wunsch, als Individuum wahrgenommen zu werden. Wenn man einen Artikel verfasst, diesen dem weltweiten Usenet zur Veröffentlichung übergibt, möchte man (meist) als Urheber mit diesem assoziiert werden und nicht im Artikelmeer als „noch eine Meinung“ untergehen. Eine Diskussion kann sich erst entwickeln, wenn die einzelnen Teilnehmer wahrgenommen werden können und man im Geiste die einzelnen Meinungen an Personen – Namensidentitäten – festmachen kann.

Diese Identität ist idealerweise in Übereinstimmung mit den gewohnten Erfahrungen im realen Leben ein „Vorname Nachname“ Konstrukt. Im Netz gehen die Meinungen

[6] ROT13 rotiert einfach einen Buchstaben 13 Positionen im Alphabet weiter. Eine 2-fache Anwendung von ROT13 liefert also wieder das Ursprungsergebnis.
`http://en.wikipedia.org/wiki/Rot13`

darüber auseinander was ein „vernünftiger Name“[7] ist. Von der Perspektive der Datenverarbeitung ist es natürlich besser, mindestens 2 Komponenten zur Verfügung zu haben und nicht zum Beispiel ein einfaches “Martin“ oder „Klaus“, da hier doch ein Verwechslungspotential gegeben ist.

5.2.3 Gruppenprofil

Die Auswertung der Namensangaben im `From` eines Artikels ermöglichen die Erstellung eines „Kundschaft“ Profils je Gruppe. Es gibt verschiedene Formen der Bindung zu einer Gruppe beziehungsweise zu einem Gruppenthema:

- Manche Gruppen werden als reine Dienstleistungsgruppen wahrgenommen: Man hat ein Anliegen, man trägt es vor, es wird (vielleicht) geholfen, man geht wieder getrennte Wege.
- Andere Gruppen bieten mehr eine lose Gemeinschaft: Man liest relativ regelmäßig mit und ab und zu nimmt man auch an einer Diskussion teil.
- Es gibt aber auch echte Gemeinschaften: Man kennt sich schon lange, man hat ein Themengebiet, das sich quasi nicht verbraucht, es gibt immer etwas zu diskutieren.

Die Datenbasis umfaßt 24 Monate. Frage ist, in wievielen dieser 24 Monate hat eine bestimmte Person (=eine bestimmte `From` Identität) mindestens einen Artikel geschrieben. Stammschreiber mit großer Bindung zu einer Gruppe bringen es auf bis zu 24 Monate, kurze Fragesteller nur auf 1 Monat. Dazwischen ergibt sich ein breites Spektrum. Monat für Monat läßt sich errechnen wie anteilig in einer Gruppe die einzelnen Typen vertreten sind.

In Abbildung 5.1 finden sich 4 Beispiele von solch ermittelten Gruppenautorenprofilen. Auf der x-Achse findet sich (wie gewohnt) die Zeit, auf der y-Achse die Gesamtheit der Artikel in diesem Monat (von 0% bis 100% dargestellt). Die 24 Linien stellen die Monate an Aktivität dar, als unterstes ist die 1 Monat Aktivitätslinie, ganz oben bei 100% (immer) die 24 von 24 Monate aktiv Linie. Die Monate 6, 12, 18 und 24 sind zur besseren Unterscheidung etwas dicker hervorgehoben.

Dies mag auf den ersten Blick etwas verwirrend wirken, somit zur besseren Illustration eine kurze Interpretation der Graphen:

[7] `http://www.realname-diskussion.info/`

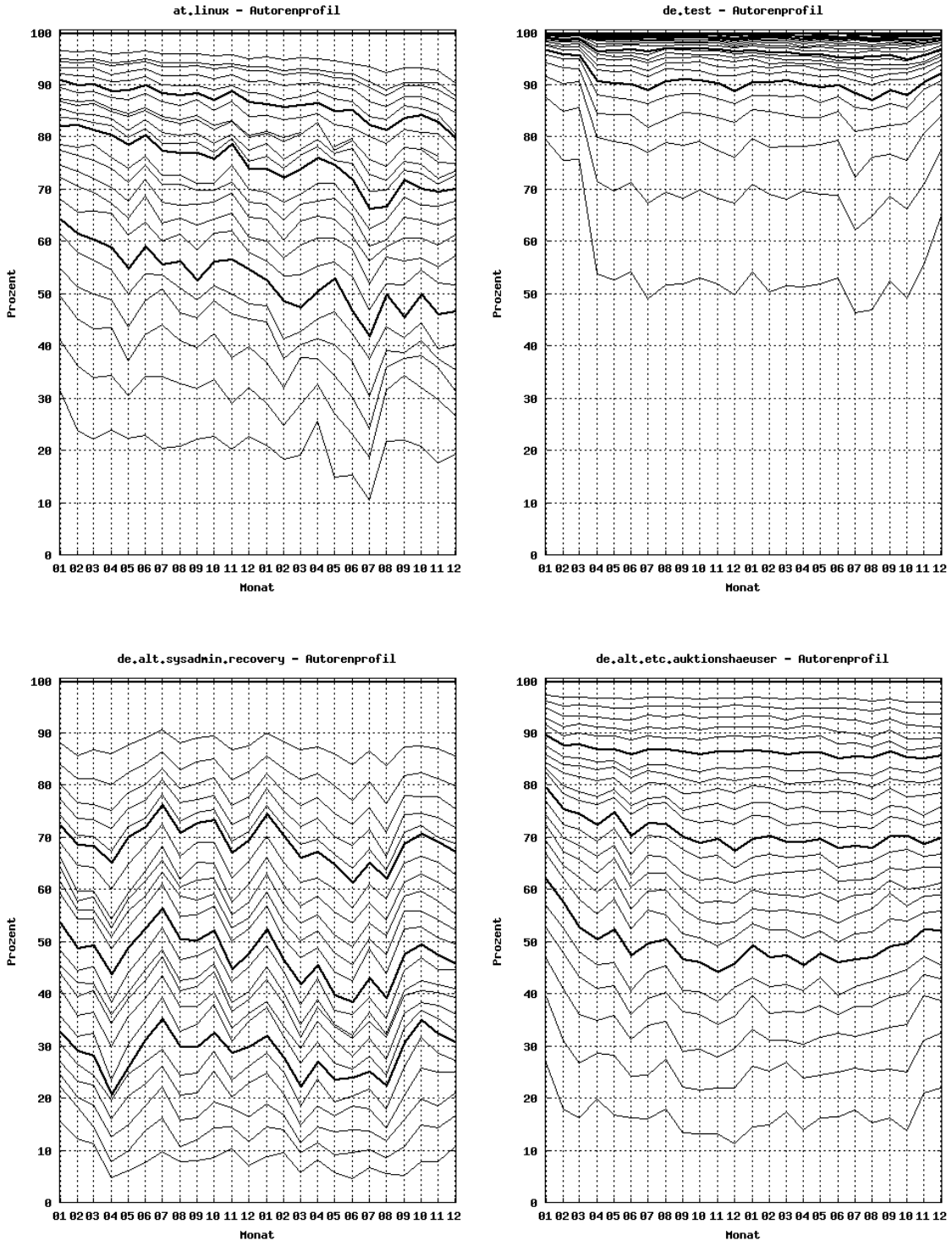

Abbildung 5.1: Autorenzusammensetzung in ausgewählten Gruppen

`at.linux`: Die 1-er Linie liegt die meiste Zeit bei 20%. Das bedeutet, jeden Monat sind ca. 20% der Artikelschreiber „Laufkundschaft“, d.h. sind nur in diesem einen Monat aktiv und sonst nie. Am oberen Ende des Graphen liegen die Linien mit den hohen Werten eng beieinander, d.h. es gibt zwar Teilnehmer, die über die ganzen 2 Jahre quasi immer aktiv waren, diese machen jedoch nur einen einstelligen Prozentanteil der Artikelschreiber je Monat aus. Global könnte man den Trend als leicht fallend interpretieren, d.h. die regulären Teilnehmer nehmen zu.

`de.test`: Die Testgruppe der `de.*` Hierachie ist natürlich ein Extremfall. 50% der Autoren sind Einzeltäter, als mehrfach Schreiber kommen wohl nur regelmäßige Programmtests in Frage.

`de.alt.sysadmin.recovery`: Diese Gruppe zeichnet sich durch eine hohe Bindung aus. Der Anteil der durchgehend aktiven Schreiber über die ganzen 2 Jahre ist über 10%, umgekehrt hat die Gruppe kaum Kurzzeitbesucher.

`de.alt.etc.auktionshaeuser`: Ein Beispiel für eine stabile Gruppe, das Profil des Publikums bleibt über lange Zeit quasi gleich, eine Balance zwischen neu Hilfesuchenden und abwandernden Profis ist gegeben.

5.3 Quotings

Eine Diskussion ist eine Abfolge von Artikeln, Antworten nehmen Bezug auf einen vorhergehenden Artikel. Ein „Quoting“ ist ein von einem Vorgängerartikel übernommener Textteil, ein Zitat, welches in einen neuen eigenen (Antwort-)Artikel eingefügt und stellenweise kommentiert wird.

Dieser Zerlegungs- und wieder Zusammenfügeprozeß ermöglicht einen bequemen, fließenden Übergang der Meinungen. Für die Datenanalyse gilt es jedoch einen Algorithmus zu finden, der die übernommenen von den neuen Textpassagen trennt.

5.3.1 Bezug nehmen

Im Bereich E-Mail hat sich eine Kultur des [Top-posting] etabliert. Bei einer Replik wird der alte Text am Ende komplett stehen gelassen und neuer oben am Anfang eingefügt. Im Bereich des Usenet herrschte von Anfang an eine andere Replikkultur, „bottom-posting“ und selektives auswählen, editieren und antworten hat bis heute wesentlich deutlicheren Bestand.

Usenet Artikel besitzen via `Message-ID` und `References` (siehe Kapitel 2.3) eine natürliche Ordnungsstruktur die Teilnehmer einer Diskussion[8] als zusammengehörig kennzeichnet. Ein Newsserver hält ein paar Tage alte Artikel online vor. Will man einen

[8] eines „Thread“

älteren Artikel in einer Diskussion nocheinmal lesen, ist es kein Problem diesen nachzuschlagen – es ist nicht nötig wie bei E-Mail den gesamten Austausch immer (fast) komplett am Ende mitzuschleppen.

Im Prinzip kann man im Usenet davon ausgehen, dass es keinen bestimmten Antwortstil gibt. Es kann Vollzitate geben, am Anfang, am Ende, Teilzitate, Durchmischungen, etc[9]. Dies bedeutet für die Datenanalyse jedoch eine Schwierigkeit, wie detektiert man in einem Artikel den Anteil an neuem versus altem Text?

5.3.2 Levenshtein Distanz

Die Levenshtein Distanz[10](LD) versucht eine Quantifizierng der Ähnlichkeit zweier Textteile. Sie errechnet die minimale Anzahl an Operationen (einfügen, löschen, ersetzen) an Zeichen, um einen Textteil in einen anderen zu transformieren.

Beispielsweise hat eine Transformation
„Vater“=>„Mutter“
eine Levenshtein Distanz von 3: 2 Zeichen werden am Anfang ersetzt, 1 Zeichen („t“) wird eingefügt.

Als Erweiterung könnte man die 3 Operationen noch verschieden stark gewichten, oder, für unsere Anwendung wichtiger, je nachdem ob das Ergebnis länger oder kürzer als das Original ist, das Vorzeichen positiv oder negativ setzen.

5.3.3 Implementation

Mit Hilfe der LD lassen sich einzelne Zeilen im neuen und alten Text vergleichen und somit ein Algorithmus zur Detektion von Zitierungen implementieren. Die meisten Artikel werden mit Zeilenlängen von ca. 72 Zeichen erstellt. Dies ist historisch bedingt und stellt einen Kompromiss zwischen simpler Lesbarkeit, einer Terminalzeilenlänge von 80 Zeichen und 80-72=8 Zeichen Reservierung für Zitierkennzeichnungen dar.

Ein Usenet übliches Zitat hat die Form:

```
>>> Dies ist eine Meinung.
>> Hier steht wiederum eine andere.
> noch eine Gegenmeinung.
Neuer Text dieses Artikels.
```

Das übliche Zitierkennzeichen ist ein > und wird am Anfang einer vom Vorgängertext übernommenen Zeile eingefügt. Werden Zeilen mehrmals übernommen, ergibt sich da-

[9] `http://einklich.net/usenet/zitier.htm` bei [Volker]
[10] Benannt nach Vladimir Levenshtein anno 1965, siehe [Levenshtein]

durch natürlich eine mehrfache Einrückung, wie oben gezeigt. Aufgabe ist es nun, nur die Zeilen mit neuem Text (=neuer Information), oben also nur die letzte Zeile, zu detektieren.

Vorschnell könnte man annehmen, die Aufgabe wäre mit der Detektion von einem > am Anfang plus identischem Zeilenrest erledigt. In der Praxis sind jedoch noch folgende „Randbedingungen“ in der Implementation zu berücksichtigen:

- Ordnung
  Die Zeilen können (und werden auch) in beliebiger Reihenfolge übernommen.
- Zitierzeichen
  Es wird nicht nur > verwendet, sondern zum Beispiel für Zitate aus externen Quellen oft ein |, von Leuten die einfach nur anders sein wollen ein :, oder ein <, etc. Ein Algorithmus sollte bezüglich Zitatzeichen tolerant sein. Weiters wird manchmal zusätzlich ein Leerzeichen eingefügt, somit ergibt sich dann für 3 Zitierebenen „> > > “ statt „>>>“
- Zeichensatzfehler
  Falsche Content-Type Angaben (siehe Kapitel 4.2.5) bewirken Dekodierungsfehler im Textteil. Dies zeigt sich sowohl als Einzelzeichenfehler, als auch als Doppelzeichenfehler wenn zum Beispiel ein ISO <-> UTF Zeichensatzwechsel nicht korrekt durchgeführt wurde. Für die Ähnlichkeitsdetektion von Zeilen bedeutet dies einen höheren Unsicherheitsfaktor.
- Kammquoting
  Ein bekannter Fehler eines verbreiteten Newsreaderprogrammes ist die strikte Einhaltung der maximalen Länge von ca. 80 Zeichen pro Zeile. Alles was länger ist, wird umgebrochen. Dies ist für neuen Text durchaus sinnvoll, bei zitierten Textpassagen führt dies jedoch zu entstellten Kammformen des Textes. Findet man für eine Zeile keine passende Zitierung, könnte evt. dieser Sonderfall eingetreten sein. Die Abhilfe ist, 2 aufeinanderfolgende Zeilen jeweils zusammenzufassen und nocheinmal zu testen.
- Signatur
  Viele Leute signieren ihre Artikel nicht nur mit ihrem Namen, sondern auch mit einem weisen Spruch, Hinweisen auf Internetseiten oder anderen Kleinigkeiten. Signaturen sind zwar immer neuer Text, können aber meist ignoriert werden, da sie ja jedem Artikel eines Autors angehängt werden. Die übliche Trennkennzeichnung ist „— “, alles was danach kommt ist Signatur und kann ignoriert werden.

Der Suche per LD funktioniert nicht bei

- Namensquotings
  Selten werden Newsreader so konfiguriert, dass Zitate mit Namen vorangestellt werden, also zum Beispiel
  `Martin> bla bla bla`
  `Martin> etc. etc. etc.`
  Dies ermöglicht für den Leser zwar eine leichtere Zuordnung wer den Text geschrieben hat, aber mehrere Zitatebenen werden zu riesigen Einrückungen und die Zitate schwer detektierbar.

- Format=Flowed
  Dem Problem der Absatzkennzeichnung im Vergleich zur Zeilenumbruchmarkierung in reinen Textartikeln wurde versucht, in [RFC2646] zu begegnen. Theoretisch eine annehmbare Lösung, praktisch haben jedoch mehr als 5 Jahre nach diesem Standardisierungsvorschlag erst 16,2% der Artikel eine F=f Markierung. Dies ist zu wenig, um einen signifikanten Effekt (schöner Zeilenumbruch bei mehreren Zitierebenen) zu bewirken, F=f Artikel werden bezüglich Zeilenumbruch meist wie normale Artikel behandelt.

5.3.4 Information

Nach Abschätzung der Artikelgrößen und Algorithmuslaufzeiten wurde das Größenlimit der Artikel, die sich zur Anwendung der Zitatentfernung eignen, auf 10 kb festgelegt. 99,99% aller Artikel sind kleiner als 10 kb. Große Artikel entstehen hauptsächlich durch die Verwendung von HTML oder angefügter (ASCII codierter) Bilder. Bedingt durch die Nichtlinearität der Algorithmuslaufzeit ist ein Größenlimit unbedingt erforderlich.

Weiters wurde die Levenshtein Distanz für die Zeilenähnlichkeit mit 3 festgelegt. Dies erlaubt 1-2 Zeichen für eine Quotingebene und 1-2 Zeichen für Codierungsfehler.

Die Zitatdetektion und -entfernung, angewandt auf die Artikeltexte ergibt eine Verringerung der durchschnittlichen Artikelgröße in `de.*` von 1020 auf 608 Bytes. – siehe Abbildung 5.2.

Der Datenbestand in `de.*` reduziert sich gesamt von 7861 Mb auf 4684 Mb, also ca. 40% des Artikeltextes sind übernommene Textzeilen von dem Artikel auf den man antwortet. 30742 Artikel sind größer 10 kb und belegen 854 Mb, ein knappes Fünftel der Restgröße (nach der Zitatentfernung).

Bemerkenswert ist nicht nur die geringe Größe des Artikeltextes, sondern auch der Vergleich mit der Headergröße. Der Header enthält die administrativen Informationen (siehe auch Kapitel 2.3) und diese sind in ca. 80% der Artikel größer als der eigentliche Haupttext – rechnet man die Zitate weg sogar in ca. 95% der Fälle.

Usenet kann somit als ein schnelles Medium betrachtet werden. Es werden keine langen Romane geschrieben, Antworten übernehmen einfach bis zur Hälfte des Textes des

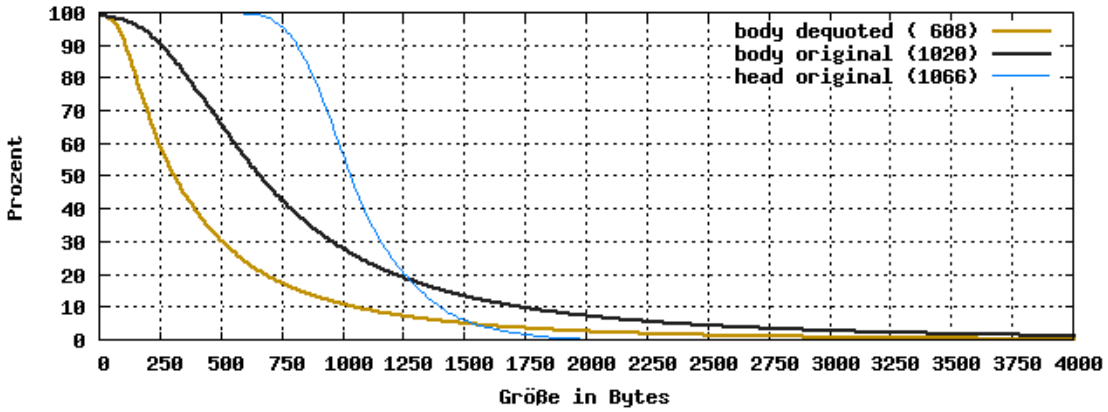

Abbildung 5.2: `de.*` – Prozent der Header/Bodys kleiner x Bytes

vorhergehenden Artikels, die eigene Meinung wird hinzugefügt und schon geht der neue Artikel wieder ins Netz.

Eine Diskussion für `at.*` findet sich in Anhang B.3

5.4 Threading

Wie bereits im vorhergehenden Kapitel angedeutet, besitzen die Artikel im Usenet eine Abhängigkeitsstruktur. Stillschweigend wurde bei der Quoting Analyse angenommen, dass diese Struktur bereits bekannt ist, d.h. es ist immer eindeutig, welcher Artikel auf welchen folgt.

In der Praxis beschränken sich Lösungansätze zur Berechnung dieser Strukturen meist auf den statischen Fall und/oder auf „kleine“ Datenmengen ([Jwz02]), kleine Fehler und nicht lineares Laufzeitverhalten stellen kein Problem dar. Im Falle eines Archives von mehreren Millionen Artikeln interessiert bei einem (Re-)Konstruktionsalgorithmus die Robustheit gegenüber allen möglichen Datenfehlern, eine noch annehmbare Laufzeit und die Korrektheit der Ergebnisse auch bei länger laufenden Diskussionen – bei einem gleichzeitig „weiterziehenden Fenster“ in den Datenpool. Eine mögliche Implementation wird hier vorgestellt.

5.4.1 Referenzen

Dass jeder Artikel eine eindeutige `Message-ID` (MID) besitzt, wurde in mehreren vorhergehenden Kapiteln bereits erwähnt. Der Startartikel einer Diskussion – eines Threads – besitzt keine extra Informationen. Alle weiteren Beiträge in einem Thread

sind immer Repliken auf einen bestehenden Artikel. Um eine Abhängigkeitsstruktur zu rekonstruieren, müssen die Vorgängerartikel irgendwie im Header der Replik vermerkt werden.

Der Eintrag `In-Reply-To` stellt die einfachste Informationsquelle dar und enthält, wie bereits der Name vermuten lässt, die MID des Artikels auf den geantwortet wurde. `In-Reply-To` kommt aber ursprünglich aus dem E-Mail Bereich und tauchte erst relativ spät in Usenet Artikeln auf. Heute besitzen erst ca. 11,7% der Artikel in `de.*` diesen Eintrag, was keine große Hilfe für das Threading Problem bedeutet.

Im Bereich Usenet von zentraler Bedeutung ist der `References` Header Eintrag. Von Anfang an war die Definition im Usenet Bereich für `References` sehr konkret:
`References` enthält eine Liste von MID der Vorgängerartikel dieses Artikels. Die MID sind geordnet, der erste Eintrag ist der älteste, der letzte der jüngste – oder anders formuliert: Bei jedem neuen Artikel ist die `References` Zeile der Inhalt der `References` des Vorgängers plus dessen MID angefügt.

5.4.2 Praxiswerte

Im realen Betrieb generieren Programme trotz spezifizierter Formate nicht immer so saubere Daten wie man das erwartet. Es gilt somit auch folgende `References` Eigenheiten zu berücksichtigen:

- Reihenfolge:
  Die Reihenfolge der MID in den `References` muß nicht unbedingt schrittweise älteste zu neuester sein.

- vollständige Artikelkette:
  Die MID Liste ist nicht immer vollständig vom Startartikel bis zum aktuellen Artikel. Manche Newsreader haben entweder Probleme, sehr lange Headerzeilen spezifikationsgemäß in mehrere kürzere Zeilen zu falten[11] oder detektieren nicht korrekt, wann das maximale Zeilenlimit erreicht ist (und damit der Newsserver die Annahme des neuen Artikels verweigert).

  Das Ergebnis ist in beiden Fällen gleich, es kann nicht mehr die ganze Kette mitübernommen werden, sondern sie muß gekürzt werden, bis sie wieder verarbeitbar ist. Wie die Kürzung erfolgt, ob nur an einem Ende gestrichen wird oder „zufällig“ einzelne MID entfernt werden, ist nirgends festgelegt.

- `Message-ID` Schleifen
  Die Daten in `der Message-ID` Kette können fehlerhafterweise oder absichtlich erzeugt nicht kreisfrei sein. Ein Beispiel aus 2 Artikeln , aufgefunden in `de.test`:

[11] [RFC2822] Kapitel 2.2.3

Artikel 1:
`Message-ID: <gfj5934zd.i3Oqkrz@hjlipp.my-fqdn.de>`
`References: <gfj5934zd.i3Oqkrz@hjlipp.my-fqdn.de>`
Artikel 2:
`Message-ID: <gfj5934zd.i3Oqkrz@hjlipp.my-fqdn.de>`
`References: <gfj5934zd.i3Oqkrz@hjlipp.my-fqdn.de>`

5.4.3 Algorithmus

Ein Threading Algorithmus sollte in der Lage sein, den Abhängigkeitsbaum an allen Endpunkten des Usenets soweit nur möglich in identer Form zu rekonstruieren. Dies erfordert eine Robustheit gegenüber der Einlaufreihenfolge der Artikel und fehlender Artikel. Ein einfacher Test zur Validierung ist, eine Teilmenge Artikel aus einem größeren Datenpool mehrmals in beliebiger Reihenfolge aufbauen zu lassen, der Ergebnisbaum am Ende sollte jeweils ident sein.

Es wurde ein Algorithmus entworfen, der beliebiges Hinzufügen aus einem Artikelstrom und wiederum Entfernen (`expire`) unterstützt und robust immer den gleichen Baum liefert. Für diese Aufgabe hält das Threadingdatenobjekt folgende Datenpunkte:

- Zeit
  Den Artikel eigenen `Date` Zeitstempel. Dieser wird benötigt um bei Ausgabe des fertigen Baumes eine Reihung von Artikeln gleicher Tiefe vorzunehmen. Üblicherweise sortieren Newsreader gleich aufsteigend, das Ergebnis ist somit optisch für den Leser überall gleich.

- Wurzelkennung
  Es gibt immer einen Startartikel eines Threads, den Wurzelartikel. Wenn ein Verweis auf diese Wurzel bei Aufbau von den Artikelketten immer mitgeschrieben wird, lassen sich alle Mitglieder eines Threads im Speicher leicht identifizieren. Dies ist im Falle von sehr großen und lange laufenden Threads interessant. Es ist mitunter nicht möglich den gesamten Thread im Speicher zu halten, die ältesten Artikel (und somit auch die Wurzel) wurden bereits entfernt und es existieren nurmehr Teilthreads im Speicher. Anhand der Wurzelkennung können diese Teilthreads als zusammengehörig identifiziert werden und gemeinsam dargestellt werden, einem „ausfransen“ der Threads an der expire Grenze kann entgegengewirkt werden.

- Vorgänger
  Jeder Artikel besitzt einen Zeiger auf einen Vorgängerartikel. Dieser repräsentiert jeweils den aktuell bestmöglichen Schätzwert des unmittelbaren Vorgängers. Dieser kann sich im Laufe der Zeit noch verbessern, bis hin zum „echten“, also dem,

der auch in einem `In-Reply-To` Header stehen würde. Ein Wurzelartikel besitzt keine `References` und somit auch niemals einen Vorgänger.

- Gruppen
  Eine Liste der Gruppen, in denen der Artikel aufscheint. Threads können in einer Gruppe gestartet werden und im weiteren Verlauf in eine andere Gruppe wechseln. Ein Vergleich der Gruppenliste zweier verlinkter Artikel gibt Aufschluss über einen Gruppenwechsel.

- Kinder
  Jeder Artikel kennt alle auf sich bezogenen Repliken. Wird der Artikel selbst entfernt, gilt es, alle seine Kinder darauf hinzuweisen, sich einen neuen Vorgänger zu suchen.

- References
  Die Liste der im Artikelheader deklarierten Vorgänger. Wie bereits in den Kapiteln 4.1.2 und 4.2.4 gezeigt, ist es sinnvoll, für die Zwischenspeicherung die MIDs als Hashwerte abzulegen. (Hashkollisionen sind nach wie vor möglich, aber das Risiko ist geringer, da der Artikelpool und Zeitbereich kleiner sind).

- Message-ID
  Die eigene `Message-ID`.

Eine zusätzliche Bewertung des `Subject` zur Zusammenfassung von Threads, wie manche Newsreader es praktizieren, wird nicht vorgenommen. Es kommt öfters ungewollt vor, dass sich 2 `Subject` gleichen[12], andererseits mutiert ein `Subject` innerhalb eines Threads viel zu schnell[13]

Die 2 Hauptoperationen, einfügen und entfernen von Artikeln, unter Zuhilfenahme der generierten Extrainformationen, seien hier kurz beschrieben:

Artikel hinzufügen

Sind keine aktiven Vorgänger vorhanden, weil es sowieso ein Wurzelartikel ist oder der Verbleib der referenzierten Artikel ungeklärt ist, tritt der einfachste Fall ein, der neue Artikel wird als neue Wurzel in der jeweiligen Gruppe eingetragen.

Finden sich ein oder mehrere aktive Vorgängerartikel, gilt es festzustellen, welcher der näheste Vorgänger ist. Dieser hat als einziger die Eigenschaft, nicht auf den `References`

[12] Zum Beispiel in einer Anzeigengruppe kann sich leicht 2 mal „S: Nokia xyz“ innerhalb eines kurzen Zeitraumes ergeben.

[13] Ein bekanntes Problem ist zum Beispiel die Übernahme des `Subject` aus dem XOVER overview des NNTP Protokolls und nicht direkt aus dem Artikel, was bei verschiedenen Newsreadern zu leichten Abweichungen im `Subject` führt.

Listen der anderen Kandidaten aufzutauchen. Es werden nun iterativ die Kandidaten gegenseitig solange gestrichen, bis nur mehr der eine gesuchte Kandidat übrig bleiben muß. Bleibt nach einer vollen Runde über alle Kandidaten mehr als ein Kandidat übrig, dann müssen die Referenzen fehlerhaft sein – es wird einfach ein beliebiger der übriggeblieben genommen.

Mit dem bekannten Vorgänger ist es jetzt möglich, sich in die Threadkette einzufügen und den Wurzelverweis zu übernehmen. Der Vorgänger Link wird gesetzt und die Kinder des Vorgängers werden gebeten ihre Vorgängerbestimmung erneut durchzuführen. Es könnte sich der neue Artikel ja unmittelbar vor ihnen eingefügt haben.

Als Nebenschauplatz sind bei neuen Linksetzungen die Gruppengrenzen ebenfalls zu überprüfen und gegebenfalls die Wurzellisten in den jeweiligen Gruppen anzupassen.

Artikel entfernen

Hat ein Artikel keine Kinder, kann er sich einfach selbst entfernen und in der Kinderliste seines Vorgängers austragen.

Bei vorhandenen eigenen Kindern müssen diese ihren Vorgänger neu berechnen.

Gruppenkorrekturen erfolgen analog wie beim Hinzufügen.

5.4.4 Information

Nach Durchlauf der Artikelbasis lässt sich der Artikelbaum der Threads rekonstruieren. Artikel aus dem gleichen Thread besitzen den gleichen Hashwert wie der Wurzelartikel. Die Tiefe im Baum wurde berechnet, die MID als Identifizierung und ein Zeitstempel (im Unix Sekunden Format).

```
THREADROOTHASH LEVEL  MID                                          TIMESTAMP
-------------------------------------------------------------------------------------------
52b32dd7cb86  0 /+-> <eka0va.ru2.ln@c50s19h4.upc.chello.no> 1041482688

bf35a72124c4  0 /+-> <f7RQ9.193990$qq5.2161414@news.chello.at> 1041489483
bf35a72124c4  1    +-> <av16vn$b0vca$1@ID-5356.news.dfncis.de> 1041506103
bf35a72124c4  1    `-> <av2f9t$bghfi$3@ID-103452.news.dfncis.de> 1041547389

12128afa9ad0  0 /+-> <3e140b73$0$13674$3b214f66@news.univie.ac.at> 1041501191
12128afa9ad0  1    +-> <3e1416c8$0$24290$91cee783@newsreader02.highway.telekom.at> 1041504082
12128afa9ad0  2    | +-> <3e141d5e$0$13160$3b214f66@news.univie.ac.at> 1041505782
12128afa9ad0  2    | +-> <3E141EC7.3020004@itx.at> 1041505991
12128afa9ad0  3    | | `-> <3e142708$0$13716$91cee783@newsreader02.highway.telekom.at> 1041508242
12128afa9ad0  2    | `-> <ajj1va.ng8.ln@window.dhis.org> 1041519018
12128afa9ad0  3    |   `-> <s41ee-s64.ln1@lisa.homeunix.net> 1041519996
12128afa9ad0  4    |     +-> <slrnb18lve.3ar.krennmair@webdynamite.com> 1041520625
12128afa9ad0  4    |     `-> <b3e2va.688.ln@window.dhis.org> 1041546155
12128afa9ad0  5    |       `-> <1LhR9.222600$qq5.2513207@news.chello.at> 1041606717
12128afa9ad0  1    +-> <av1jin$b2as6$1@ID-126186.news.dfncis.de> 1041519001
12128afa9ad0  1    `-> <slrnb1aoeq.7km.mabu@black.buntstift.at> 1041588698
```

```
THREADROOTHASH LEVEL  MID                                          TIMESTAMP
-------------------------------------------------------------------------------------------
6b55f2a89c97  0 /+-> <g2r81vocobt9415p0iotj2g312d8rfcegq@4ax.com> 1041526220
6b55f2a89c97  1    `-> <av2f27$bghfi$2@ID-103452.news.dfncis.de> 1041547144
6b55f2a89c97  2      `-> <4j4b1v85dghmmu0vb2fb84ru7k1pj8d8ri@4ax.com> 1041601284
6b55f2a89c97  3        `-> <q654va.paq.ln@window.dhis.org> 1041602586
6b55f2a89c97  4          `-> <mbfb1vk12av618jmaeokuelke150s6khdc@4ax.com> 1041612296
6b55f2a89c97  5            +-> <av4khg$c4ojh$1@ID-111736.news.dfncis.de> 1041618289
6b55f2a89c97  6            | `-> <n6jh1vcif84gi20ndsd8ufima066tc89nj@4ax.com> 1041813046
6b55f2a89c97  7            |   `-> <2okh1vkgp8b647ejhtnj3jpj23l9l752hv@4ax.com> 1041814432
6b55f2a89c97  5            `-> <av5fi8$c6rj1$1@ID-157514.news.dfncis.de> 1041646262
6b55f2a89c97  6              +-> <av5i3g$c6lb8$2@ID-157514.news.dfncis.de> 1041648863
6b55f2a89c97  7              | `-> <inih1vgd40lr2e7i63lqie6hf9fge5c503@4ax.com> 1041812323
6b55f2a89c97  8              |   `-> <uryyb1hv6b.u4.hfrarg@jeyru.de> 1041824971
6b55f2a89c97  6              `-> <78ih1v4gp8hr4aj2bqgtuegbsb80nssbg9@4ax.com> 1041811843

35c64e274a85  0 /+-> <sgr81vc4nv1vitsfihvf4p0p9omo7oh1c3@4ax.com> 1041526301
```

Für Berechnungen, wie zum Beispiel beim Entfernen der Zitate in Kapitel 5.3 ist nur der unmittelbare Vorgänger interessant, es bietet sich ein simples Speicherformat via der Hashindexwerte (siehe Kapitel 4.1.2) an:

```
ARTICLE                 LEVEL PREDECESSOR
-----------------------------------------------------
20030101004731x313defad52 0 -
20030101004957xac90f56b95 1 20030101004731x313defad52
20030101010808x24097873c4 2 20030101004957xac90f56b95
20030101011006xd5f27b7bae 2 20030101004957xac90f56b95
20030101095653x818fc045b6 1 20030101004731x313defad52
20030101011637xffb53eee05 0 -
20030101022406x4cf5630307 0 -
20030101012811xbb97fc60aa 1 20030101022406x4cf5630307
20030101131123x7b1663c989 2 20030101012811xbb97fc60aa
20030101122042xb26e168617 3 20030101131123x7b1663c989
20030101224432xf163c6fbec 4 20030101122042xb26e168617
20030101115219xc764ce3964 0 -
```

Den Gesamteindruck der errechneten Threadtiefen über die Hierachien veranschaulicht Abbildung 5.3:

45 bis 65% der Artikel der Diskussionen kommen nicht über Tiefe 5 hinaus. Allerdings war die maximal festgestellte Tiefe in `at.*` 484 und in `de.*` 527, d.h. „ewig“ laufende Endlosdiskussionen existieren, ein künstliches Kürzen der `References` ist in solchen Fällen praktisch unvermeidbar.

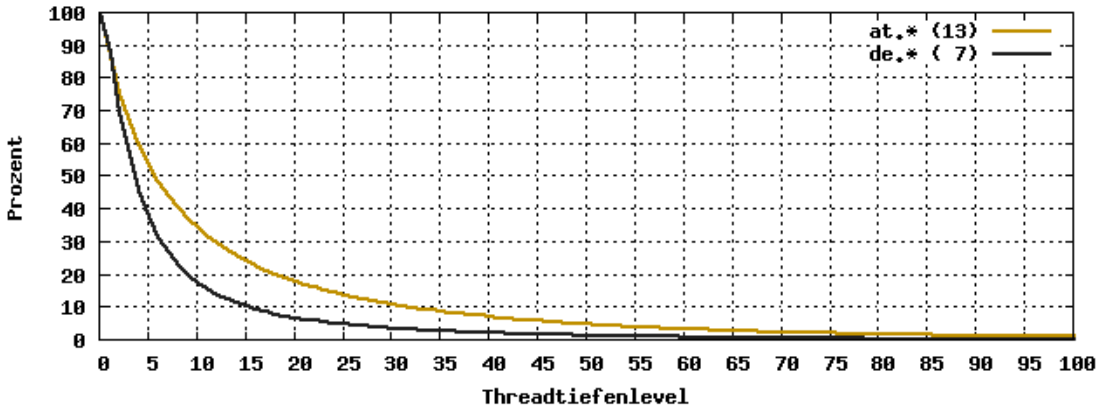

Abbildung 5.3: Prozent der Artikel mit Threadtiefen kleiner x

5.5 Spam

Wie auch in anderen vernetzten Kommunikationsmedien ergibt sich im Usenet die vorteilhafte Situation, viele Empfänger mit relativ wenig Aufwand erreichen zu können. Geschichtlich gesehen kommt es jedoch immer zum selben Ergebnis. Einzelne Netzteilnehmer mißbrauchen die Vorteile, um ihre Artikel massenhaft zu verbreiten, sei es, um möglichst viele Personen mit ihrer „Botschaft“ zu beglücken, oder aus simplem Profitstreben – Werbung für ein Produkt oder eine Website.

Die Teilnehmer des Usenet mußten schon frühzeitig Konzepte entwickeln, um Artikelfluten Einhalt zu bieten, denn es reichte schon ein Server als Injektionspunkt, um eine weltweite Verbreitung loszutreten[14]. Ein Kombinationsansatz sei hier vorgestellt.

5.5.1 Filteransätze

Die Aufgabe ist, den „Nährwert“ Teil im Datenfluß möglichst groß und den „Störanteil“ an Texten möglichst klein zu halten. Als Strategien gibt es:

- Beschränkung der Schreibfähigkeit von Servern.
  Ursprünglich waren alle Server offen für jeden. Heute ist genau das Gegenteil normal, jeder Server nimmt nur Artikel von einem kleinen Kreis Berechtigter (meist nur die Kunden des jeweiligen ISP[15]) an. Destruktive Teilnehmer können leicht auf einen ISP eingegrenzt werden, beziehungsweise Müll aus dessen Richtung einfach ausgefiltert werden (siehe Kapitel 5.1.2)

[14] Im Gegensatz zu E-Mail, wo es erforderlich ist, mit mehr Aufwand Listen von Empfängern zusammenzustellen und jede E-Mail einzeln auszuliefern

[15] ISP = Internet Service Provider. Ein Unternehmen, welches einen Internetzugang anbietet.

- [Breidbart Index][16]
  Eines der bekanntesten vereinbarten Limits gegen Artikelflutungen. Ein einfach zu berechnender Index, wann „zuviel“ an x Artikelkopien in y Gruppen erreicht sind, und automatisiert ein Limit durchgesetzt wird. Verschiedene Hierachien implementieren mitunter engere beziehungsweise weitere Limits.

- Cleanfeed
  Als [Cleanfeed] bezeichnet man einen Filter, der in die Artikelannahme eines Newsservers vorgeschaltet wird und die „nicht gern gesehenen“ Artikel ausfiltert. Dies können verschiedenste Kriterien sein, insbesondere jedoch in HTML kodierte Artikel, Artikel mit überwiegend Daten Bestandteilen („binaries“) und verschiedene Varianten von Detektoren, die idente Artikel feststellen.

Obowhl heute fast alle Server eine Variante von Cleanfeed fahren und Usenet ob der prozentual wesentlich geringeren Teilnehmerquote gegenüber E-mail für Spammer nicht mehr einen hohen Stellenwert besitzt, gibt es heute nach wie vor ein Spamproblem im Usenet.

Die Daten dieser Diplomarbeit wurden jeweils immer nachts gewonnen (siehe Kapitel 4.1.1), der Müll des jeweiligen Tages wurde meist schon größtenteils von international organisierten Despammern entfernt – es bleibt aber immer etwas übrig.

Wie lassen sich Artikelflutungen erkennen?

- Die Artikel werden in einem großen Schub an einem Punkt des Netzes eingespeist.
- Der verkündete Inhalt ist sehr ähnlich.

Das Einspeispunktproblem wurde bereits in Kapitel 5.1 behandelt. Für einen potentiellen Spamfilter gilt es noch, die Ähnlichkeit zweier Artikel quantifizierbar zu machen.

5.5.2 Nilsimsa

Der Nilsimsa Algorithmus ([Nils02]) errechnet einen Fingerabdruck – einen Hashwert – eines Datenblockes. Er tut dies, indem ein 5-Zeichen Fensterausschnitt Zeichen für Zeichen über den Datenblock geschoben wird und in jedem Schritt in diesen 5 Zeichen alle möglichen Dreierzeichenkombinationen ausgewertet werden. Diese Kombinationen bilden einen Zeiger in ein Array von Zählern, wobei dann ein Zähler jeweils beim Vorkommen einer bestimmten Kombination erhöht wird. Eine Reduktionsfunktion bildet am Ende die Zähler auf einen 256 Bit Fingerabdruck (meist dargestellt als 64 Hexadezimalzeichen) ab.

[16] Benannt nach seinem Erfinder Seth Breidbart.

Ein wesentliches Merkmal der Nilsimsafunktion ist, dass ähnliche Datenblöcke ähnliche Hashwerte ergeben. Anders formuliert, kleine Veränderungen in den Inputdaten verändern auch nur einzelne Bits des Nilsimsa Output Hashwertes.

Ein Beispiel:

```
Ich habe genug Geld um mir einen Porsche zu kaufen,
ein Ferrari ist nichts für mich.
54f1481084e093670dff8869b7301fd21847aab43dc456dfb5eacb406ba7e070

Ich habe genug Kohle um mir einen Ferrari zu kaufen,
ein Porsche ist nichts für mich.
55f95a3080b0c36f0daf886f663e7d9218479af82cc4765ff5eacb407f25e264
```

Die Nilsimsa Distanz sei hier[17] als Anzahl der verschiedenen Bits zwischen 2 Hashwerten definiert. Zwei beliebige zufällige Nachrichten sollten somit eine Distanz in der Umgebung von 128 besitzen, je ähnlicher sich 2 Inputblöcke sind, desto geringer wird die Distanz. Obiges Beispiel hat eine Distanz von 43, eine Ähnlichkeit ist gegeben. Wegen der Kürze der Nachricht wirkt sich der Tausch eines Wortes jedoch relativ stark aus.

5.5.3 Information

Es lässt sich nun eine Detektion konstruieren, die Artikel von gleichen Einspeisepunkten auf ihre Ähnlichkeit hin untersucht. Der Artikelstrom durchläuft einen 10000 Einträge langen Buffer. Bei jedem neuen Artikel wird ein Vorgänger mit gleichem Einspeisepunkt im Buffer gesucht, wenn einer existiert wird die Ähnlichkeit (Nilsimsa Distanz) errechnet.

Ein Wert von 255 bedeutet: keinen Vorgänger gefunden.
Eine Distanz von 0 bedeutet: komplett identer Artikel.
Artikel mit einer Größe kleiner 20 Byte werden ignoriert.

Berechnet man die Nilsimsa Distanzen aller Vorgänger-Nachfolger Artikelpaarungen (diese wurden in Kapitel 5.4 rekonstruiert) im Archiv, ergibt sich eine Verteilung wie in Abbildung 5.4

Die Berechnung über die Originalartikel liefert je Hierachie eine Kurve die Richtung Distanz 0 strebt, also ähnliche Artikel. Nach Entfernung von Zitaten wie in Kapitel 5.3 beschrieben, besteht ein Artikel natürlich vermehrt aus neuen Informationen. Die zusätzliche Originalität bewirkt eine Kurvenverschiebung nach rechts.

[17] In der ursprünglichen Definition ist die Nilsimsa Distanz „Anzahl gleicher Bits minus 128“ gegeben, der einfacheren Anwendung bzw. dem Verständnis wegen sei hier eine alternative Definition verwendet.

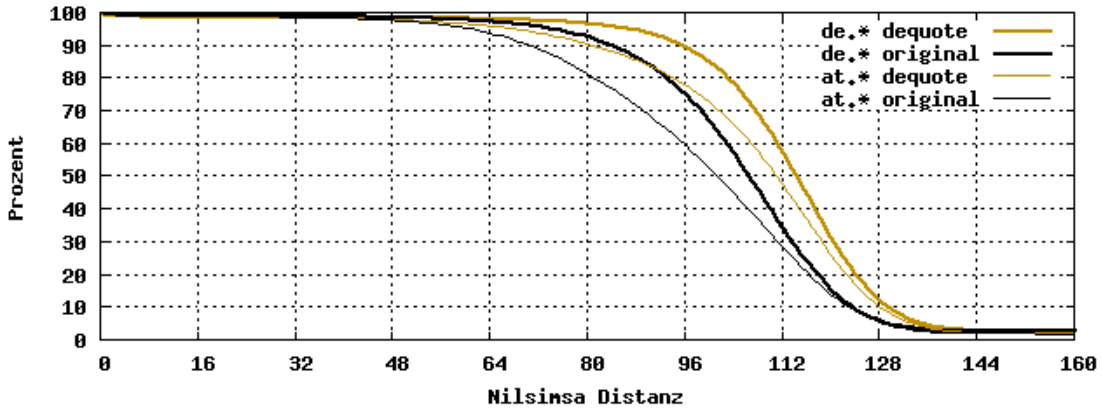

Abbildung 5.4: Prozent der Artikel mit Nilsimsa Distanz kleiner x

Dieser Graph veranschaulicht die prinzipielle Eignung des Algorithmus. Als Spamkandidaten kommen aber natürlich nur Artikel in Umgebung der Distanz 0 in Frage.

Definiert man eine Spamserie als mindestens 5 Artikel (als Kompromiss zwischen „ab 5 Stück werden Artikelkopien für den Leser lästig“ und Breidbartindex als obere Grenze, bei der sowieso vorgefiltert wird) mit durchschnittlicher Nilsimsa Distanz untereinander von 10, ergeben sich in `de.*` 7007 und `at.*` 1892 Kandidaten.

Eine Begutachtung der `Subject`s lässt folgende Unterteilungen annehmen:

- „Echter“ Spam

```
$100 INVESTMENT. GUARANTEED RESULTS OR YOUR MONEY BACK.
!!!! A chance of a lifetime !!!
(_O_) I want a Man In Me Now / SWINGERS & SEX (_O_)
080 Professional Legal Forms for do it yourself! 08
I.N.N.O.C.E.N.T__Y.O.U.N.G__T.E.R.R.I.__S.H.O.W.S.__I.T.__A.L.L.
MAKE MONEY EASILY, THE LEGAL WAY.
REALLY CHEAP Computer books just follow any of the links below
90 KB/s download schneller als die polizei erlaubt! Movies/Games/etc
```

- „Fleißige“ Leute die bei Anzeigen glauben „viel hilft viel“ oder „Missionare“

```
==> Verkauf Notebook Sony Vaio PCG-FX505 - 512MB RAM <==
Bewusstseinskontrolle und Politik in der neuesten Geschichte
Boykott von US-Waren als Antwort der Kriegsgegner
DAS HIER KÖNNTE DEIN LEBEN VERÄNDERN, WENN DU ES GELESEN HAST!
Fragebogen-Befragung
Ich brauche Eure Hilfe!
```

- „Testpostings“

```
test (ignore , no reply)
ignore - no reply pp
test5, ignore
test 123
```

Obwohl der implementierte Algorithmus bereits wenige falsche Positive (zum Beispiel auch ein paar FAQ) liefert, kann man noch eine Verbesserung anstreben.

5.5.4 Wörterbuch

Per Definition sind die `de.*` und `at.*` Hierachien deutschsprachig. Es gibt manchmal Hierachiebesucher bei denen Englisch durchaus in Ordnung ist, bei Artikelserien die in obigen Detektor fallen wird es jedoch eher keine normale Diskussion sein.

Unter Zuhilfenahme eines deutschen und englischen Wörterbuches läßt sich eine Sprachabschätzung der Artikel vornehmen. Artikelserien die mehrheitlich englisch sind, können als Spam angenommen werden. Als Wörterbuchbasis dient eine Version[18] von Mitte Oktober 2004 des Wörterbuchs von `http://www.dict.cc/`, welche ca. 150000 deutsche und englische verwertbare Einträge besitzt.

Die Artikel werden in Wörter aufgespaltet und beide Wörterbücher durchsucht. Ein Artikel wird in „D“ klassifiziert, wenn mehrheitlich deutsche Wörter gefunden werden, in „E“, wenn mehrheitlich englische und in „-“, wenn das Ergebnis nicht eindeutig ist.

Eine Kategorisierung des gesamten Datenbestandes ergibt Tabelle 5.1. Die Sprachendetektion angewandt auf den Ergebnissatz des vorherigen Nilsimsa Detektors liefert Tabelle 5.2. Eine graphische Aufbereitung des Nilsimsadatenergebnisses aus Tabelle 5.2 ist in Abbildung 5.5 zu sehen.

	D	E	-
de.*	7703841 (95,40%)	211510 (2,62%)	160150 (1,98%)
at.*	414452 (91,69%)	19692 (4,36%)	17863 (3,95%)

Tabelle 5.1: Sprachschätzung im Gesamtdatenbestand

In Abbildung 5.5 stellt die x-Achse die Zeit dar, man sieht am unteren Rand Wochenmarkierungen, Monatsmarkierungen und eine Jahrestrennmarkierung. Die y-Achse entspricht der sortierten Gruppenliste wie in Anhang B.4, die binäre Codierung linksseitig in der Grafik dient als Orientierungshilfe.

[18] Damals noch unter GPL v2 Lizenz downloadbar – die aktuelle Version ist heute nur mehr eingeschränkt verfügbar.

	D	E	-
de.*	4053 (57,84%)	1833 (26,16%)	1121 (16,00%)
at.*	499 (26,37%)	1183 (62,57%)	210 (11,10%)

Tabelle 5.2: Sprachschätzung in detektierten Artikelwellen

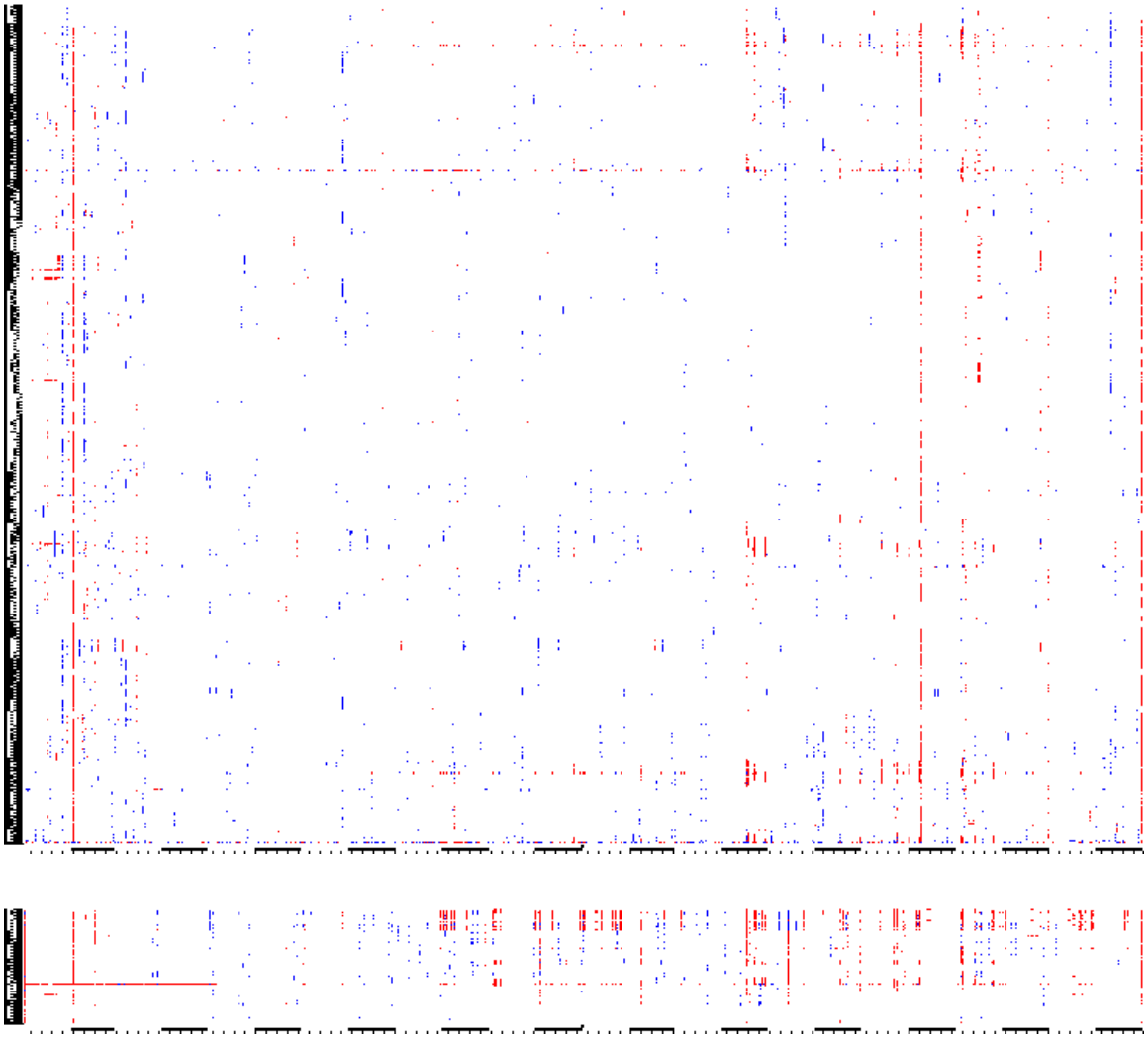

Abbildung 5.5: Nilsimsa mit Wörterbuch, `de.*` oben, `at.*` unten

Ein Pixel entspricht einem Tag in einer bestimmten Gruppe. Ist das Pixel rot, gab es zu diesem Zeitpunkt einen Spamartikel der Kategorie „E“ oder „-“. Ist ein Pixel blau, war es ein Artikel der Kategorie „D“.

Horizontale Streifen lassen sich als Testserien interpretieren. Im `de.*` Teil fällt nahe dem oberen Rand `de.alt.test` auf, ganz am unteren Rand ist es `de.test`. In `at.*` ist die auffallendste Linie `at.test`

Vertikale (fast) durchgehende Streifen sind „echte“ Spamwellen. In `de.*` stechen 3 Linien hervor, Anfang Februar 2003, August 2004 und Silvester 2004.

Zwischendurch gibt es nur kurze vertikale Striche. Dies lässt sich durch das Wissen um den Breidbart Index erklären. Spammern ist bekannt, dass sie leicht in diesen Filtern hängenbleiben, wenn sie zu viele Artikel in Serie plazieren. Ein kurzer Spamlauf, zufällig in der Mitte der Hierachie über ein paar Gruppen, fällt wesentlich weniger auf.

Im `at.*` fallen im oberen Teil die vertikalen Striche auf. Dies sind die `at.anzeigen.*` Gruppen in denen „viel hilft viel“ eine Verdichtung der Striche bewirkt.

Zusammenfassend ist die Qualität des `aconews` Servers als sehr gut zu bewerten. Die Spammenge hält sich in Grenzen und die Vorfilterung ist gut umgesetzt (soweit sich dies ohne manuelle Begutachtung eines *großen* Teiles des Archives beurteilen lässt).

5.5.5 Praxiseinsatz

Noch während des Verfassens dieser Diplomarbeit wurde der Newsserver der TU Graz[19] (der weltweit lese- und schreiboffen konfiguriert ist) von einer Spamserie heimgesucht.

Die in diesem Kapitel verwendete Nilsimsa plus Ursprungshostdetektion wurde an die Verhältnisse der TU angepasst, portiert, mit einem zusätzlichen Löschprogramm ausgestattet und nach Absprache mit dem Zentralen Infomatikdienst der TU zur Überwachung des TU Newsservers scharfgeschaltet.

Das Ergebnis war zufriedenstellend, die weiteren Spamserien wurden innerhalb kurzer Zeit detektiert und gelöscht[20]. Mit der schnellen Löschung der Artikel und somit dem ausbleibendem Erfolg der Anlockung von Besuchern auf die beworbenen Websites wurde die TU recht schnell von den Spammern wieder verschont.

5.6 Suchprobleme

Ein traditionelles Problem von größeren Datenmengen ist die Suche nach Informationen in diesen. Im gegebenen Fall von einem Usenet Archiv bedeutet dies meist die Suche nach einem Artikel, den man bereits einmal gelesen hat oder eine Suche nach

[19] `http://www.zid.tugraz.at/ki/internet/news/tug.html`

[20] `http://newsarchiv.tugraz.at/browse/tu-graz.cancel-reports/msg05612.html`
`http://newsarchiv.tugraz.at/browse/tu-graz.cancel-reports/msg05619.html`
`http://newsarchiv.tugraz.at/browse/tu-graz.cancel-reports/msg05649.html`
`http://newsarchiv.tugraz.at/browse/tu-graz.cancel-reports/msg05722.html`

bestimmten Kriterien und als Resultat Artikelvorschläge, die die Kriterien möglichst genau erfüllen.

Sucht man einen genau bestimmten Artikel, lässt sich das Problem heutzutage immer öfter mit dem Lösungsansatz „Rechenleistung“ implementieren. Unscharfe Anfragen erfordern jedoch Abschätzungen über den Datenbestand und sind ein aktives Forschungsgebiet mit noch vielen ungelösten Problemen.

In diesem Kapitel wird versucht, einen kurzen Einstieg auftretender Randbedingungen für Usenet Suchmaschinen zu geben.

5.6.1 Spezifische Suche

Die Suche mit einem genauer bekannten Kriterium ermöglicht ein einfacheres Design von Suchmaschinen. Im Prinzip entspricht es einer großen Datenbank, in der nach bestimmten Schlüsseln gesucht wird. Entscheidend ist jedoch die Konstruktion und Verknüpfung der Inhalte, um aus dem Wissen um die Usenet Artikelstruktur sinnvolle und effiziente Anfragen zu ermöglichen.

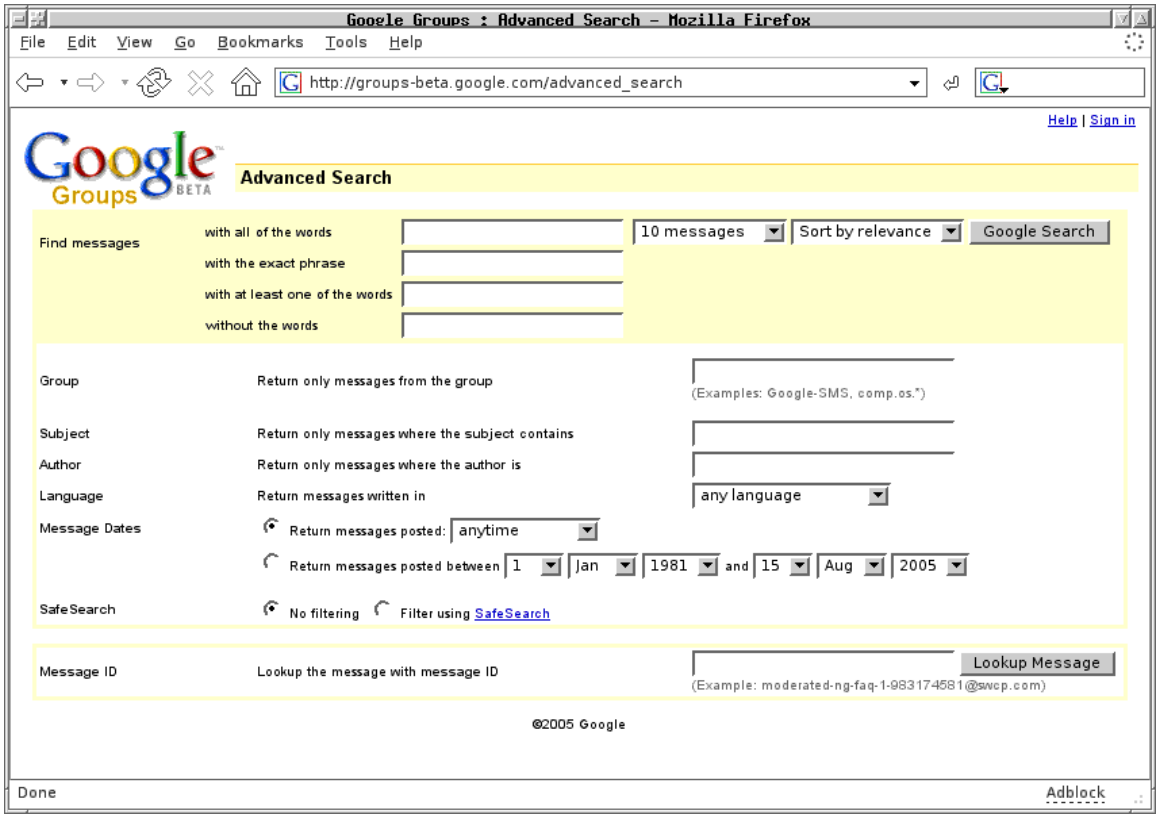

Abbildung 5.6: Google Groups Suchmaske

Als Beispiel der „üblichen“ Kriterien, im Usenet Artikel zu suchen, ist in Abbildung 5.6 mit einem Screenshot von Google Groups gezeigt. Die Google Groups Suchmaske ermöglicht primär Suchanfragen nach den „nackten“ Kenndaten von Usenet Artikeln. Von Interesse sind hierbei natürlich die Datenpunkte im Artikelkopf: Gruppe, Betreff, Autor, Datum und Nachrichten-ID.

Etwas komplexer ist der Parameter „Sprache“, wo bei der Abspeicherung von Artikeln in der Datenbank ein zusätzlicher Datenpunkt mit einer Sprachenabschätzung generiert wird, was später zur Eingrenzung des Suchbereiches verwendet werden kann. Der Block der obersten 4 Zeilen offeriert die Suche nach konkreten Wörtern. Dies ist einfach realisiert und praktisch wenn man nach bestimmten Produktbezeichnungen sucht. Bei allgemeinen Wörtern des Sprachsatzes wird das Suchproblem wesentlich komplexer.

5.6.2 Ungefähre Suche

Bei einer unscharfen Suchanfrage ist es nicht mehr möglich, durch simples Nachsehen in Tabellen den Datensatz in einer Datenbank zu lokalisieren. Die Daten könnten auch „ähnlich“ vorliegen. Es gilt, eine Wahrscheinlichkeit der Übereinstimmung zu finden. Ein Ziel ist es, mehrere Datenmerkmale in größeren Klassen zusammenzufassen und dann (in der hoffentlich kleinen) Schnittmenge der Suchanfrage dieser Klassen nach der Lösung zu suchen.

Welche Reduktionen am Usenet Artikelrohtext sind notwendig, um möglichst wenige Kernmerkmale zu erhalten? Als Testdatenbestand wurde die Gruppe `at.linux` ausgewählt, eine Gruppe als Beispiel, in der man sicher einmal nach einer Lösung für ein bestimmtes Linux Problem suchen wird. Die Daten über 2 Jahre dieser Gruppe belaufen sich auf 36359 Artikel in 18,8 Mb Text. Es wurden 127253 verschiedene Wörter gefunden, die sich mit Hilfe des bereits in Kapitel 5.5.4 erwähnten Wörterbuches in 8122 englische, 15680 deutsche und 103451 unbekannte Wörter unterscheiden lassen.

Die hohe Anzahl an unbekannten Wörtern verlangt eine genauere Untersuchung obiger Sortierergebnisse. Es ergeben sich folgende Beobachtungen, die bei der Datenvorverarbeitung berücksichtigt werden sollten, also bevor die Daten mit Ähnlichkeitsalgorithmen von Suchmaschinen weiterverarbeitet werden:

- Die Rohdaten enthalten nicht nur Zeichensatzfehler, sondern auch Tippfehler, Usenet übliche Ausschmückungen von bestimmten Wörtern und umgangssprachliche Modifikationen.
- Die benutzten Wörterbücher enthalten kaum Fachbegriffe. Insbesondere Begriffe der Linux Fachwelt sind als komplett unbekannt anzunehmen.
- Ca. 10% der unbekannten Begriffe sind Kombinationen aus Buchstaben und Zahlen, die Konstanten aus Computercode oder Konfigurationsdateien entstammen.

- Ca. 65% der unbekannten Begriffe treten nur ein einziges Mal auf, ca. 88% aller unbekannten Begriffe treten maximal 5 mal in Erscheinung. Es ist entscheidend für die Effizienz einer Suche, diese Einzelfälle in Klassen ungefähr gleicher Bedeutung zusammenzufassen.

- Die (verwendeten) Wörterbücher kennen nur einzelne Vertreter einer ganzen Wortfamilie. Zum Beispiel:

  Als Deutsch wurde erkannt:
  unterschied(407x), unterschiede(82x), unterschieden(14x), unterschiedlich(49x), unterschiedliche(61x).

  Als unbekannt wurden jedoch eingestuft:
  unterschiedlichem(2x), unterschiedlichen(58x), unterschiedlicher(17x), unterschiedliches(1x), unterschiedlichsten(5x), unterschiedlichster(2x).

  Abhilfe schafft eine „stemming“ Stufe, eine Verarbeitungsstufe, die Wörter auf ihren Wortstamm zurückführt, was aber eine viel größere und gepflegte Wörterbibliothek verlangt.

- Die Wörter, die am häufigsten vorkommen, besitzen keinen primären Informationswert und können im allgemeinen ignoriert werden.

  Die 15 meistgefundenen deutschen Wörter sind:
  ich(45659x), die(40521x), das(36927x), und(36069x), der(29407x), nicht(29248x), ist(27668x), mit(20587x), auch(18089x), ein(17513x), auf(16152x), aber(15386x), den(15048x), von(13569x), eine(12208x).

  Zum Vergleich die häufigsten unbekannten Wörter:
  linux(12036x), http(6979x), martin(6319x), mal(6162x), com(5181x), usr(5138x), hab(5128x), „2003“(4568x), mfg(4014x), des(3856x), dev(3780x), bartolich(3668x), gmx(3621x), dir(3606x), ber(3552x).

Zusammenfassend sieht man einen erheblichen Vorverarbeitungsaufwand, wenn man über die simple genaue Wortsuche hinausgehen und die entscheidenden Datenpunkte in einem Artikel von den unwichtigen Teilen trennen will, um sie einer späteren Beziehungsanalyse zuzuführen und unscharfe Suchanfragen zu ermöglichen.

Das Gebiet der unscharfen Suchalgorithmen in Texten ist eine Diplomarbeit für sich selbst. Versuche, den gegebenen `at.linux` Datenpool einem Klassifizierungsalgorithmus nach der „Latent Sematic Indexing“[21] Methode zuzuführen, scheiterten nach hundert Artikeln an der benötigten hohen Rechenzeit und wurden abgebrochen. Als Erkenntnis bleibt die notwendige Aufrüstung der Wörterbücher und weiteren Verbesserung der Zitat/Kodierungsdetektoren, bevor man sich mit dem eigentlichen Suchproblem beschäftigt.

[21] `http://en.wikipedia.org/wiki/Latent_Semantic_Indexing`

6 Perspektiven

Der von dieser Diplomarbeit gebotene Rundgang durch das Usenet System eröffnet dem Leser hoffentlich viele neue Perspektiven und bringt Licht in dunkle Ecken der Konstruktionsprobleme eines ähnlich angelegten Kommunikationssystems. Welche Erkenntnisse gewonnen wurden und welche Weiterentwicklungen es noch geben könnte, versucht dieses Kapitel zu erörtern.

6.1 Zusammenfassung

Das Usenet als Kommunikationsmedium hat über die Jahre die gesamte Entwicklung von Kindesbeinen über Kinderkrankheiten, stürmische Teenagerwachstumsperioden und schließlich erwachsene Stabilität und Reife durchlaufen. Obwohl noch aktiv an Erweiterungen gearbeitet wird, wird sich der Charakter und Aufbau des Mediums wohl nicht mehr radikal ändern. Ansätze für neue Diskussionssmedien sollten die Erkenntnisse und Erfahrungswerte der Ära Usenet berücksichtigen.

6.1.1 Was ist positiv

Das NNTP Protokoll (Kapitel 3.1) ist, wie viele andere auf ASCII basierende Internetprotokolle, eine Erfolgsgeschichte. Simpel im Design und leicht verständlich ist es einfach zu implementieren. Dies führte auf Clientseite zu einem bunten Strauß an Newsprogrammen, für wirklich jeden Geschmack findet sich ein passendes Programm. Die Offenheit des Protokolls ermöglichte auch experimentelle Erweiterungen im laufenden Betrieb zu testen und später diese fließend in der Population einzuführen, weil sie für gut befunden wurden.

Das dezentrale Design (Kapitel 3.2.2) gab dem Usenet eine die Zeit überdauernde Robustheit. Die „erzwungene“ Kooperation der Administratoren bringt ein Minimum an technischem Wissen an die (Administrations-) Knotenpunkte des Netzes, nur wer sich an die gemeinschaftlichen und technischen Spielregeln hält darf mitspielen (Kapitel 5.1.2 und 5.5.1). Andererseits ermöglicht es genug Freiraum für von unten kommende Erneuerungsbewegungen. Jeder kann neue Pfade beschreiten und versuchen, Anhänger für seine Veränderungen zu begeistern.

Vom Standpunkt der Meinungsfreiheit ist Usenet noch immer ein potentiell anonymes

Medium. Die Detektion der wahren Identität ist nicht 100% realisierbar (Kapitel 5.1 und 5.2), was für den Meinungsaustausch in einer demokratischen Gesellschaft sehr wichtig ist. Das Potential von anonymen [Internet Troll]en im Usenet System wiegt die Vorteile für die Diskussionskultur gesamt gesehen auf.

Dem Spamproblem (Kapitel 5.5) wurde recht früh mit automatisierten Filtern, im Konsens beschlossenen Limits und Säuberungsprogrammen („Cancelbots“) begegnet. Heutzutage wird Usenet wegen der gesunkenen Popularität weitgehend von Spammern gemieden.

6.1.2 Was ist problematisch

Die Planung der (zukünftigen) Kapazitäten eines Nachrichtenaustauschsystems ist immer ein Problem. Ursprünglich nur für eine handvoll Artikel konzipiert, produziert Usenet heute mehr als ein Terabyte/Tag Umsatz[1] an Daten. Die gleichzeitige Entwicklung der Rechnerleistung hat diese Entwicklung zwar gut abgefedert, aber diese Datenmenge wird noch immer mit den gleichen Datenstrukturen wie vor 20 Jahren verwaltet: Gruppe, Artikelnummer und Artikel `Message-ID`. Dies führt zu Problemen für einen effizienten Abgleich (Kapitel 3.3), als auch bei der Speicherung (Kapitel 3.4). Es wurde verabsäumt, frühzeitig (Datenbank-) Konzepte für die effiziente Integration großer („fremder“) Datenobjekte, in Anzahl als auch Größe, zu entwickeln.

Die Inspektion der Headerinformationen (Kapitel 4.2) zeigt eine Fülle von Dingen, die schief laufen können. Es gilt, für robuste Software jedweden Blödsinn, der angeliefert wird, möglichst frühzeitig zu erkennen und wenn möglich zu stoppen. Der Wachstumsprozeß und die liberale Internetkultur[2] führten zu zuviel Toleranz bei der Umsetzung der Standards in die Praxis. Nun gibt es „Abweichungen“ (Kapitel 4.2.5), die sich zu einem chronischen Problem entwickelt haben und noch lange ein Störfaktor sein werden.

Die frühe Einführung der Zitatkultur (Kapitel 5.3) ermöglichte einen sehr effizienten Diskussionsstil. Über die Jahre verwässerte der Prozeß jedoch durch Anstöße aus mehreren Richtungen (verschiedene Zitatzeichen, Zeilenendeumbruch Implementation, halb erfolgreiche Standards (Format=flowed), Kammquotings, ...) so sehr, dass die Zitate, eine Kernfunktionalität, an Nützlichkeit verloren haben. Dies begünstigt ein Abwandern der Nutzer in andere Medien, zum Beispiel Mailinglisten und Webforen.

[1] Sep.04: 1403 Gb/day for a Usenet full feed
`http://documentation.highwinds-software.com/docs/index/managing_feed_growth.html`

[2] „Be liberal in what you can accept, conservative in what you generate.“
Diese Weisheit wird dem Internetpionier Jon Postel zugeschrieben, der sie 1981 in RFC793, Kapitel 2.10, als „Robustness Principle“ niederschrieb.

6.2 Ausblick

Pessimistisch betrachtet geht es mit dem Usenet bergab (Kapitel 4.1.3). Optimistisch betrachtet wird es das Usenet ewig geben, es ist schlichtweg zu groß und zu nützlich für bestimmte Zielgruppen.

Im speziellen an der TU Graz bietet sich eine günstige Konstellation, die zu einem Fortbestand des Newsservers führt (und die Diskussionsforen des TUGonline Systems vereinsamen lässt):

- Ein konstanter Zustrom technisch orientierter Teilnehmer.
- Ein signifikanter Kommunikationsbedarf zwischen tausenden Personen zur Koordination von Lehrveranstaltungen.
- Eine genügend große Zahl an Teilnehmern, um auch allgemeinen Themen Diskussionsraum zu bieten.

Die breite Masse der Internetteilnehmer setzt jedoch vermehrt auf die Diskussionsform des Webforums. Einen Webbrowser hat jeder Rechner schon heute vorinstalliert und der Einstieg gestaltet sich recht einfach. Man muß sich nicht mit einem extra Stück Software plagen und weicht den Reibungspunkten aus, wenn man nicht auf die Eigenheiten von in 20 Jahren gewachsener Usenet Kultur trifft. Im Webforum sind (fast) alle gleich, man könnte es mit „das Internet ist neu und bunt und alle haben Spaß“ beschreiben.

Aus der Technikerperspektive klafft jedoch eine Lücke zwischen dem Usenet und „modernen“ webbasierten Foren. Es ist kein sanfter Übergang auf eine neue Plattform, sondern eher ein Sprung. Die Webforum Entwicklung implementiert viele gute Usenet Konzepte nicht. Zum Beispiel die Darstellung von Baumstrukturen (Kapitel 5.4) von Diskussionen ist vielen Webforen unbekannt, meist ist nur eine lineare Artikelauflistung verfügbar. Konzepte wie eine langfristige Archivierung fehlen dem jungen Medium Webforum völlig. An alten Problemfällen werden gänzlich neue Lösungansätze versucht, zum Beispiel Quotings (Kapitel 5.3) durch explizite „[QUOTE] ... [/QUOTE]“ Konstrukte im Text, anstatt von speziellen Quotingzeichen.

Versuche, das Usenet zu retten und dem modernen Internet Nutzer näher zu bringen, gibt es. Mehrere Implementationen ermöglichen den Zugriff auf einen Newsserver per Webbrowser Schnittstelle – Beispiel siehe Abbildung 6.1. Die bisherigen Prototypen sind jedoch als Newsserver Aufsatz konzipiert und reichen weder aus, dem Usenet Stammpublikum einen neuen Zugang schmackhaft zu machen, noch neues Publikum für das alte Medium Usenet zu begeistern.

Ein lohnenswertes Experiment wäre die Implementation eines vollständigen (mächtigen) traditionellen Newsclients mit Hilfe moderner „Web 2.0“[3] Technologien, in enger

[3] `http://en.wikipedia.org/wiki/Web_2.0`

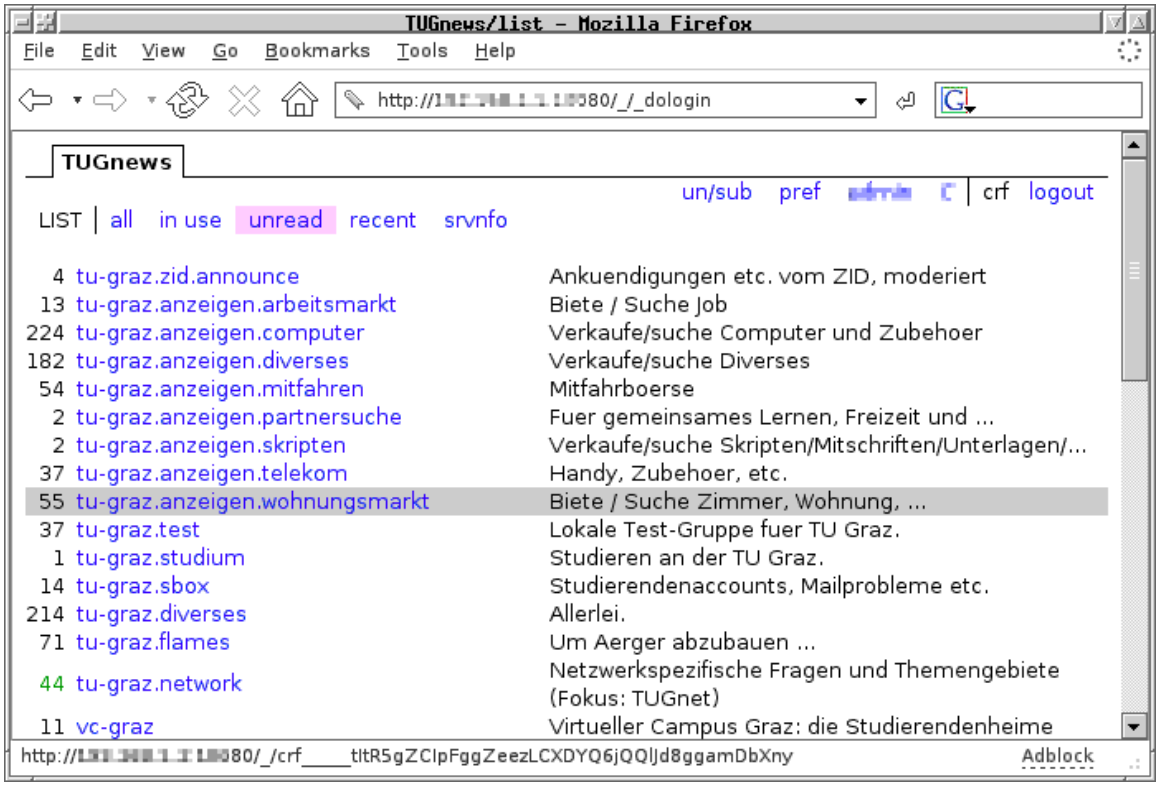

Abbildung 6.1: Beispiel eines Webbrowser basierten Newsclients

Verbindung mit einem Newsserver. Moderne Webbrowser bieten eine ausreichend starke Funktionalität, um für viele Anwendungen eine separat zu installierende Software überflüssig zu machen. Der einfache Einstieg von überall mit bewährter Funktionalität wäre gewährleistet. Durch die enge Kopplung an einen Newsserver Unterbau bestünde die Möglichkeit, „good behaviour“ Standards durch die Limitierungen des Webinterfaces zu forcieren. Die bewährten Werte des Usenet würden auf eine modernde Plattform portiert und eine attraktive Alternative gegenüber dem „Wildwuchs“ der derzeitigen Webforen darstellen.

A Praktische Durchführung

Die im Haupttext genannten Daten wurden – so nicht explizit gekennzeichnet – durch selbsterstellte Programme und Algorithmus Implementierungen aus den gesammelten Daten extrahiert. Dieser Anhang gibt einen Einblick auf die praktische Durchführung und den entwickelten Programmcode.

A.1 Entwicklungsumgebung

Als Entwicklungsumgebung diente ein handelsüblicher x86 PC mit 2 Ghz Taktfrequenz und einer 200 Gb Datenplatte. Diese Rechenleistung entsprach gut den Erfordernissen für die praktische Durchführung. Auf der Datenplatte konnten alle Originaldaten und mehrere Zwischenstufen der daraus gewonnen Daten gespeichert werden. Die Rechenleistung war ausreichend, um die Laufzeiten selbst der komplexeren Algorithmen kleiner als eine Woche zu halten.

Als Software wurde als Betriebsystem Linux eingesetzt. Zur Entwicklung der erforderlichen Programme zur Datenanalyse wurde fast gänzlich auf die objektorientierte Skriptsprache Ruby[1] gesetzt. Ruby als Skriptsprache der neuesten Generation bietet eine „alles ist ein Objekt“ Philosophie von Daten. In Kombination mit einem sehr dynamischen Typensystem eignet sie sich sehr gut für die Implementierung und Konzeption neuer Algorithmen und Prototypen. Als interpretierte Sprache ist Ruby etwas langsamer als andere populäre Sprachen, die schnellen „edit->run“ Entwicklungszyklen (also ohne Compilationszeiten) wiegen dies jedoch gut auf.

Für Textmanipulationen wurde weiters auf die GNU Text Utilities[2] gesetzt, die bei jeder modernen Linux Distribution inkludiert sind. Diese Utilities sind in optimiertem C geschrieben und eigenen sich ob ihrer schnellen Ausführungsgeschwindigkeit gut für große (Text-)Datenmengen. Wurde in dieser Diplomarbeit beispielsweise eine Datei mit einem Ergebnis pro Zeile generiert, lassen sich Zeilen mit einem gewünschten Kriterium leicht per „grep“ auswählen und dann per „wc -l“ die Anzahl der Ergebnisse bestimmen – ohne dafür ein eigenes Programm schreiben zu müssen.

[1] Die Ruby Hauptwebsite ist `http://ruby-lang.org/en/`.
Die Ruby Sprachreferenz schlechthin ist [Thom04], welche im Laufe dieser Diplomarbeit sich als sehr nützlich erwies.

[2] `http://www.gnu.org/software/textutils/textutils.html`

A.2 Inhaltsübersicht der beigefügten CD

Die beigefügte CD enthält folgende Verzeichnisstruktur:

`/latex`
Dieses Verzeichnis enthält den LaTeX Quelltext dieser Diplomarbeit, inklusive aller eingefügten Bilder, Tabellen und Textdateien.

`/papers`
Hier finden sich die im Literaturverzeichnis referenzieren Texte und Standards.

`/pictures`
Im Laufe der Diplomarbeit wurden für jede einzelne Gruppe mehrere Graphen generiert, jedoch nur einzelne Exemplare in diesem Text verwendet. Eine Sammlung der generierten Graphen für jede Gruppe findet sich in `/pictures`

`/scripts`
Als Hauptstück enthält dieses Verzeichnis alle selbst entwickelten Programmteile, nach Kapiteln geordnet.

`/sources`
Abschließend dient dieses Verzeichnis als Sammlung von Programmen und Bibliotheken die in dieser Diplomarbeit Verwendung fanden.

A.3 Externe Quellen

Folgende Programme wurden nicht vom Autor dieser Diplomarbeit entwickelt, sondern sind Programmpakete aus dem Angebot des Internets, die sich für Teilprobleme eigneten und somit nicht selbst entwickelt werden mussten.

`/sources/amatch-0.1.5.tgz`
Amatch ist eine Stringvergleich Bibliothek und bietet die Levenshtein Funktion die in Kapitel 5.3 genutzt wurde.
`http://raa.ruby-lang.org/project/amatch/`

`/sources/classifier-1.3.0.tgz`
Classifier ist eine Bibliothek, die Bayesian, LSI/LSA und andere Klassifikationen von Text implementiert. Aus dieser Bibliothek wurden die Text zu Wörtern Trennungsroutinen in Kapitel 5.5 und 5.6 verwendet.
`http://raa.ruby-lang.org/project/classifier/`

`/sources/gnuplot-3.8j.0.tar.gz`
Gnuplot ist das Plotpaket mit dem (fast) sämtliche Graphen in diesem Text erstellt wurden.
`http://www.gnuplot.info/`

`/sources/newsleech-0.1.1.tar.gz`
Newsleech ist ein kleines Programm welches in der Implementation des in Kapitel 3.3.2 beschrieben Abgleichverfahrens den NNTP Teil übernimmt.
`http://cube.dyndns.org/~rsnel/newsleech/`

`/sources/nilsimsa-0.2.4.tar.gz`
Ist die Original Nilsimsa C Implementation. Aus dieser wurde eine Ruby Implementation für Kapitel 5.5 abgeleitet.
`http://ixazon.dynip.com/~cmeclax/nilsimsa.html`

`/sources/ruby-1.8.2.tar.gz`
Die Programmiersprache Ruby selbst.
`http://www.ruby-lang.org/en/`

`/sources/rubymail-0.14.tar.gz`
`/sources/rubymail-0.17.tar.gz`
Die Rubymail Bibliothek implementiert die Parserroutinen um einen `From` Headereintrag in seine Einzelteile zu spalten. Version 0.14 wird in Kapitel 3.3.1 verwendet, Version 0.17 fand in Kapitel 5.2 Anwendung.
`http://raa.ruby-lang.org/project/rubymail/`

A.4 Erstellte Programme

Die selbsterstellten Ruby Skripte befinden sich im Verzeichnis `/scripts`. Dieses unterteilt sich wiederum in einen `/lib` Teil, der mehrfach verwendete Bibliotheksroutinen enthält und in `/kap...` Verzeichnisse, in denen sich die spezifischen Skripte zu den jeweiligen Kapiteln befinden.

Während der Entwicklung befanden sich alle Skripte in einem Verzeichnis. In der hier benutzten Strukturierung nach Kapitel sind die Skripte ohne Korrektur von Verzeichnisangaben nicht lauffähig. Weiters wird darauf hingewiesen, dass die Skripte nicht streng in der Kapitel Reihenfolge entwickelt wurden und deswegen ein Skript in einem frühen Kapitel eventuell ein Verständnis eines Teiles eines späteren Kapitels verlangt.

Aus Datenschutzgründen und in Folge der großen Datenmengen ist es nur vereinzelt möglich gekürzte Testdaten beizufügen. Ein nachrechnen der ermittelten Werte ist somit leider nicht möglich. Der Autor hofft trotzdem, dass ein Studium der Programmfragmente ein Verständnis der gewählten Implementierungswege ermöglicht.

Ein experimenteller Aufbau, in dem alle Skripte sauber der Reihe nach ausgeführt werden (was in der Praxis wegen der manuellen Zwischenschritte via GNU Textutils nicht möglich ist), würde für einen einmaligen Durchlauf mehrere Wochen benötigen.

A.4.1 Zu Kapitel 3.3.1

`ticker.rb`
Implementiert den in Abbildung 3.1 gezeigten TU Newsserver Newsticker via NEWNEWS Sychronisation.

`rmail` entspricht der Rubymail 0.14 Bibliothek.
`nntp.rb` ist eine Ruby NNTP Implementation. Dies ist eine leicht modifizierte Version der Originalversion von einer Person namens „nazgul“. Für die Originalversion konnte leider kein aktueller Download Link mehr gefunden werden.

A.4.2 Zu Kapitel 3.3.2

In diesem Verzeichnis ist der `at.*` Artikelabgleicher per LISTGROUP Verfahren, mit dem die Daten für diese Diplomarbeit jeweils in der Nacht archiviert wurden. Die Daten werden in einem Format ähnlich dem traditionellen (siehe Kapitel 3.4.1) gespeichert, mit dem Unterschied, dass die Gruppenhierachie flach ist und Artikelheader und -body getrennt in `.head` und `.body` abgelegt werden.

`01dolist` holt eine Gruppenübersicht von einem Newsserver.
`02doindex` liest für jede Gruppe den aktuellen Artikelindex per LISTGROUP.
`03dosync` synchronisiert anhand der Indexdaten. Neue Artikel werden geholt und in ein `cache.at` Verzeichnis gespeichert, nicht mehr aufscheinende in ein `archive.at` Verzeichnis verschoben.

`inter` ist ein Hilfeskript für `02doindex`, welches die Index Differenzbildung durchführt.
`newsleech` ist ein externes Programm zum Artikel download.

A.4.3 Zu Kapitel 4.1.2

`copy2hashname.rb`
Die Artikeldaten werden von dem traditionellen Speicherformat in das vorgeschlagene Hashformat (siehe Kapitel 3.4.3) umkopiert.

A.4.4 Zu Kapitel 4.1.3

`plot-total.rb`
Dieses Skript generiert Abbildung 4.1.

A.4.5 Zu Kapitel 4, Datenbank

`split-tobodyhead.rb`
Kopiert die `.head` und `.body` Rohdaten in jeweils ein großes Datenbank File. Alle Header und Bodys kommen in ein großes `.data` File und ein simpel aufgebautes `.indx` File mit Einträgen der Form
`ArtikelID Startposition Länge`
ermöglicht ein einfaches Wiederauffinden von Daten (siehe auch Kapitel 3.4.4). Artikelbodys werden vor dem Speichern nach ASCII Klartext dekodiert, soweit erforderlich und möglich.

`0-rendertest.rb`
Ein Testprogramm, welches einen Rohartikel einliest und nach ASCII Klartext dekodiert wieder ausgibt.

A.4.6 Zu Kapitel 4, Headercheck

`checkforheaders.rb`
Implementiert mehrere Varianten nach (defekten) Angaben in Artikelheadern zu suchen. Zusätzlich wird eine Headerübersicht generiert (Beispielfiles `listh...` und `badkeys...`)

`keysortnprinttop.rb`
Liest die Headerübersicht wieder ein und listet die 50 häufigsten.

A.4.7 Zu Kapitel 4, Aufspalter

`split-bygroup.rb`
Generiert für jede Gruppe eine Übersichtsliste ihrer Artikel.

`split-byhead.rb`
Extrahiert aus allen Artikeln den Inhalt eines bestimmten Headers und speichert diese in eine Übersichtsliste. 27 ausgesuchte Headertypen werden untersucht.

A.4.8 Zu Kapitel 4.2.1

`checknullfilesize.rb`
Detektiert ob der `.head` oder `.body` Teil eines Artikels 0 Byte groß ist (`nulls..` sind gefundene Beispiele).

A.4.9 Zu Kapitel 4.2.2

`listsubj...` sind Beispiele des nicht ordnungsgemäßen Zeilenumbruchs.

A.4.10 Zu Kapitel 4.2.4

`mid-stats.rb`
liefert die `Message-ID` Zählung für Abbildung 4.3.

A.4.11 Zu Kapitel 4.2.6

`plot-reader.rb`
Ist jenes Skript, welches zur Generierung der Newsreader Statistiken in Abbildung 4.5 verwendet wurde. Bedingt durch die verschiedenen Zusammensetzung von `at.*` und `de.*` muß es händisch stark editiert werden.

A.4.12 Zu Kapitel 5.1

`path.rb`
Liest alle `Path` Einträge und schätzt jeweils den Einspeiseort.

Die eigentliche Arbeit verrichtet jedoch die Bibliotheksfunktion `DAHeader::guess_path`.

A.4.13 Zu Kapitel 5.2

`decode-from.rb`
Dekodiert einen `From` Headereintrag nach ASCII Klartext.

`decode-from-name.rb`
Extrahiert den Namensteil von `From`.

`identity.rb`
Generiert die Gruppenprofile in Abbildung 5.1. Je Gruppe werden zuerst alle möglichen Identitäten errechnet, die Verteilung der Identitäten im jeweiligen Monat festgestellt, prozentual gewichtet und ein Datensatz für Gnuplot geschrieben.

A.4.14 Zu Kapitel 5.3

`stat-bodysize.rb`
Errechnet die Daten für Abbildung 5.2.

`matchtest.rb`
Ist das simple Vater ->Mutter Levenshtein Beispiel.

`0-testdequote.rb`
Kurzes Testprogramm für einzelne Dequoting Fälle.

`dequoter.rb`
Wendet den Zitatentferner auf eine ganze Datenbank aus Artikelbodys an, unter Berücksichtigung der Artikel Abhängigkeiten aus der Threadberechnung.

`output.testdequote`
Ist ein Beispiel für die Funktion des Algorithmus. Zeigt Original, Antwort und Antwort mit entfernten Zitaten (trotz Kammquoting).

Der eigentlich Algorithmus ist in der Bibliotheksfunktion `DAText::dequote` implementiert.

A.4.15 Zu Kapitel 5.4

`load.dump`
Ist ein Thread Testdatensatz vom TU Newsserver.

`0-simpletest.rb`
Nimmt einen 7 Artikel großen Thread der Testdaten und berechnet diesen
(siehe `0-simpletest.rb.output` für ein mögliches Ergebnis)

`0-addremovetext.rb`
Führt einen Einfüge/Entfernen Test, in beliebiger Reihenfolge, durch und vergleicht am Ende ob die berechneten Threads jeweils ident sind.

`load.dump.debugenabled`
Der Berechnungsvorgang der TU Testdaten bei Aktivierung aller Debugmeldungen.

`output.tree.at.linux`
Die errechnete Struktur von `at.linux`.

`thread-bygroup.rb`
Die Durchführung der Threadrekonstruktion über die gesamte Datenbasis, Gruppenweise.

`thread-bygroup.rb.output-de`
Beispielausgaben für den Lauf über `de.*`

`threads-uniq.rb`
Korrektur der generierten Threadlevel Datenfiles um Duplikate, die entstehen, wenn ein Thread die Gruppe(n) wechselt.

`stat-threadlevel.rb`

Generiert die Statistikdaten für 5.3

Der Threading Algorithmus wurde in der Bibliothek `DAThread` implementiert.

A.4.16 Zu Kapitel 5.5

`nilsimsa-20050418.tar.gz`
Ist eine für diese Diplomarbeit durchgeführte Ruby Portierung des in C geschriebenen Originalprogrammes ([Nils02]) von Nilsimsa. Diese Entwicklung wurde der Ruby Gemeinde unter
`http://rubyforge.org/projects/nilsimsa/`
zur Verfügung gestellt.

`nils-ofbodys.rb`
Berechnet alle Nilsimsa Werte aller Artikel Bodys.

`stat-nils.rb`
Generiert die Statistik für Abbildung 5.4.

`nils-n-path-spamdetect.rb`
Implementiert die Spamwellendetektion unter Berücksichtigung des `Path` Eintrages.

`nils-find-language.rb`
Kategorisiert die gefundenen Spamverdächtigen nach Sprache.

`nils-list-spamsubjects.rb`
Listet die `Subject`s der Spamverdächtigen.

`plot-nilsimsa-plane.rb`
Plottet das Spamdetektor Ergebnis in Abbildung 5.5

`CBspam.rb`
Ein Nilsimsa Spamdetektor als TUGnews/Cancelbot Submodul.

`dictcon.rb`
Konvertiert die `dict-wordlist...txt` Wörterbuch Files in ein einfacheres `dict.xx` Format.

A.4.17 Zu Kapitel 5.6

`classify-words`
Klassifiziert die Wörter der Artikeltexte in `at.linux`

Die `worddump...` Files sind die Ergebnisse.

A.4.18 Zu Anhang B

`plot-plane.rb`
Zeichnet Abbildung B.2.

`grouptable.rb`
Generiert die Tabelle in Anhang B.4.

A.4.19 Bibliotheken

Funktionen, die von mehreren Skripts verwendet werden, sind in Bibliotheksmodule zusammengefasst.

`DA.rb` stellt das Hauptmodul dar, welches alle Untermodule inkludiert. Es wird von jedem Skript per „require 'DA'“ am Anfang inkludiert und sollte sich somit im Suchpfad befinden.

DAdecode

Beinhaltet die Funktionen `decode_utf8` zur Dekodierung von „UTF-8“ Text und `decodeqp` zur Dekodierung von „quoted-printable“ Text.

Die Funktion `render_body` konvertiert einen Artikelbody soweit wie es möglich ist in ASCII Klartext.

DAdict

Beinhaltet nur eine Ladefunktion für die `dict` Wörterbücher und stellt diese als Objekte `de` und `en` zur Verfügung.

DAheader

Ist eine Sammlung von Hilfsroutinen zur Verarbeitung von Artikelheadern. Diese beinhaltet Routinen zum Einlesen und Aufteilen eines Headers und spezifische Funktionen für bestimmte Header Bestandteile.

Die Funktion `guess_path` ist die Implementierung des in Kapitel 5.1.3 beschriebenen `Path` Schätzalgorithmus.

DAindex

Offeriert ein Datenbankobjekt, welches mit Hilfe von simplen ASCII Indexdateien ein Objekt in einem großen Datenfile wiederfindet. Eine Indexdatei hat die Form:
`YYYYMMTTSSMMSSx1234567890 Position Länge`
`YYYYMMTTSSMMSSx1234567890 Position Länge`
`YYYYMMTTSSMMSSx1234567890 Position Länge`
`find_id` liefert die Zeile im Indexfile, während `load` das gesamte gesuchte Objekt retourniert.

DAplot

Stellt Routinen zur Verfügung, die eine Hilfestellung bei der Generierung der „Ebenen“ Grafiken (Abbildung 5.5 und B.2). Im ersten Abschnitt sind dies Routinen zum Handling von Datenarrays, im zweiten Abschnitt Plotroutinen für die Grafikgenerierung.

DAtext

Implementiert in `dequote` (und ein paar Unteroutinen) den in Kapitel 5.3.3 beschriebenen Algorithmus zur Zitatentfernung.

DAthread

Ist die Implementation des in Kapitel 5.4.3 vorgestellten Threading Algorithmus.

Die Funktion `add` übernimmt einen neuen Artikel als Parameter (MessageID, Zeitstempel, Gruppennamen, Referenzen) und fügt diesen in die Baumstruktur ein.

Die Funktion `remove` (MessageID) entfernt einen Artikel aus dem Wissen.

`calctree` berechnet aus einem gegebenen Startpunkt (Artikelnummer, Starttiefe, Gruppe) ein Array mit der Struktur des Threads, welches wiederum mittels `drawtree` in einer ASCII Baumdarstellung veranschaulicht werden kann.

Für ein Beispiel der genauen Aufrufsyntax (und den Gebrauch der Objektvariablen zur Auffindung von Threadwurzeln) wird auf `0-simpletest.rb` in der Skriptsammlung verwiesen.

DAutil

`mean` berechnet den Mittelwert eines Arrays und `only_chars` testet ob ein String nur Buchstaben beinhaltet.

DAwords

Ist eine leicht modifizierte Version eines Modules des `classifier` Paketes und bietet eine Konvertierungsfunktion Text ->Hash von Wörtern.

DAyears

Generiert ein Array aller Tage von 1.1.2003 bis 31.12.2004, entweder als Zahlen oder Strings.

B Ergänzende Tabellen und Graphen

Zur Vervollständigung finden sich auf den folgenden Seiten Tabellen und Graphen die im Haupttext eher störend viel Platz eingenommen hätten, aber für manche (ergänzende) Details doch von Interesse sind.

B.1 Artikelumsatz at.*

In Kapitel 4.1.4 wurde nur der Graph für `de.*` besprochen. Hier der Graph für `at.*`:

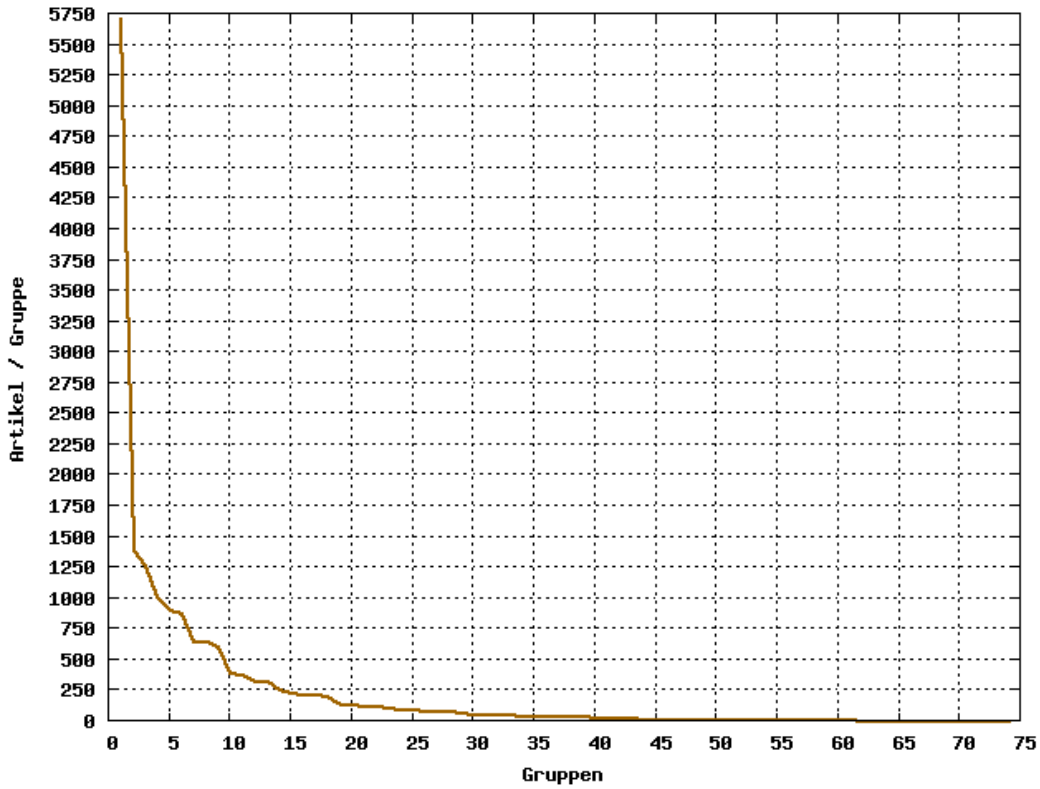

Abbildung B.1: `at.*` – durchschnittlicher Artikelumsatz pro Gruppe in 28 Tagen

Top 1: 32,6%, Top 2: 40,5%, Top 3: 47,7%, Top 5: 58,6%, Top 10: 76,6%

B.2 Aktivität global

Auf der y-Achse sind die Gruppen markiert, auf der x-Achse wie üblich die Zeit (die untersten Pixel sind die Wochenmarkierungen, darüber die Monatsmarkierungen und in der Mitte eine Jahreswechselmarkierung).

Ein Pixel stellt die Artikelanzahl an diesem Tag in einer Gruppe dar, je heller desto höher war der Umsatz.

Sichtbar als vertikale dunklere Streifen, sprich weniger Artikelumsatz, sind im Wesentlichen die Wochenenden, der Jahreswechsel und die Osterferien.

Spamwellen sind nicht sichtbar, siehe dazu Kapitel 5.5.4.

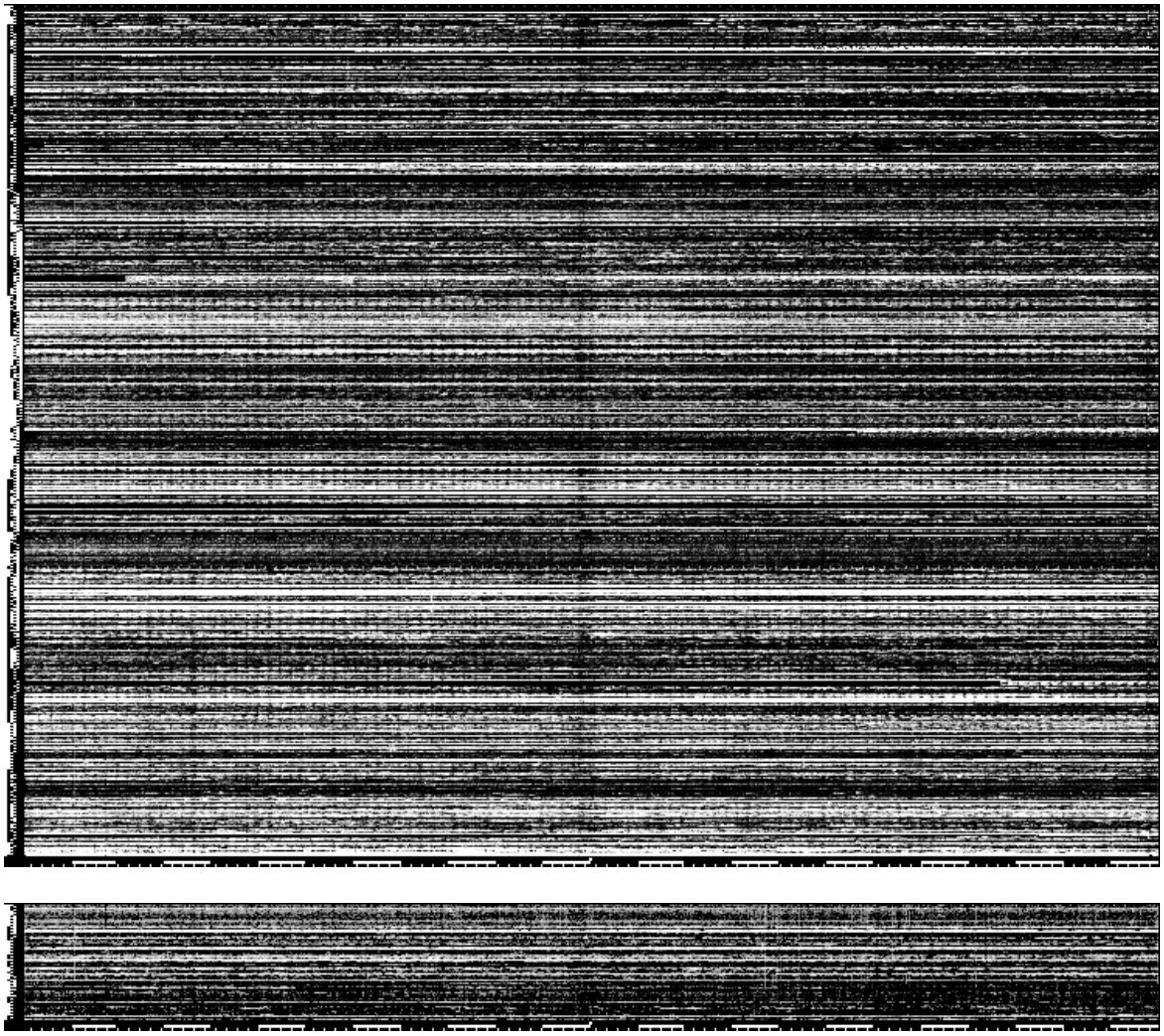

Abbildung B.2: Gesamtübersicht Aktivität, oben `de.*`, unten `at.*`

B.3 Header/Body Größe at.*

Siehe auch Kapitel 5.3.4.

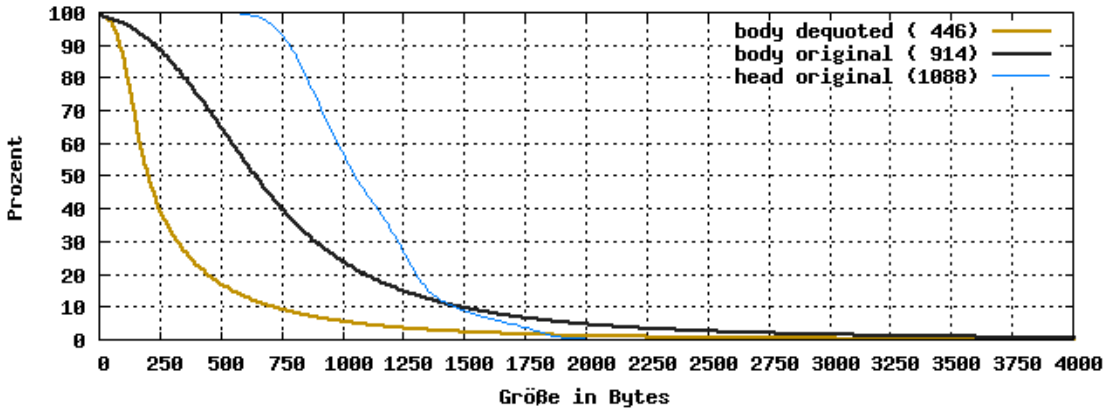

Abbildung B.3: `at.*` – Prozent der Header/Bodys kleiner x Bytes

`at.*` schrumpft von 394Mb Text auf 193Mb.
Die durchschnittliche Artikelgröße schrumpft von 914 auf 446 Bytes.
1532 Artikel sind größer 10kb und belegen 41,6Mb, also rund ein Fünftel der Restgröße.

B.4 Gruppenliste

Anbei noch eine Komplettliste der untersuchten Gruppen, plus zusätzlicher statistischer Werte je Gruppe.

- Artikelanzahl
  Artikel pro Gruppe. Der Prozentsatz stellt den Anteil an der Artikelzahl der jeweiligen Hierachie dar.
- AProfil
  Gruppenprofil der Autoren in dieser Gruppe – siehe Kapitel 5.2.3.
  Die 5 Spalten enthalten den durchschnittlichen Prozentwert für die Monatslinien 1, 6, 12, 18 und 23. Ein „?“ bedeutet, es waren nicht genügend Daten vorhanden um einen Wert zu ermitteln.

Gruppe	Artikelanzahl		GProfil 1	6	12	18	23
de.admin.infos	1052	0.013%	12	?	?	?	?
de.admin.lists	73	0.001%	50	?	?	?	?
de.admin.misc	298	0.004%	60	86	?	?	?
de.admin.net-abuse.announce	244	0.003%	57	36	?	?	?
de.admin.net-abuse.mail	59452	0.714%	15	45	66	80	94
de.admin.net-abuse.misc	1064	0.013%	43	90	?	?	?
de.admin.net-abuse.news	7070	0.085%	24	52	76	97	?
de.admin.news.announce	649	0.008%	19	57	84	?	92
de.admin.news.groups	32111	0.385%	17	47	63	79	96
de.admin.news.misc	4452	0.053%	12	41	69	?	95
de.admin.news.nocem	614	0.007%	78	?	?	?	?
de.admin.news.regeln	9188	0.110%	11	34	55	89	97
de.alt.0d	67	0.001%	71	100	?	?	?
de.alt.admin	14377	0.173%	23	52	73	85	96
de.alt.anime	16132	0.194%	12	39	52	73	92
de.alt.arnooo	5020	0.060%	18	38	60	?	?
de.alt.astrologie	14317	0.172%	31	63	81	93	98
de.alt.augenoptik	2485	0.030%	40	75	87	?	100
de.alt.auto.smart	4685	0.056%	30	58	76	83	97
de.alt.comics	1009	0.012%	48	83	?	?	?
de.alt.comm.datentausch-dienste	10168	0.122%	28	63	82	92	?
de.alt.comm.mgetty	1648	0.020%	43	75	?	93	?
de.alt.comp.cygwin+co	1473	0.018%	46	76	88	?	?
de.alt.comp.emulatoren	350	0.004%	64	92	?	?	?
de.alt.comp.sap-r3	11965	0.144%	20	61	78	92	98
de.alt.comp.the-bat	3958	0.048%	45	75	?	?	?
de.alt.dateien.misc	4600	0.055%	59	63	?	?	?
de.alt.dummschwatz	121211	1.455%	35	55	68	?	?
de.alt.etc.auktionshaeuser	104407	1.253%	16	49	70	86	96
de.alt.etc.koerperpflege	2318	0.028%	28	80	96	?	?
de.alt.etc.wunschgewicht	1097	0.013%	35	?	?	?	?
de.alt.fan.aldi	91020	1.093%	20	56	75	87	96
de.alt.fan.bluemchen	859	0.010%	57	?	?	?	?
de.alt.fan.boehse-onkelz	1084	0.013%	31	62	?	?	?
de.alt.fan.comedy	42	0.001%	86	?	?	?	?
de.alt.fan.die-aerzte	1857	0.022%	28	72	84	?	100
de.alt.fan.fastfood	7790	0.094%	15	47	65	82	96
de.alt.fan.fruehstyxradio	900	0.011%	29	59	?	?	?
de.alt.fan.furry	636	0.008%	34	74	?	?	?
de.alt.fan.haraldschmidt	23944	0.287%	19	52	73	86	95
de.alt.fan.harry-potter	12234	0.147%	18	47	69	88	?
de.alt.fan.helgeschneider	40	0.000%	86	100	?	?	?
de.alt.fan.konsumterror	18716	0.225%	19	52	71	88	96
de.alt.fan.misc	152	0.002%	62	?	?	?	?
de.alt.fan.pluesch	32552	0.391%	20	36	50	?	75
de.alt.fan.pratchett	1356	0.016%	24	69	?	95	?
de.alt.fan.prince	2440	0.029%	15	43	62	79	96
de.alt.fan.rrr	2380	0.029%	14	50	84	?	?
de.alt.fan.stefan-raab	2134	0.026%	32	63	86	?	?
de.alt.fan.tabak	47807	0.574%	9	31	55	73	88
de.alt.fan.tastische4	603	0.007%	28	54	82	?	?
de.alt.fan.tolkien	10596	0.127%	17	47	64	76	94
de.alt.flame	1792	0.022%	43	71	86	?	?
de.alt.folklore.computer	19311	0.232%	11	39	61	80	95
de.alt.folklore.ddr	32950	0.396%	13	38	49	66	84
de.alt.folklore.urban-legends	25218	0.303%	16	44	65	80	94
de.alt.folklore.usenet	2200	0.026%	20	59	81	?	100
de.alt.folklore.west-berlin	12	0.000%	100	?	?	?	?
de.alt.games.half-life	4370	0.052%	25	63	?	94	?
de.alt.games.konsolen	5978	0.072%	28	61	79	?	?
de.alt.games.linux	1027	0.012%	42	?	?	?	?
de.alt.games.pbem	607	0.007%	44	?	?	?	?
de.alt.games.quake	1541	0.018%	35	67	85	92	?
de.alt.games.schach	5441	0.065%	20	57	81	94	100
de.alt.games.unreal	2709	0.033%	32	71	86	95	?
de.alt.gblf	1794	0.022%	35	59	79	?	?
de.alt.gruppenkasper	97601	1.172%	34	66	78	85	91
de.alt.hoerfunk	4501	0.054%	27	59	80	?	98
de.alt.jugendschutz	512	0.006%	64	97	?	?	?
de.alt.mud	1028	0.012%	24	72	?	100	?
de.alt.music.alternative	75	0.001%	59	?	?	?	?
de.alt.music.hiphop	16141	0.194%	22	45	67	78	?
de.alt.music.jazz	2394	0.029%	34	65	81	89	?
de.alt.music.lyrics	350	0.004%	74	?	?	?	?
de.alt.music.metal	17648	0.212%	17	43	61	80	92
de.alt.naturheilkunde	33588	0.403%	32	64	78	89	98
de.alt.netdigest	3378	0.041%	14	47	70	90	98
de.alt.paranormal	4857	0.058%	38	71	89	95	?

Gruppe	Artikelanzahl		GProfil 1	6	12	18	23
de.alt.radio-scanner	6537	0.078%	28	66	83	92	?
de.alt.rec.ascii-art	1494	0.018%	28	57	84	?	?
de.alt.rec.digitalfotografie	132245	1.588%	20	59	79	90	97
de.alt.rec.fantasy	2000	0.024%	23	57	79	?	?
de.alt.rec.getraenke	3545	0.043%	28	65	84	96	?
de.alt.recovery.scientist	2122	0.025%	17	81	?	?	?
de.alt.recovery.webauthor	19611	0.235%	13	33	55	77	94
de.alt.sci.ergonomie	204	0.002%	56	?	?	?	?
de.alt.sci.geschichte-spekulativ	1583	0.019%	32	66	90	?	?
de.alt.soc.anarchie	476	0.006%	41	?	93	?	?
de.alt.soc.antifa	1305	0.016%	37	73	95	?	?
de.alt.soc.aufmerksamkeitsdefizit	2381	0.029%	22	59	?	?	?
de.alt.soc.knigge	9717	0.117%	21	56	82	91	?
de.alt.soc.krieg	91	0.001%	79	?	?	?	?
de.alt.soc.punk	5501	0.066%	22	59	48	73	?
de.alt.soc.tierrechte	991	0.012%	40	76	?	?	?
de.alt.soc.transgendered	1960	0.024%	36	56	79	?	?
de.alt.soc.verschwoerung	51410	0.617%	23	59	77	87	95
de.alt.soc.wtc-attentat	11	0.000%	100	?	?	?	?
de.alt.sources.linux-patches	2	0.000%	?	?	?	?	?
de.alt.sport.mountain-bike	9973	0.120%	24	64	83	97	?
de.alt.sport.tischtennis	1153	0.014%	21	55	79	?	?
de.alt.sport.winter	950	0.011%	50	78	?	?	?
de.alt.sysadmin.recovery	83187	0.999%	8	28	47	68	87
de.alt.talk.dunkle-seite	35337	0.424%	17	?	49	64	?
de.alt.talk.kasper	21199	0.254%	15	23	44	57	82
de.alt.talk.unmut	35855	0.430%	20	47	66	80	92
de.alt.technik.gps	6233	0.075%	33	76	89	?	?
de.alt.technik.misc	2737	0.033%	33	64	78	93	?
de.alt.technik.waffen	15085	0.181%	19	48	70	81	96
de.alt.test	27719	0.333%	64	91	95	?	98
de.alt.tv.mash	790	0.009%	29	65	84	?	?
de.alt.tv.reality-shows	3370	0.040%	25	74	?	?	?
de.alt.tv.reality-soap	30	0.000%	79	?	?	?	?
de.alt.tv.soaps	32	0.000%	82	?	?	?	?
de.alt.tv.wissenschaft	25	0.000%	75	?	?	?	?
de.alt.ufo	5062	0.061%	33	69	77	84	?
de.alt.umfragen	836	0.010%	51	83	83	?	?
de.alt.wg-geschichten	616	0.007%	42	70	?	?	?
de.alt.windsurfen	2036	0.024%	26	67	89	97	?
de.answers	3252	0.039%	2	3	?	21	50
de.comm.abuse	5091	0.061%	22	57	77	90	98
de.comm.anbieter.announce	491	0.006%	24	?	?	?	?
de.comm.anbieter.festnetz.misc	5069	0.061%	26	54	76	85	95
de.comm.anbieter.festnetz.tarife	3664	0.044%	27	63	78	?	?
de.comm.anbieter.misc	545	0.007%	50	81	?	?	?
de.comm.anbieter.mobil	34574	0.415%	20	53	71	85	94
de.comm.chatsystems	2827	0.034%	31	61	74	?	?
de.comm.funk.cb	4427	0.053%	27	68	?	?	?
de.comm.funk.misc	1780	0.021%	46	76	?	96	?
de.comm.geraete.analog.misc	1618	0.019%	47	81	?	?	?
de.comm.geraete.analog.modem	2804	0.034%	48	80	88	?	97
de.comm.geraete.isdn.computer	2392	0.029%	46	85	93	?	?
de.comm.geraete.isdn.misc	3312	0.040%	46	79	89	93	?
de.comm.geraete.isdn.tk-anlage	17929	0.215%	34	70	83	91	96
de.comm.geraete.misc	480	0.006%	62	95	?	?	?
de.comm.geraete.mobil.misc	10235	0.123%	32	75	91	97	100
de.comm.geraete.mobil.nokia	21411	0.257%	31	75	89	97	98
de.comm.geraete.mobil.pager	2430	0.029%	19	33	?	78	?
de.comm.geraete.mobil.siemens	21367	0.256%	30	73	89	95	99
de.comm.ham	36468	0.438%	17	49	70	82	93
de.comm.infosystems.misc	149	0.002%	68	100	?	?	?
de.comm.infosystems.suchmaschinen	7708	0.093%	20	51	68	90	?
de.comm.infosystems.www.authoring.misc	41918	0.503%	18	51	70	82	93
de.comm.infosystems.www.browsers	1534	0.018%	49	89	100	?	?
de.comm.infosystems.www.pages.announce	166	0.002%	71	?	?	?	?
de.comm.infosystems.www.pages.misc	8019	0.096%	23	54	77	?	?
de.comm.infosystems.www.servers	5379	0.065%	35	76	91	100	?
de.comm.internet.commerce	2083	0.025%	39	73	?	97	100
de.comm.internet.infrastruktur	4124	0.050%	18	46	66	87	?
de.comm.internet.misc	6576	0.079%	37	71	83	92	?
de.comm.misc	524	0.006%	64	?	?	?	?
de.comm.protocols.misc	203	0.002%	61	90	?	?	?
de.comm.protocols.tcp-ip	1970	0.024%	31	68	84	?	100
de.comm.provider.mail	8756	0.105%	25	61	84	96	?
de.comm.provider.metronet	9846	0.118%	10	19	38	?	79
de.comm.provider.misc	11733	0.141%	27	62	82	93	?
de.comm.provider.status	2066	0.025%	33	83	96	?	?

Gruppe	Artikelanzahl		GProfil 1	6	12	18	23
de.comm.provider.suche	2336	0.028%	44	79	96	?	?
de.comm.provider.t-online	15201	0.182%	32	64	79	90	97
de.comm.provider.tarife	2696	0.032%	40	80	89	100	?
de.comm.provider.usenet	3498	0.042%	25	59	80	90	?
de.comm.provider.webspace	10890	0.131%	29	65	83	95	99
de.comm.software.40tude-dialog	22203	0.267%	17	59	81	?	?
de.comm.software.browser.internet-explorer	1417	0.017%	52	83	?	?	?
de.comm.software.browser.misc	755	0.009%	48	76	95	?	?
de.comm.software.crosspoint	8273	0.099%	11	33	?	69	88
de.comm.software.forte-agent	7109	0.085%	19	55	70	?	95
de.comm.software.gnus	7872	0.094%	9	38	70	86	97
de.comm.software.groupware	865	0.010%	38	81	?	?	?
de.comm.software.janaserver	1244	0.015%	34	76	?	?	?
de.comm.software.mailreader.misc	4712	0.057%	31	63	84	93	?
de.comm.software.mailreader.pegasus	5000	0.060%	28	66	83	?	96
de.comm.software.mailreader.the-bat	4391	0.053%	18	64	85	?	?
de.comm.software.mailserver	15997	0.192%	24	64	79	91	95
de.comm.software.misc	1618	0.019%	53	85	90	100	?
de.comm.software.mozilla.browser	38730	0.465%	25	62	80	90	?
de.comm.software.mozilla.mailnews	37455	0.450%	25	65	83	93	?
de.comm.software.mozilla.misc	18239	0.219%	22	58	77	88	?
de.comm.software.mozilla.nightly-builds	6381	0.077%	15	46	57	79	?
de.comm.software.newsreader	17222	0.207%	22	55	72	83	94
de.comm.software.newsserver	3078	0.037%	28	63	85	?	100
de.comm.software.outlook-express	12754	0.153%	42	71	81	87	92
de.comm.software.webserver	856	0.010%	51	98	?	?	?
de.comm.technik.dsl	31984	0.384%	32	70	85	92	98
de.comm.technik.isdn	5251	0.063%	40	73	83	91	?
de.comm.technik.misc	625	0.008%	57	83	96	?	?
de.comm.technik.mobil	2807	0.034%	38	74	90	96	100
de.comm.uucp	373	0.004%	46	95	?	?	?
de.comp.advocacy	1339	0.016%	52	81	93	?	?
de.comp.audio	14516	0.174%	33	68	84	90	98
de.comp.cad	5049	0.061%	28	60	79	91	95
de.comp.datenbanken.misc	6395	0.077%	34	68	83	91	100
de.comp.datenbanken.ms-access	18541	0.223%	31	68	83	89	96
de.comp.datenbanken.mysql	19856	0.238%	36	77	89	94	97
de.comp.editoren	5733	0.069%	20	58	72	89	?
de.comp.gnu	1384	0.017%	40	81	?	?	100
de.comp.graphik	6674	0.080%	46	78	92	?	?
de.comp.hardware.announce	297	0.004%	32	?	?	?	?
de.comp.hardware.cpu+mainboard.amd	33630	0.404%	28	67	84	93	?
de.comp.hardware.cpu+mainboard.intel	15011	0.180%	31	68	82	91	95
de.comp.hardware.cpu+mainboard.misc	12315	0.148%	32	64	79	90	96
de.comp.hardware.cpu+mainboard.uebertakten	4206	0.050%	33	76	?	?	?
de.comp.hardware.drucker	17745	0.213%	37	76	88	96	99
de.comp.hardware.graphik	18094	0.217%	34	72	86	95	99
de.comp.hardware.kuehlung+laermdaemmung	25088	0.301%	21	61	83	92	?
de.comp.hardware.laufwerke.brenner	56656	0.680%	24	65	80	90	96
de.comp.hardware.laufwerke.cd+dvd	7351	0.088%	45	79	88	95	99
de.comp.hardware.laufwerke.festplatten	35700	0.429%	31	70	84	92	96
de.comp.hardware.laufwerke.misc	2651	0.032%	43	77	86	?	98
de.comp.hardware.misc	25690	0.308%	33	71	85	93	98
de.comp.hardware.monitore	7198	0.086%	44	85	92	?	?
de.comp.hardware.netzwerke.misc	16993	0.204%	36	71	84	92	97
de.comp.hardware.netzwerke.wireless	25206	0.303%	36	82	93	96	99
de.comp.hardware.scanner	3339	0.040%	50	85	94	?	?
de.comp.lang.assembler.misc	255	0.003%	67	100	?	?	?
de.comp.lang.assembler.x86	4164	0.050%	29	62	?	97	?
de.comp.lang.c	15252	0.183%	32	63	76	82	94
de.comp.lang.delphi.datenbanken	6183	0.074%	22	59	76	93	100
de.comp.lang.delphi.misc	40359	0.484%	16	50	70	84	95
de.comp.lang.delphi.non-tech	3713	0.045%	20	52	75	98	?
de.comp.lang.forth	564	0.007%	25	60	67	?	?
de.comp.lang.funktional	579	0.007%	35	75	100	?	?
de.comp.lang.iso-c++	15007	0.180%	21	51	70	83	92
de.comp.lang.java	91678	1.101%	21	59	77	88	95
de.comp.lang.javascript	22174	0.266%	43	83	91	?	?
de.comp.lang.misc	6266	0.075%	29	58	74	?	98
de.comp.lang.pascal	1258	0.015%	35	72	?	?	100
de.comp.lang.perl.cgi	3532	0.042%	40	76	?	94	97
de.comp.lang.perl.misc	10744	0.129%	28	65	83	93	96
de.comp.lang.php.datenbanken	9904	0.119%	36	75	87	93	97
de.comp.lang.php.installation	3221	0.039%	49	77	85	?	100
de.comp.lang.php.misc	65163	0.782%	23	66	84	93	98
de.comp.lang.php.netzprotokolle	1076	0.013%	46	83	93	?	?
de.comp.misc	3459	0.042%	46	81	88	94	?
de.comp.objekt	519	0.006%	44	87	100	?	?

Gruppe	Artikelanzahl		GProfil 1	6	12	18	23
de.comp.office-pakete.lotus-smartsuite	2306	0.028%	23	67	81	89	96
de.comp.office-pakete.misc	443	0.005%	68	?	?	?	?
de.comp.office-pakete.ms-office	2506	0.030%	55	91	?	?	?
de.comp.office-pakete.staroffice.install	1602	0.019%	35	70	86	?	94
de.comp.office-pakete.staroffice.misc	12129	0.146%	23	57	74	87	96
de.comp.office-pakete.staroffice.writer	11994	0.144%	23	54	72	87	97
de.comp.os.be	1438	0.017%	27	59	86	?	100
de.comp.os.misc	446	0.005%	57	89	?	?	?
de.comp.os.ms-windows.anwendungssoftware	2380	0.029%	46	82	?	?	?
de.comp.os.ms-windows.misc	38749	0.465%	30	69	82	90	97
de.comp.os.ms-windows.netzwerke	11097	0.133%	45	81	90	95	99
de.comp.os.ms-windows.programmer	3382	0.041%	33	71	90	?	?
de.comp.os.ms-windows.treiber	1300	0.016%	59	89	?	100	?
de.comp.os.msdos	4413	0.053%	27	55	70	?	?
de.comp.os.os2.advocacy	490	0.006%	30	?	?	?	?
de.comp.os.os2.apps	6007	0.072%	8	34	61	?	100
de.comp.os.os2.misc	4495	0.054%	10	41	64	86	100
de.comp.os.os2.networking	1871	0.022%	16	53	79	93	?
de.comp.os.os2.programmer	543	0.007%	22	71	?	?	?
de.comp.os.os2.setup	3000	0.036%	12	48	73	?	?
de.comp.os.unix.apps.gnome	5219	0.063%	27	60	78	92	100
de.comp.os.unix.apps.kde	21904	0.263%	19	61	83	94	98
de.comp.os.unix.apps.misc	5882	0.071%	19	55	74	91	?
de.comp.os.unix.bsd	11753	0.141%	15	47	69	85	98
de.comp.os.unix.discussion	3434	0.041%	22	60	78	97	?
de.comp.os.unix.linux.hardware	27458	0.330%	29	65	81	92	98
de.comp.os.unix.linux.infos	1426	0.017%	8	11	?	?	?
de.comp.os.unix.linux.isdn	4633	0.056%	39	79	88	91	100
de.comp.os.unix.linux.misc	110368	1.325%	21	58	77	88	97
de.comp.os.unix.linux.moderated	2586	0.031%	41	79	89	97	?
de.comp.os.unix.misc	2821	0.034%	32	69	85	94	100
de.comp.os.unix.networking.misc	9784	0.117%	26	61	83	90	98
de.comp.os.unix.networking.samba	8248	0.099%	38	79	89	?	98
de.comp.os.unix.programming	8219	0.099%	24	60	78	89	98
de.comp.os.unix.shell	9038	0.108%	24	62	82	91	99
de.comp.os.unix.sinix	187	0.002%	34	?	?	?	?
de.comp.os.unix.x11	4705	0.056%	29	68	84	95	?
de.comp.os.vms	1560	0.019%	17	52	69	?	100
de.comp.security.firewall	39361	0.473%	26	60	76	90	96
de.comp.security.misc	54820	0.658%	21	51	71	86	95
de.comp.security.virus	4630	0.056%	37	90	?	?	?
de.comp.software.announce	2697	0.032%	29	81	96	?	?
de.comp.software.graphik	2473	0.030%	34	77	?	?	?
de.comp.software.misc	4402	0.053%	46	80	90	94	97
de.comp.software.shareware	2431	0.029%	34	68	85	92	?
de.comp.standards	675	0.008%	37	65	?	?	?
de.comp.sys.amiga.archive	631	0.008%	48	?	?	?	?
de.comp.sys.amiga.comm	95	0.001%	55	?	?	?	?
de.comp.sys.amiga.misc	3243	0.039%	20	51	70	?	?
de.comp.sys.amiga.tech	1494	0.018%	22	64	91	?	?
de.comp.sys.atari	931	0.011%	27	68	86	?	?
de.comp.sys.handhelds.misc	3328	0.040%	43	79	?	98	?
de.comp.sys.handhelds.newton	768	0.009%	28	76	93	?	100
de.comp.sys.handhelds.palm-pilot	37036	0.445%	22	61	79	91	97
de.comp.sys.handhelds.psion	2875	0.035%	23	66	83	93	?
de.comp.sys.handhelds.windows-ce	8150	0.098%	31	76	91	99	?
de.comp.sys.mac.internet	15221	0.183%	14	41	62	83	95
de.comp.sys.mac.lokale-netze	9802	0.118%	21	56	79	88	92
de.comp.sys.mac.misc	115792	1.390%	11	34	53	71	91
de.comp.sys.mac.programmieren	2703	0.032%	21	60	77	93	100
de.comp.sys.mac.soc	8606	0.103%	9	35	62	80	93
de.comp.sys.misc	153	0.002%	86	100	?	?	?
de.comp.sys.next	634	0.008%	23	71	?	?	?
de.comp.sys.notebooks	56733	0.681%	26	70	85	94	98
de.comp.sys.novell	19445	0.233%	15	46	65	81	95
de.comp.text.dtp	7012	0.084%	19	53	73	86	95
de.comp.text.misc	368	0.004%	55	90	?	?	?
de.comp.text.ms-word	1996	0.024%	49	82	91	96	?
de.comp.text.tex	54590	0.655%	19	58	77	87	93
de.comp.text.xml	4730	0.057%	29	66	81	?	?
de.comp.tv+video	50933	0.611%	27	69	86	92	98
de.etc.bahn.announce	1333	0.016%	21	59	82	93	?
de.etc.bahn.bahnpolitik	11816	0.142%	11	37	55	78	93
de.etc.bahn.eisenbahntechnik	18831	0.226%	10	34	56	73	90
de.etc.bahn.historisch	6741	0.081%	12	42	68	88	97
de.etc.bahn.misc	59905	0.719%	10	32	51	70	86
de.etc.bahn.stadtverkehr	21212	0.255%	9	33	54	72	92
de.etc.bahn.tarif+service	51398	0.617%	11	34	53	71	89

Gruppe	Artikelanzahl		GProfil 1	6	12	18	23
de.etc.beruf.misc	737	0.009%	58	90	?	?	?
de.etc.beruf.selbstaendig	71858	0.863%	24	57	73	82	91
de.etc.fahrzeug.auto	110355	1.325%	21	53	71	83	94
de.etc.fahrzeug.misc	499	0.006%	70	98	?	?	?
de.etc.finanz.banken+broker	13967	0.168%	25	60	78	88	94
de.etc.finanz.boerse	4071	0.049%	29	75	?	?	?
de.etc.finanz.boerse.misc	14062	0.169%	33	67	81	?	?
de.etc.finanz.misc	22605	0.271%	28	62	77	87	96
de.etc.finanz.software	3832	0.046%	39	76	88	?	?
de.etc.handel.auktionshaeuser	187	0.002%	100	?	?	?	?
de.etc.handel.misc	7	0.000%	100	?	?	?	?
de.etc.handel.versandhaeuser	27	0.000%	100	?	?	?	?
de.etc.haushalt	35219	0.423%	22	58	80	89	98
de.etc.lists	250	0.003%	32	39	?	?	?
de.etc.militaer	8037	0.096%	22	58	86	?	?
de.etc.misc	345	0.004%	67	?	?	?	?
de.etc.notfallrettung	21534	0.259%	16	42	64	81	94
de.etc.schreiben.lyrik	11537	0.138%	24	57	74	83	95
de.etc.schreiben.misc	2035	0.024%	32	61	83	?	94
de.etc.schreiben.prosa	3837	0.046%	25	59	?	?	100
de.etc.selbsthilfe.angst	13055	0.157%	27	58	67	73	100
de.etc.selbsthilfe.gehoer	1071	0.013%	35	75	?	?	?
de.etc.selbsthilfe.misc	387	0.005%	66	?	?	?	?
de.etc.selbsthilfe.missbrauch	2101	0.025%	20	43	74	?	?
de.etc.sprache.deutsch	118759	1.426%	21	50	68	78	87
de.etc.sprache.klassisch	3119	0.037%	22	55	76	?	97
de.etc.sprache.misc	11229	0.135%	23	52	67	82	94
de.markt.arbeit.biete.it-berufe	386	0.005%	59	95	100	?	?
de.markt.arbeit.biete.misc	191	0.002%	73	97	?	?	?
de.markt.arbeit.d	11579	0.139%	38	71	83	96	?
de.markt.arbeit.suche	825	0.010%	60	93	?	?	?
de.markt.arbeit.vermittler	4866	0.058%	42	64	78	91	?
de.markt.buecher	2619	0.031%	41	82	?	?	?
de.markt.comm	3110	0.037%	34	76	94	97	?
de.markt.comp.handhelds	1317	0.016%	47	89	98	?	?
de.markt.comp.hardware.cpu+mainboard	6748	0.081%	26	67	85	97	100
de.markt.comp.hardware.graphik	2727	0.033%	35	81	96	?	?
de.markt.comp.hardware.laufwerke	4576	0.055%	27	71	93	?	?
de.markt.comp.hardware.misc	12270	0.147%	24	65	85	96	?
de.markt.comp.misc	6009	0.072%	30	76	91	?	100
de.markt.comp.software	3347	0.040%	38	79	?	?	?
de.markt.fahrzeug.auto	3273	0.039%	48	88	96	99	?
de.markt.fahrzeug.misc	857	0.010%	58	95	100	?	?
de.markt.misc	6387	0.077%	35	77	93	?	?
de.markt.musik	2760	0.033%	45	83	93	?	?
de.markt.spiele.computer	2488	0.030%	33	80	92	?	100
de.markt.spiele.misc	480	0.006%	65	?	?	?	?
de.markt.tiere	1070	0.013%	51	85	95	?	?
de.markt.wohnen	1988	0.024%	61	91	97	100	?
de.newusers.infos	2134	0.026%	?	?	18	?	?
de.newusers.questions	6219	0.075%	29	55	62	74	?
de.org.ccc	18572	0.223%	21	56	74	90	97
de.org.mensa	3311	0.040%	27	61	79	?	100
de.org.misc	98	0.001%	61	?	?	?	?
de.org.politik.misc	9290	0.112%	34	76	91	97	?
de.org.politik.spd	27091	0.325%	25	62	81	96	?
de.rec.alpinismus	12669	0.152%	17	50	70	85	96
de.rec.bodyart	2842	0.034%	24	56	77	85	?
de.rec.buecher	15859	0.190%	24	56	69	81	93
de.rec.denksport	11034	0.132%	24	55	74	88	?
de.rec.drachen	2988	0.036%	20	56	78	92	100
de.rec.fahrrad	115503	1.387%	17	49	67	79	93
de.rec.film.heimkino	27583	0.331%	22	59	78	90	98
de.rec.film.kritiken	360	0.004%	21	42	?	100	?
de.rec.film.misc	55021	0.660%	17	45	64	80	93
de.rec.fotografie	195848	2.351%	16	44	64	78	91
de.rec.garten	39986	0.480%	24	61	75	87	95
de.rec.heimwerken	66466	0.798%	20	57	75	86	94
de.rec.hoerspiel	4900	0.059%	30	67	85	87	97
de.rec.kunst.misc	7514	0.090%	41	65	73	82	90
de.rec.kunst.theater	1367	0.016%	35	61	77	?	?
de.rec.luftfahrt	14385	0.173%	16	52	72	86	96
de.rec.mampf	75041	0.901%	20	55	73	86	93
de.rec.misc	523	0.006%	38	52	?	?	?
de.rec.modelle.bahn	84246	1.011%	12	38	57	74	88
de.rec.modelle.misc	33313	0.400%	17	46	65	81	91
de.rec.motorrad	74885	0.899%	15	43	65	79	95
de.rec.motorroller	6546	0.079%	27	69	85	94	100

Gruppe	Artikelanzahl		GProfil 1	6	12	18	23
de.rec.musik.elektronisch	2952	0.035%	39	71	79	90	97
de.rec.musik.hifi	34458	0.414%	25	59	76	87	94
de.rec.musik.klassik	22912	0.275%	21	48	64	74	92
de.rec.musik.machen	37462	0.450%	19	48	66	80	89
de.rec.musik.misc	4795	0.058%	38	72	84	?	98
de.rec.musik.nachtleben	256	0.003%	50	?	?	?	?
de.rec.musik.recherche	16469	0.198%	22	60	80	92	98
de.rec.musik.rock+pop	10892	0.131%	25	57	77	88	97
de.rec.orakel	38	0.000%	?	?	?	?	?
de.rec.outdoors	13484	0.162%	22	60	78	89	96
de.rec.reisen.camping	18468	0.222%	22	56	75	86	97
de.rec.reisen.misc	40411	0.485%	23	59	74	86	96
de.rec.sammeln	1101	0.013%	51	91	?	?	?
de.rec.sf.babylon5.misc	9052	0.109%	11	36	51	?	81
de.rec.sf.misc	10452	0.125%	15	41	59	72	93
de.rec.sf.perry-rhodan	33649	0.404%	10	33	54	75	91
de.rec.sf.stargate	13141	0.158%	11	40	67	88	?
de.rec.sf.startrek.10vorne	5859	0.070%	24	43	?	89	100
de.rec.sf.startrek.deep-space-9	753	0.009%	25	77	100	?	?
de.rec.sf.startrek.enterprise	17179	0.206%	13	43	66	83	?
de.rec.sf.startrek.misc	5718	0.069%	18	49	69	79	94
de.rec.sf.startrek.technologie	1691	0.020%	21	54	75	?	?
de.rec.sf.startrek.voyager	455	0.005%	39	86	?	?	?
de.rec.sf.starwars	2604	0.031%	23	57	72	?	100
de.rec.spiele.brett+karten	2728	0.033%	34	70	86	?	?
de.rec.spiele.computer.action	24478	0.294%	18	52	69	87	95
de.rec.spiele.computer.adventure	5952	0.071%	25	59	77	90	100
de.rec.spiele.computer.lan-party	966	0.012%	51	80	?	?	?
de.rec.spiele.computer.misc	2562	0.031%	35	73	82	98	?
de.rec.spiele.computer.rpg	11836	0.142%	15	50	72	88	98
de.rec.spiele.computer.simulation	4938	0.059%	27	68	82	92	98
de.rec.spiele.computer.strategie	4741	0.057%	29	69	88	98	?
de.rec.spiele.computer.technik	2560	0.031%	29	65	?	?	97
de.rec.spiele.miniaturen	4239	0.051%	13	34	?	?	100
de.rec.spiele.misc	521	0.006%	53	?	?	?	?
de.rec.spiele.rpg.live	733	0.009%	33	72	84	?	?
de.rec.spiele.rpg.misc	10169	0.122%	13	41	61	76	97
de.rec.sport.budo	11254	0.135%	23	53	70	86	96
de.rec.sport.eishockey	1615	0.019%	27	61	?	95	?
de.rec.sport.fallschirm	4361	0.052%	15	51	73	90	?
de.rec.sport.fussball	46464	0.558%	14	35	52	72	93
de.rec.sport.gleitschirm	2665	0.032%	20	52	75	89	97
de.rec.sport.golf	5380	0.065%	18	49	71	?	98
de.rec.sport.inlineskating	4910	0.059%	21	52	69	85	?
de.rec.sport.laufen	47865	0.575%	17	49	71	86	?
de.rec.sport.laufen.misc	4951	0.059%	23	?	?	?	?
de.rec.sport.laufen.trainingswoche	3767	0.045%	12	?	?	?	?
de.rec.sport.laufen.veranstaltungen	2533	0.030%	18	?	?	?	?
de.rec.sport.misc	3297	0.040%	43	77	87	93	97
de.rec.sport.motorsport.formel1	21837	0.262%	12	38	58	73	95
de.rec.sport.motorsport.misc	6306	0.076%	13	36	56	?	92
de.rec.sport.motorsport.motorrad	3021	0.036%	19	46	68	87	?
de.rec.sport.paintball	548	0.007%	53	77	?	?	?
de.rec.sport.segeln	21269	0.255%	18	51	68	79	93
de.rec.sport.tauchen	51738	0.621%	14	46	66	79	94
de.rec.tanz	8060	0.097%	16	47	69	81	96
de.rec.tiere.aquaristik	91008	1.093%	17	53	75	87	96
de.rec.tiere.hunde	79093	0.949%	23	56	71	84	97
de.rec.tiere.katzen	59977	0.720%	19	51	69	80	91
de.rec.tiere.misc	3373	0.040%	33	69	84	94	100
de.rec.tiere.pferde	22943	0.275%	18	46	63	76	93
de.rec.tiere.ratten	2454	0.029%	28	58	89	95	?
de.rec.tiere.terraristik	2276	0.027%	29	70	83	?	?
de.rec.tiere.voegel	3617	0.043%	32	64	82	94	?
de.rec.tv.akte-x	854	0.010%	27	66	?	100	?
de.rec.tv.buffy	17520	0.210%	12	36	51	69	90
de.rec.tv.futurama	736	0.009%	35	79	?	?	?
de.rec.tv.lindenstrasse	28048	0.337%	12	37	54	69	88
de.rec.tv.misc	39942	0.479%	15	43	65	80	94
de.rec.tv.serien	15752	0.189%	15	46	63	76	92
de.rec.tv.simpsons	6528	0.078%	26	55	65	76	81
de.rec.tv.technik	21290	0.256%	33	73	86	91	97
de.sci.alternativ	4695	0.056%	26	80	?	?	?
de.sci.architektur	20287	0.244%	22	53	72	82	96
de.sci.astronomie	18168	0.218%	21	60	77	86	97
de.sci.biologie	8225	0.099%	30	63	82	91	98
de.sci.chemie	19207	0.231%	27	60	76	87	95
de.sci.electronics	87670	1.052%	16	48	66	81	93

Gruppe	Artikelanzahl		GProfil 1	6	12	18	23
de.sci.genealogie	3226	0.039%	31	71	85	91	96
de.sci.geo	1573	0.019%	35	73	?	96	?
de.sci.geschichte	35857	0.430%	24	55	73	86	96
de.sci.informatik.ki	1062	0.013%	47	72	?	?	?
de.sci.informatik.misc	6052	0.073%	28	69	84	93	98
de.sci.ing.elektrotechnik	23669	0.284%	23	58	74	84	94
de.sci.ing.misc	6449	0.077%	26	62	76	86	97
de.sci.mathematik	60946	0.732%	25	60	75	84	93
de.sci.medizin.allergie	964	0.012%	53	84	?	?	?
de.sci.medizin.diabetes	35523	0.426%	17	47	63	78	90
de.sci.medizin.misc	31421	0.377%	29	66	82	91	98
de.sci.medizin.pflege	1779	0.021%	41	74	88	?	?
de.sci.medizin.psychiatrie	10812	0.130%	37	69	81	93	?
de.sci.meteorologie	964	0.012%	41	75	?	?	?
de.sci.misc	1044	0.013%	47	82	93	?	?
de.sci.oekonomie	3865	0.046%	33	69	91	?	?
de.sci.paedagogik	969	0.012%	43	?	?	?	?
de.sci.philosophie	33946	0.408%	30	63	77	90	98
de.sci.physik	58983	0.708%	23	58	74	87	94
de.sci.politologie	1274	0.015%	47	79	88	91	?
de.sci.psychologie	18159	0.218%	37	74	86	92	98
de.sci.raumfahrt	19278	0.231%	18	47	70	84	97
de.sci.soziologie	1176	0.014%	58	?	?	?	?
de.sci.theologie	27736	0.333%	23	56	77	92	?
de.soc.arbeit	3536	0.042%	46	85	95	?	?
de.soc.datenschutz	2364	0.028%	40	77	89	?	?
de.soc.drogen	22784	0.274%	24	53	72	86	95
de.soc.familie.kinder	24728	0.297%	23	54	73	84	97
de.soc.familie.misc	1881	0.023%	52	84	?	?	?
de.soc.familie.vaeter	16326	0.196%	28	58	72	81	95
de.soc.gleichberechtigung	8754	0.105%	27	60	80	94	100
de.soc.handicap	881	0.011%	41	77	93	?	?
de.soc.jugendarbeit	922	0.011%	52	89	98	?	?
de.soc.kontakte.freizuegig	9308	0.112%	46	84	93	?	?
de.soc.kontakte.misc	1619	0.019%	59	91	96	?	?
de.soc.kultur.aegypten	444	0.005%	42	93	?	?	?
de.soc.kultur.japan	3318	0.040%	21	54	?	90	97
de.soc.kultur.misc	250	0.003%	62	?	?	?	?
de.soc.medien	790	0.009%	56	96	?	?	?
de.soc.menschenrechte	3698	0.044%	46	81	90	95	?
de.soc.misc	2086	0.025%	55	?	?	?	100
de.soc.netzkultur.misc	2257	0.027%	32	66	?	90	?
de.soc.netzkultur.umgangsformen	24991	0.300%	26	58	73	80	94
de.soc.pflichtdienste	5367	0.064%	25	58	83	?	100
de.soc.politik.misc	217586	2.612%	24	59	75	86	93
de.soc.politik.texte	4336	0.052%	32	70	87	?	100
de.soc.recht.arbeit+soziales	33654	0.404%	29	65	81	89	96
de.soc.recht.datennetze	23687	0.284%	23	56	71	86	99
de.soc.recht.familie+erben	2484	0.030%	41	76	87	?	?
de.soc.recht.marken+urheber	12499	0.150%	30	61	76	87	96
de.soc.recht.misc	130389	1.565%	23	58	75	86	94
de.soc.recht.steuern+buchfuehrung	21666	0.260%	31	68	82	90	98
de.soc.recht.strafrecht	15990	0.192%	28	59	74	87	97
de.soc.recht.strassenverkehr	45906	0.551%	18	50	71	83	95
de.soc.recht.wohnen	24283	0.292%	29	62	77	90	96
de.soc.schule	452	0.005%	100	?	?	?	?
de.soc.senioren	14871	0.179%	27	?	?	73	?
de.soc.studium	13487	0.162%	28	65	83	92	96
de.soc.studium.verbindungen	4559	0.055%	21	57	79	95	100
de.soc.subkultur.bdsm	5966	0.072%	30	61	80	90	96
de.soc.subkultur.gothic	17676	0.212%	14	38	56	75	92
de.soc.subkultur.misc	72	0.001%	91	?	?	?	?
de.soc.umwelt	12156	0.146%	28	55	73	?	94
de.soc.verkehr	26838	0.322%	23	52	72	85	95
de.soc.weltanschauung.buddhismus	14572	0.175%	32	60	77	83	?
de.soc.weltanschauung.christentum	73004	0.876%	21	55	72	84	94
de.soc.weltanschauung.islam	5094	0.061%	45	?	?	?	?
de.soc.weltanschauung.misc	3043	0.037%	44	80	100	?	?
de.soc.weltanschauung.scientology	18924	0.227%	27	59	79	89	95
de.soc.wirtschaft	17575	0.211%	32	66	73	86	95
de.soc.zensur	1081	0.013%	53	81	?	?	?
de.talk.bizarre	98466	1.182%	31	51	62	74	?
de.talk.jokes	20611	0.247%	20	57	78	89	97
de.talk.jokes.d	2593	0.031%	36	65	?	90	97
de.talk.jugend	38775	0.465%	34	44	56	77	91
de.talk.liebesakt	87078	1.045%	29	65	81	91	96
de.talk.misc	46863	0.563%	48	65	74	?	100
de.talk.romance	47501	0.570%	29	63	76	85	96

Gruppe	Artikelanzahl		GProfil 1	6	12	18	23
de.talk.tagesgeschehen	153659	1.845%	21	53	69	83	94
de.test	96010	1.153%	55	90	96	99	99
at.anzeigen.arbeitsmarkt	1294	0.282%	61	91	?	96	?
at.anzeigen.computer.mac	1129	0.246%	36	78	89	?	?
at.anzeigen.computer.pc	8337	1.820%	26	69	86	94	98
at.anzeigen.computer.sonstiges	3269	0.713%	34	77	92	?	?
at.anzeigen.fahrzeuge.auto	2685	0.586%	41	85	94	95	98
at.anzeigen.fahrzeuge.motorrad	1342	0.293%	47	88	?	?	?
at.anzeigen.fahrzeuge.sonstiges	703	0.153%	54	93	?	?	?
at.anzeigen.kontakte	830	0.181%	64	93	?	?	?
at.anzeigen.mitfahrboerse	425	0.093%	63	?	?	?	?
at.anzeigen.musik	951	0.208%	54	91	?	?	?
at.anzeigen.sonstiges	5528	1.207%	31	78	90	?	?
at.anzeigen.telekomm	1938	0.423%	41	82	91	95	?
at.anzeigen.veranstaltung	501	0.109%	48	86	?	?	?
at.anzeigen.wohnen	2057	0.449%	47	86	95	96	?
at.freizeit.auto	23034	5.027%	21	47	61	75	88
at.freizeit.film	529	0.115%	49	?	?	?	?
at.freizeit.motorrad	33145	7.234%	16	45	64	79	93
at.freizeit.nonsens	149151	32.554%	28	46	70	69	?
at.freizeit.rollenspiele	242	0.053%	59	86	?	?	?
at.freizeit.sf.sonstiges	117	0.026%	45	69	?	?	?
at.freizeit.sf.startrek	1397	0.305%	25	52	66	?	?
at.freizeit.sonstiges	16761	3.658%	21	46	49	62	91
at.freizeit.sport	280	0.061%	69	?	?	?	?
at.gesellschaft.finanzen	1383	0.302%	34	73	87	?	?
at.gesellschaft.humor	1963	0.428%	33	79	94	?	?
at.gesellschaft.kontakte	3485	0.761%	33	56	66	?	?
at.gesellschaft.notfalldienste	91	0.020%	68	?	?	?	?
at.gesellschaft.politik	23394	5.106%	21	50	70	81	90
at.gesellschaft.recht	26333	5.748%	18	48	67	81	92
at.gesellschaft.sonstiges	244	0.053%	67	?	?	?	?
at.gesellschaft.studium	146	0.032%	70	?	?	?	?
at.gesellschaft.zivildienst	403	0.088%	36	86	?	100	?
at.internet.breitband	5238	1.143%	25	61	81	?	98
at.internet.provider	10300	2.248%	18	51	73	86	95
at.internet.sonstiges	3144	0.686%	26	57	76	93	?
at.linux	36359	7.936%	20	52	75	86	94
at.medien.fernsehen	3414	0.745%	23	56	77	92	?
at.medien.print	53	0.012%	81	100	?	?	?
at.medien.radio	646	0.141%	31	62	?	100	?
at.medien.sonstiges	32	0.007%	91	?	?	?	?
at.region.graz	6539	1.427%	19	56	76	86	96
at.region.noe	8367	1.826%	18	44	54	?	?
at.region.steiermark	463	0.101%	47	93	?	?	?
at.sonstiges	1074	0.234%	28	63	?	?	?
at.telekomm.mobil	15706	3.428%	15	43	64	80	94
at.telekomm.sonstiges	2084	0.455%	23	60	77	95	?
at.telekomm.technik.isdn	374	0.082%	65	?	?	?	?
at.telekomm.technik.sonstiges	540	0.118%	51	?	100	?	?
at.test	9799	2.139%	56	91	96	98	?
at.tuwien.admins	223	0.049%	37	?	72	?	?
at.tuwien.cg	426	0.093%	69	95	?	?	?
at.tuwien.general	192	0.042%	57	?	?	?	?
at.tuwien.hardware	1132	0.247%	44	93	?	?	?
at.tuwien.infosys.rnue	2305	0.503%	41	84	?	?	?
at.tuwien.os.winnt	31	0.007%	80	?	?	?	?
at.tuwien.software	519	0.113%	56	91	?	?	?
at.tuwien.student	1197	0.261%	31	69	89	?	?
at.tuwien.tunet	411	0.090%	31	71	?	90	?
at.tuwien.zid.neuigkeiten	394	0.086%	23	35	64	?	?
at.univie.club	38	0.008%	88	?	?	?	?
at.univie.edv	413	0.090%	46	87	?	?	?
at.univie.general	119	0.026%	84	?	?	?	?
at.univie.publizistik	5914	1.291%	16	56	66	?	91
at.usenet.announce	78	0.017%	57	?	?	?	?
at.usenet.cancel-reports	100	0.022%	100	?	?	?	?
at.usenet.einsteiger	894	0.195%	31	67	84	?	92
at.usenet.gruppen	2490	0.543%	16	54	78	?	?
at.usenet.infos	169	0.037%	24	?	?	?	70
at.usenet.missbrauch	313	0.068%	43	86	?	?	?
at.usenet.schmankerl	106	0.023%	34	78	?	?	?
at.usenet.sonstiges	766	0.167%	28	76	76	?	?
at.verkehr.bahn	16887	3.686%	11	39	59	75	91
at.verkehr.sonstiges	132	0.029%	48	?	?	?	?
at.verkehr.strasse	5696	1.243%	15	43	60	83	?

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[—] Alle im Text vorkommenden Links zu Internetquellen wurden zum Zeitpunkt der Abgabe der Arbeit, Anfang September 2005, als gültig verifiziert.

[Harr95] Harrison, Mark
The USENET Handbook
May 1995, O'Reilly & Associates, Inc.
ISBN: 1-56592-101-1

[Jwz02] Zawinski, Jamie
Message threading
`http://www.jwz.org/doc/threading.html`

[Nils02] cmeclax
The Nilsimsa Handbook (and code)
`http://ixazon.dynip.com/~cmeclax/nilsimsa.html`

[Nils04] E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, P. Samarati
An Open Digest-based Technique for Spam Detection
Proc. of the 2004 International Workshop on Security
in Parallel and Distributed Systems
`http://seclab.dti.unimi.it/Papers/pdcs04.pdf`

[Thom04] Thomas, Dave (with Chad Fowler and Andy Hunt)
Programming Ruby
Oct 2004, The Pragmatic Programmers LLC
ISBN: 0-9745140-5-5
`http://www.pragmaticprogrammer.com/titles/ruby/index.html`

[Salz92] Salz, Richard
InterNetNews: Usenet transport for Internet sites.
Summer 1992 Usenix conference
`ftp://ftp.isc.org/isc/inn/extra-docs/innusenix.pdf`

[Sankar04] Pal, Sankar K.; Mitra, Pabitra.
Pattern Recognition Algorithms for Data Mining
2004, CRC Press LLC
ISBN: 1-58488-457-6

[Spaf86] Spafford, Gene
Comments on Reorganization
`http://groups.google.com/group/net.news.group/msg/d21a82a9ba7596c8`

[Volker] Gringmuth, Volker
Volkers Usenet-Seiten
`http://einklich.net/usenet/`

[RFC977] RFC 977: *Network News Transport Protocol* (Feb. 1986)
`ftp://ftp.rfc-editor.org/in-notes/rfc977.txt`

[RFC1036] RFC 1036: *Standard for Interchange of USENET Messages* (Dec. 1987)
`ftp://ftp.rfc-editor.org/in-notes/rfc1036.txt`

[RFC1321] RFC 1321: *The MD5 Message-Digest Algorithm* (Apr. 1992)
`ftp://ftp.rfc-editor.org/in-notes/rfc1321.txt`

[RFC2047] RFC 2047: *MIME Part Three:*
Message Header Extensions for Non-ASCII Text (Nov. 1996)
`ftp://ftp.rfc-editor.org/in-notes/rfc2047.txt`

[RFC2076] RFC 2076: *Common Internet Message Headers* (Feb. 1997)
`ftp://ftp.rfc-editor.org/in-notes/rfc2076.txt`

[RFC2646] RFC 2646: *The Text/Plain Format Parameter* (Aug. 1999)
`ftp://ftp.rfc-editor.org/in-notes/rfc2646.txt`

[RFC2822] RFC 2822: *Internet Message Format* (Apr. 2001)
`ftp://ftp.rfc-editor.org/in-notes/rfc2822.txt`

[RFC2980] RFC 2980: *Common NNTP Extensions* (Oct. 2000)
`ftp://ftp.rfc-editor.org/in-notes/rfc2980.txt`

[nntpext] IETF Working Group, *NNTP Extensions* (laufend)
`https://datatracker.ietf.org/public/idindex.cgi?command=show_wg_id&id=1176`

[ASCII] Wikipedia, *ASCII*
`http://en.wikipedia.org/wiki/Ascii`

[A News] Wikipedia, *A News*
`http://en.wikipedia.org/wiki/A_News`

[B News] Wikipedia, *B News*
`http://en.wikipedia.org/wiki/B_News`

[Breidbart Index] Wikipedia, *Breidbart Index*
`http://en.wikipedia.org/wiki/Breidbart_Index`

[C News] Wikipedia, *C News*
`http://en.wikipedia.org/wiki/C_News`

[Cleanfeed] Wikipedia, *Cleanfeed*
`http://en.wikipedia.org/wiki/Cleanfeed`

[Great Renaming] Wikipedia, *The Great Renaming*
`http://en.wikipedia.org/wiki/The_great_renaming`

[Internet Troll] Wikipedia, *Internet Troll*
`http://en.wikipedia.org/wiki/Internet_troll`

[ISO 8859] Wikipedia, *ISO 8859*
`http://en.wikipedia.org/wiki/Iso-8859`

[Levenshtein] Wikipedia, *Levenshtein distance*
`http://en.wikipedia.org/wiki/Levenshtein_Distance`

[Top-posting] Wikipedia, *Top-posting*
`http://en.wikipedia.org/wiki/Top-posting`

[UDP] Wikipedia, *Usenet Death Penalty*
`http://en.wikipedia.org/wiki/Usenet_Death_Penalty`

[Unix time] Wikipedia, *Unix time*
`http://en.wikipedia.org/wiki/Unix_time`

[Usenet] Wikipedia, *Usenet*
`http://en.wikipedia.org/wiki/Usenet`

[Unicode] Wikipedia, *Unicode*
`http://en.wikipedia.org/wiki/Unicode`

[X-No-Archive] Wikipedia, *X-No-Archive*
`http://en.wikipedia.org/wiki/X-No-Archive`

[Zeilenende] Wikipedia, *End-of-line*
`http://en.wikipedia.org/wiki/End-of-line`