EXPLOITING LINKS AND TEXT STRUCTURE ON THE WEB
A QUANTITATIVE APPROACH TO IMPROVING SEARCH QUALITY

Von der Fakultät für Elektrotechnik und Informatik
der Gottfried Wilhelm Leibniz Universität Hannover
zur Erlangung des Grades eines

DOKTORS DER NATURWISSENSCHAFTEN
Dr. rer. nat.

genehmigte Dissertation von
Dipl.-Inf. (FH) Christian Kohlschütter
geboren am 26. Oktober 1979 in Hof/Bayern

2011

Referent: Prof. Dr. techn. Wolfgang Nejdl
Korreferent: Prof. Dr.-Ing. Bernardo Wagner
Tag der Promotion: 21.01.2011

Abstract

As Web search is becoming a routine activity in our daily lives, users scale up their expectations concerning Search Quality. This comprises factors such as accuracy, coverage and usability of the overall system. In this thesis, I describe quantitative strategies to improving search quality from two complementary perspectives, link structure and text structure, which are key topics in the field of Web Information Retrieval. I utilize some fundamental properties of the Web, presumably of human behavior after all, that are theoretically justified as well as relatively easy to apply.

Link Structure. Humans do not create or follow links to Web pages arbitrarily. In fact, most of the links refer to own pages (at host-level), a fact that I exploit for simplifying the PageRank computation, particularly the principal Eigenvector of the corresponding link matrix. Also, apparently humans seem to likely link to pages that are relevant to the originating document. I present a corresponding method for automatically identifying a topic of a text query solely based on link structure, utilizing multiple topic-specific PageRank vectors.

Text Structure. Humans also do not create or read text on Web pages arbitrarily. I show that the creation process of Web text is governed by a statistical law that corroborates the Quantitative Linguistic theory, yet I extend current models by the following notions: text on Web pages can be separated into blocks of "short text" and blocks of "long text", depending on the number of words contained in the block. A large amount of actual "full text" is attributed to the class of long text whereas short text appears to mainly cover the navigational text fragments usually referred to as "boilerplate". I present a simple, yet very effective strategy that utilizes this property for accurate main content extraction, ranking and classification.

As an attempt to unification, I conclude that the processes of browsing HTML pages and of creating HTML text can be seen as a combination of two orthogonal motivations. This perspective not only facilitates highly efficient and effective algorithms, it also aids in understanding the corresponding human behavior.

Zusammenfassung

Mit der wachsenden Bedeutung des World Wide Web im täglichen Leben steigt auch die Erwartungshaltung gegenüber Suchmaschinen und deren Qualität. Dies umfasst Aspekte wie z.B. Treffergenauigkeit, Abdeckung und Nutzbarkeit (Usability) des Gesamtsystems. In der vorliegenden Dissertation beschreibe ich quantitative Strategien zur Verbesserung der Suchqualität aus zwei sich ergänzenden Perspektiven, Linkstruktur und Textstruktur, zwei Kernthemen im Bereich des Web Information Retrieval. Hierbei betrachte und nutze ich einige fundamentale Eigenschaften des Web (und vermutlich des menschlichen Verhaltens im Allgemeinen), welche theoretisch fundiert und zugleich relativ einfach anwendbar sind.

Linkstruktur. Menschen setzen und folgen Hyperlinks auf Webseiten nicht willkürlich. In der Tat ist es so, dass ein Großteil auf eigene Seiten zeigt (auf Host-Ebene). Diese Eigenschaft nutze ich für eine Vereinfachung der PageRank-Berechnung, bei der der Haupteigenvektor der dazugehörigen Link-Matrix gesucht wird. Es hat sich gezeigt, dass Links häufig dann gesetzt werden, wenn die verbundenen Seiten thematisch zusammen hängen. Diese Eigenschaft nutze ich, um, nur mittels Linkstruktur und themenspezifischen PageRank-Vektoren, zu einer Freitext-Suchanfrage automatisch relevante Themen zu finden.

Textstruktur. Menschen setzen und lesen auch Text auf Webseiten nicht willkürlich. Ich zeige, dass der Erzeugungsprozess von Text im Web beschrieben werden kann durch ein statistisches Textgesetz, welches im Einklang mit Erkenntnissen aus der quantitativ-linguistischen Texttheorie steht. Hierbei erweitere ich jedoch bestehende Modelle wiefolgt: Text im Web besteht aus zweierlei Arten von Blöcken, jene mit kurzem Text und solche mit langem Text, abhängig von der Anzahl der eingeschlossenen Wörter. Ein Großteil des eigentlichen Haupttext einer Webseite kann mit *Langtext* beschrieben werden, wohingegen *Kurztext* hauptsächlich die navigationsspezifischen Textfragmente, den sogenannten “Boilerplate”, beschreibt. Diese textuelle Gesetzmäßigkeit mache ich mit Hilfe einer einfachen aber effektive Strategie zum akkuraten Extrahieren von Text, zum Ranking und zur Klassifikation von Webseiten nutzbar.

Als Versuch einer Vereinheitlichung schlussfolgere ich, dass die Prozesse der Erzeugung bzw. Rezeption von HTML Links und Text als Kombination zweier orthogonaler Motivationen beschrieben werden können. Diese Perspektive erlaubt nicht nur hocheffektive Algorithmen, sie ermöglicht auch ein besseres Verständnis menschlichen Verhaltens.

Keywords

Search Engines, Link and Text Structure, Quantitative Models

Schlagworte

Suchmaschinen, Link- und Textstruktur, Quantitative Modelle

Acknowledgments

I am grateful to have had numerous inspiring discussions with many people who shared their insights with me on a variety of topics, which eventually led to the present thesis. I would like to thank them all.

First and foremost, I would like to express my deepest gratitude to my supervisor Professor Dr. Wolfgang Nejdl for giving me the opportunity to conduct my thesis research, for his advice, guidance and profound support throughout this work.

I am very thankful to also have Professors Dr.-Ing. Bernardo Wagner and Dr.-Ing. Markus Fidler in the thesis committee, spending their time on my dissertation.

I also owe Professors Dr. Gabriel Altmann and Dr. Reinhard Köhler a special debt of gratitude for providing a plethora of excellent work in the field of Quantitative Linguistics, and for helping me, by their publications as well as by private correspondence, to deeper understand the problem domain from a complementary perspective. Their unparalleled help to introduce me to the community of Quantitative Linguistics deserves my deepest respect.

Special thanks go to Dr. Peter Fankhauser for dragging my attention to Machine Learning, for deep and insightful comments and great discussions.

The members, colleagues and former colleagues of the L3S Research Center deserve many thanks for providing a stimulating and fun environment, for fruitful discussions and for interesting collaborations at research and project work.

My research was finally made possible by having a full-time position at L3S as a research associate, which was mainly funded by the European Commission's FP6/FP7 projects NEPOMUK, ELEONET and SYNC3 and so, indirectly, by the taxpayers.

Many thanks also go to the German National Merit Foundation (Studienstiftung des deutschen Volkes) for their conceptual support during my studies.

Finally, I would like to thank my parents for their support and encouragement and for giving me the opportunity, volition and stimulus to seek challenges in the academia. Most importantly, I thank my wife, Anastasiya, for her love, exceptional patience and support as well as for her continuous belief in me and my work.

Christian Kohlschütter

Publication List

The algorithms and experimental results presented in this thesis have been published in several conference proceedings in the field of Information Systems and as a book chapter in the field of Quantitative Linguistics, as follows.

1. Christian Kohlschütter, Paul-Alexandru Chirita, Wolfgang Nejdl. Efficient Parallel Computation of PageRank. In: Advances in Information Retrieval, 28th European Conference on IR Research, ECIR 2006, London, UK, April 10-12, 2006, pp. 241-252, 2006, Springer, 3-540-33347-9.

2. Christian Kohlschütter, Paul-Alexandru Chirita, Wolfgang Nejdl. Using Link Analysis to Identify Aspects in Faceted Web Search SIGIR 2006 Workshop on Faceted Search, Aug 10, 2006, Seattle, WA, USA.

3. Christian Kohlschütter, Paul-Alexandru Chirita, Wolfgang Nejdl. Utility Analysis for Topically Biased PageRank. In: Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007, pp. 1211-1212, 2007, ACM, 978-1-59593-654-7.

4. Christian Kohlschütter, Wolfgang Nejdl. A Densitometric Approach to Web Page Segmentation. CIKM 2008: Proceedings of the 17th ACM Conference on Information and Knowledge Management, Napa Valley, California, USA

5. Christian Kohlschütter. A Densitometric Analysis of Web Template Content. 18th International World Wide Web Conference (WWW2009), Madrid, Spain

6. Christian Kohlschütter. A Densitometric Classification of Web Template Content. In: Emmerich Kelih, Viktor Levickij, Gabriel Altmann (Editors), Methods

of Text Analysis: Omnibus volume. – Chernivtsi: CNU, 2009. pp. 133-155. ISBN 978-966-423-043-5

7. Christian Kohlschütter, Peter Fankhauser, Wolfgang Nejdl. Boilerplate Detection using Shallow Text Features. WSDM 2010: Third ACM International Conference on Web Search and Data Mining New York City, NY USA. (Nominated for the Best Paper Award)

In the course of developing the Ph.D. thesis, I have published several other research papers, which helped me understanding the broader scope of Information Retrieval, while shaping the focus of the present work. They may be regarded as related work:

8. Paul-Alexandru Chirita, Christian Kohlschütter, Wolfgang Nejdl, Raluca Paiu. Using ODP Metadata to Personalize Search. In: SIGIR 2005: Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Salvador, Brazil, August 15-19, 2005, pp. 178-185, 2005, ACM, 1-59593-034-5.

9. Tereza Iofciu, Christian Kohlschütter, Raluca Paiu, Wolfgang Nejdl. Keywords and RDF Fragments: Integrating Metadata and Full-Text Search in Beagle++. Workshop on The Semantic Desktop - Next Generation Personal Information Management and Collaboration Infrastructure at the International Semantic Web Conference 6 November 2005, Galway, Ireland

10. Sergey Chernov, Christian Kohlschütter, Wolfgang Nejdl. A Plugin Architecture Enabling Federated Search for Digital Libraries. In: Digital Libraries: Achievements, Challenges and Opportunities, 9th International Conference on Asian Digital Libraries, ICADL 2006, Kyoto, Japan, November 27-30, 2006, Proceedings, pp. 202-211, 2006, Springer, 3-540-49375-1.

11. Sergey Chernov, Bernd Fehling, Christian Kohlschütter, Wolfgang Nejdl, Dirk Pieper, Friedrich Summann. Enabling Federated Search with Heterogeneous Search Engines: Combining FAST Data Search and Lucene. vascoda Federated Search Project Report. March 2006.

12. Sergey Chernov, Christian Kohlschütter, Wolfgang Nejdl. Exploring a European Market of Learning Objects with ELEONET. Proceedings of 2nd European Conference on Technology Enhanced Learning (EC-TEL 2007), Crete, Greece, September 17-20, 2007.

13. Christian Kohlschütter, Maria Nejdl, Dierk Höppner. Enhanced Federated Search for Digital Object Repositories. 2nd European Workshop on the Use of Digital Object Repository Systems in Digital Libraries (DORSDL2) in conjunction with ECDL 2008.

14. Christian Kohlschütter, Fabian Abel, Dimitrios Skoutas. A Novel Interface for Exploring, Annotating and Organizing News and Blogs Articles. The 3rd Annual Workshop on Search in Social Media (SSM 2010), co-located with the ACM WSDM 2010 Conference on Web Search and Data Mining, 2010.

15. Christian Kohlschütter. Exploring the New York Times Corpus with NewsClub. HCIR 2010, Fourth Workshop on Human-Computer Interaction and Information Retrieval in conjuction with IIiX 2010, Aug 2010, New Brunswick, NJ, USA.

But come – be quick – search we the bag, and learn
What stores of gold and silver it contains.

Homer, Odyssey

Contents

List of Tables

List of Figures

List of Abbreviations

AuC	Area under Curve
BF	Block Fusion
BTE	Body Text Extraction
CSS	Cascading Style Sheets
DOM	Document Object Model
HFC	Hierarchical Faceted Categories
HTML	Hypertext Markup Language
I/O	Input/Output
ID	Identifier
IR	Information Retrieval
LAN	Local Area Network
MAP	Mean Average Precision
NDCG	Normalized Discounted Cumulative Gain
NMI	Normalized Mutual Information
ODP	Open Directory Project
P2P	Peer-to-Peer
RAM	Random Access Memory
RMSE	Root Mean Square Error
ROC	Receiver Operating Characteristic
TDQ	Text-Density Quotient
TLD	Top-level Domain
TREC	Text REtrieval Conference
URL	Uniform Resource Locator

Chapter 1

Introduction

Within less than 20 years the Hypertext project "World Wide Web" has turned from a research project at CERN into a word-wide information and communication platform for almost 2 billion people[1] (or 25% of the world's population), consisting of an almost innumerable amount of pages in the order of trillion unique URLs.[2] The Web has become the number-1 user application on the Internet, and is about to replace the traditional PC Desktop paradigm by providing an entrance to basically everything through the Web browser [119].

The development of the Web into an omnifarious, omnipresent, and eventually omniscient system for organizing the world's data allows for a plethora of new opportunities from a social, political and economical perspective. However: in order to fully exploit the capabilities of the Web, we must understand its fundamental properties and establish corresponding hypotheses, models and theories. At last, we are not only striving for understanding the data but also the one who makes and consumes it, that is, the human being.

Given the vast extent of the Web, in terms of size as well as diversity, we may approach this task from two opposite directions:

1. *Bottom-up.* Find and select particularly interesting scenarios on the Web, describe them and integrate them into the big picture.

[1] `http://www.internetworldstats.com/stats.htm`

[2] `http://googleblog.blogspot.com/2008/07/we-knew-web-was-big.html`

2. *Top-down.* Perform statistical analyses of particular quantitative properties of the Web at corpus scale and create well-fitting stochastic models.

This exploration is an important aspect in the field of *Web Science* [59]. While the former approach may help us to characterize and understand emerging trends and at describing particular singularities, the latter empowers us to identify the invariants of the Web at large. The focus of my thesis will be on the latter direction.

1.1 Web Search

A key Web activity that strongly depends on the stability and the invariants of the Web, and at the same time is required to find and highlight hot topics, emerging trends and hidden gems, is Web Search. Search engines allow users to access information on the World Wide Web without exact knowledge of its location.

Traditionally, search engines turn keyword queries into ranked lists of matching hypertext (HTML) documents that are accessible through hyperlinks, sorted by relevance to the query and supposed importance to the user. Ultimately of course, we may regard search engines as portals to any kind of information that is accessible on the Web, not only at document level and not only using keyword search.

Obviously a key challenge is to populate the top-ranked slots solely with documents that – for every query – match the user's search intent as closely as possible. Regardless of how well a particular ranking function works, the quality of the induced ranking of query results directly depends on the quality of the data that can be retrieved. As opposed to knowledge bases with a defined purpose, schema, traceable provenience information, history recording, consistency checking and a defined target audience, the Web mostly is unstructured, inconsistent and full of ambiguity, bias, duplicate and false information. It is therefore essential to discover explicit and latent document properties such as origin, context, popularity, recency, length, writing style, etc. and use this information as further signals for ranking in addition to the documents' actual text.

1.2 Problem Statement

A big challenge for improving the overall quality of search results is to find methods and generic, shallow, quantitative features that can be utilized at Web-scale, i.e., that are independent of a particular language, user community or document subset and that are relatively easy to compute. We may classify these shallow features into two broader, complementary categories. The ones, which are dependent on structures at text level and those, which are dependent on structures at hyper-text level, that is, the link structure.

Both categories exhibit a multitude of problems and possible solutions. In this thesis, I will focus on the following question: how can we stochastically model human behavior when compositing HTML pages on the Web, and how can we utilize this for Web search? More specifically, the following questions arise:

Problem 1 *How can we – at Web scale – model and utilize the inherent link structure of a Web page (its outgoing and incoming links) for improving search quality?*

Problem 2 *How can we – at Web scale – model and utilize the inherent textual structure of a Web page for improving search quality?*

1.3 Proposed Solutions

1.3.1 Link Structure

An established strategy to estimating the importance of a web page, regardless of its content, is the PageRank algorithm. It essentially models the average user's web surfing behavior by an interpolation between a *random walk with restart* in the Web graph (the links between individual pages on the Web) and a static, adjustable visiting probability for each individual page. As these visiting probabilities can be arranged differently for individual users, communities and situations, PageRank is a good candidate for conveying personalized and topic-specific rankings in Web search and thus promises better top-ranked search results.

Efficient Parallel Computation of PageRank [77]. Unfortunately, the computation of PageRank is a time- and memory-extensive process as one needs to iteratively compute a score for each individual page in the Web graph (billions of pages), based upon the the scores of all other pages that link to this particular page (zero to millions of pages). I present an improvement to the PageRank algorithm to speed-up, parallelize and distribute the PageRank computation. My reformulation exploits the property of the Web's link structure of host-based link locality, which allows for an optimized computation for all pages referring from the same host (over 90% of all links in the corpus), yielding a dramatic reduction of communication overhead (to only 0.6%) over traditional parallelized and non-parallelized solutions.

Utility Analysis for Topically Biased PageRank [78]. As a direct consequence of the accelerated computation of PageRank (by orders of magnitudes) we can now explore the actual capabilities of topic-specific PageRanks on a much broader scale. For instance, it is crucial to know to which granularity such a biased computation makes sense. I present the results of a thorough quantitative analysis of biasing PageRank on categories of the Open Directory Project (ODP) web catalog. I show that the quality of Biased PageRank generally increases with the nesting level up to a certain point, thus sustaining the usage of more specialized categories to bias on, in order to improve topic-specific search. Particularly deeper categories apparently do not yield a significantly different ranking compared to their ancestor categories, which indicates a real-world upper bound to the problem of topic-specific PageRank.

Using Link Analysis to Identify Aspects in Faceted Web Search [76]. Now being able to compute individual topic-specific scores for page importance, two questions come up naturally: Can we use these individual scores to identify the topic-specificity of an individual web page? If so, can we infer topics inherent in a search *query* by looking at the corresponding search results? I affirm both questions by the demonstration and evaluation of a corresponding system utilizing ODP categories to compute topically-biased PageRank vectors for all pages of a 9 million pages Web crawl and a validation test collection from Google AdWords. In 91.6% of the evaluated test cases the system was able to infer the topic of a query – without understanding or even looking at the textual contents of the retrieved pages.

1.3.2 Text Structure

An established strategy to retrieving textual pages that are relevant to a particular keyword query is to treat each document as a bag of words. That is, regardless of the text's structure each occurrence of a query term is regarded a match. While this makes perfect sense in reviewed, high-quality knowledge bases, the approach is somewhat inappropriate for the Web. There is no restriction on how Web pages may be constructed or look like, they may be composites of several independent texts, they may contain a feedback section for user comments and they may particularly be contaminated by non-relevant text fragments, such as navigational elements, templates and advertisements, commonly referred to as *boilerplate* text [12]. Segmenting and cleansing the pages from the non-relevant text may generally improve the quality of search results and thus is an important aspect of our problem.

A Densitometric Approach to Web Page Segmentation [75]. Unfortunately, the automatic decomposition of a web page into individual segments is a non-trivial task, especially since there appears to be no generic solution that fits all web sites; the internal HTML and CSS structures are just too diverse across the Web as they may be defined individually by the web site owner/designer. One may consider a visual analysis of the content for the purpose of segmentation, based on the rendered graphical representation as in the Web browser − which still is computationally expensive. I present an approach that combines the benefits of the two approaches: while still working at HTML hypertext-level, I utilize metrics that estimate the *visual density* of a text area, similar to the approaches that would be applied by a vision-based strategy, by introducing the "text density" metric, which denotes the average number of words per line in an HTML text node (lines are constituted by word-wrapping the text at a fixed boundary). The approach reduces the segmentation problem to a 1D-partitioning task, where adjacent text elements with dissimilar text densities indicate a segmentation gap. I present an evaluation of my BlockFusion algorithm (that employs this text density metric) on a large reference Web corpus for the segmentation problem and a smaller reference collection for a near-duplicate detection task, in both cases yielding a significantly higher goodness (87%) than the state-of-the-art.

A Densitometric Classification of Web Template Content [74, 73]. The successful adaption of the density metric to the problem of web page segmentation indicates that there is a direct relationship between text density and the text class or style of a particular element. Using a statistical analysis on a large reference Web corpus, I show that at the level of text density there is a strong distinction between two classes of text, namely navigational boilerplate and full text. Using a simple Beta-Gaussian mixture model, we can describe the text density distribution of the corpus very accurately, achieving an almost perfect fit (99.8%); this structure corroborates recent findings from the field of Quantitative Linguistics. Moreover, I apply the model to the problem of boilerplate detection and also show that there is a high correlation between the frequency of a text block and its brevity (number of words).

Boilerplate Detection using Shallow Text Features [79]. Obviously, besides text density there may be other shallow text features that could be used to detect and separate boilerplate text from the real content. In the present thesis, I evaluate the applicability of overall 67 shallow features to the problem of boilerplate detection. I show that using a simple decision tree classifier with only two types of features (text density + link density, or number of words + link density, respectively), we can achieve a detection quality that is superior to state-of-the-art techniques. From this findings I derive a simple, plausible and well-fitting stochastic model for describing the boilerplate-aware text creation process (98.8%), based upon a simple Shannon Random Writer model. With the help of this model, we can finally quantify the impact of boilerplate removal to retrieval performance, seeing significant improvements over the baseline, at almost no cost.

Chapter 2

Foundations and Terminology

We start by introducing important notions and foundations of Web Search in general and in particular towards a definition of "quality" in this domain.

2.1 Search Quality

In simple words, the main purpose of performing a search on the Web is to find. What to expect to be found, what can be found, what to request, what is actually searched, what is actually found and what is perceived as found depends upon several interconnected factors between the searcher, the searchable items, the origin and/or producer of these items and the algorithms and tools that are being employed in the search process. Given the enormous size and diversity of the Web, and its increasing societal importance, we can eventually expect any kind of information to be available for search and retrieval.

As the realistic result of such a complex process can only be suboptimal, the goal is not to achieve optimality but to *approximate* it. Given the fuzzy definition of the search process, we should probably define "search quality" only as a subjective impression of the *goodness of fit* between the expected results (considering an imaginary optimum) and the observed results. We may then attempt to quantify the *improvement of search quality* by measuring and evaluating certain properties of the retrieved search results. Any statistically significant improvement of one or more particular properties can thus be regarded an improvement of search quality.

Generally, we can see the Web as a special kind of (read: universal) document collection that be used for purposes of information retrieval. Yet, as opposed to traditional, smaller, centrally-maintained data collections, the sheer size and diversity of the information on the Web as a whole not only poses challenges in terms of technical feasibility (complexity of acquisition, storage/maintenance of data as well as algorithmic computation) but it also enables us to analyze human behavior (in their roles as information seekers as well as information producers) on a very large scale, as opposed to finding peculiarities of the data in the smaller collections. If we are thus able to find a (maybe even incomplete) statistical formulation or stochastic approximation of human behavior that can be observed at macro scale, these properties may directly help us to improve the goodness of fit and thus, search quality.

In this thesis, I narrow the scope of what can be found to *hyper-text (HTML) documents* and what can be requested to *free text* – which commonly is being referred to as keyword search. This simplified problem domain still covers the most common actions in Web Search, for example through search engines like Google[1], Yahoo![2] or Bing[3] and enables us to employ standard text retrieval methods as a baseline. A wealth of further readings on the principles of Web Search and Information Retrieval in general can be found in [9, 87, 100, 31]. In the following sections, I will highlight some important aspects of Web document modeling for the purpose of retrieval (search) as well as metrics and strategies to evaluate result quality.

2.2 Web Page Modeling and Retrieval

Document. In our context we may refer the term *web page* to the raw HTML-coded data, to the parsed plain text, to the associated metadata such as URL, title and link structure (in- and outgoing hyperlinks), and to the rendered representation in

[1] `http://www.google.com/`

[2] `http://www.yahoo.com/`

[3] `http://www.bing.com/`

the browser as well (to render, the browser needs to additionally take referenced/embedded resources like CSS style sheets, images etc. into account). Even though the definition of a "document" in general is a subject of discussion [21], I will use the terms *document, web page* and *page* synonymously here. Apart from the individual documents, putting them into context (such as the neighborhood of pages that link to a particular document) or into a class (e.g., all pages from Germany) gives additional information that can be used for the retrieval of individual web pages, especially when searching for the top-k best matching results.

Index. It is generally advisable to use all possible dimensions of a document as features for search, but from a technical perspective it is desirable to maximize the utility value of the data while minimizing acquisition, computation and storage costs. What at least ends up being searchable is an index structure consisting of simplified representations of each web page's text (a tokenized, stemmed, truncated, weighted *bag of words* or *term vector* populating an inverted index, optionally with positional information for phrase queries), and a text-independent weight indicating relative page importance with respect to the other pages indexed.

Query. Analogously, the textual query issued by the user may be analyzed in various ways (to detect certain phrases, patterns, etc.), but in the simple case, the words are translated to terms in the same way as for the documents, again forming a *bag of words* (web search queries tend to be really short, though, likely less than 3 words [63]). Additionally, the user might be able to specify a *page bias* on particular web pages through other means than the keywords. The user may want to restrict the source to pages from Germany or put a focus on pages related to another page or cluster of pages, for example.

Retrieval. Matching the query with indexed information yields the search results. The matching mainly is based upon the proximity between the query's bag of terms and each document's bag of terms, optionally weighted by each term's utility in the corpus and the document (e.g., plain $TF \times IDF$ [98] or Okapi BM25 [99]). Proximity can be defined as the cosine between the term vectors (Vector Space Model [101]) or as a multi-dimensional Euclidean distance (Extended Boolean Model [102]), for example; the closer the more relevant a document is regarded with respect to a query.

To account for the popularity or authoritativeness of the source (regardless of the actual content) the scoring process may also take the contextual relevance of a web page as a scoring factor (w.r.t. the corpus or to the page bias, if specified). In top-k retrieval, the results are then being ranked in decreasing order of score, whereas only the k highest ranking results are being shown to the user. It is thus crucial to satisfy the user already through these k results.

Noise. By definition, this retrieval process may deliver too few results (if the query is incorrectly specified, over-specified or, in the unlikely case on the Web, that there are no matches for a particular query) as well as *irrelevant* results. The latter may surely be due to an individual ranking/scoring algorithm but is mainly due to *noise* in the searchable data, whereas we may refer *noise* to any kind of unwanted (= non-relevant) information, including spam.

Since a document that contains all possible terms would achieve top scores, which clearly is not desirable, it is not sufficient to only weight individual terms according to their overall semantic relevance but also to detect a document's main content (the central information) or at least detect any irrelevant elements and remove/downweight them. Similarly, a page should not necessarily be regarded relevant to a query just because of its popularity (in terms of incoming links), nor should all outgoing links be treated equally (links in a navigation menu fulfill a structurally different purpose than hyperlinks in full text, for example). Thus, an important challenge (also from a technical perspective) is to avoid and remove noise as early as possible and to reduce the impact of the remaining noise to the search task.

2.3 Quantitative Evaluation

When it comes to measuring and comparing the performance of a model or algorithm, numerous metrics may be employed. For our tasks, we are mainly interested in two types of tests: the goodness of a binary classification (how well can we separate correct/relevant from incorrect/irrelevant search results?) as well as the goodness of a statistical model to the real data (how well can we describe it?). While it is obviously simple to rephrase a stochastic model as mathematical formula, the creation of a "gold standard" for the purpose of evaluating classification goodness requires substantial human effort. Each individual item (e.g., a web page) needs to be judged relevant (binary, graded or labeled with different classes) with respect to the given query or context. To gain insights about the Web as a whole, the judgment process needs to be performed on a somewhat *representative* basis.

Even though a model or algorithm has to bear up against the properties of the entirety, this does not necessarily implicate "the bigger the better". An evaluation on just one thousand documents each from a different website may deliver more insight about the Web's properties than an evaluation on a million documents from a single source. The key is to preserve the inherent structures of the Web's entirety within the evaluated subset, especially diversity (to avoid overfitting) and the Web's Zipfian properties (word and link distributions, for example). Subsets can be created by manually constructing datasets (e.g., a web crawl using an authoritative seed or a collection of separate, special purpose datasets), or by random sampling of a very big, unbiased initial Web dataset (e.g. a large-scale web crawl). To achieve reproducible results, oftentimes (quasi-)standard collections are employed (e.g., in the TREC competitions [107], the Webspam challenge[4], CleanEval [12], and others), which usually have been compiled carefully. In this thesis, I use such collections whenever possible, feasible and useful.

Once a collection and the evaluation criteria on what is relevant and non-relevant are defined, we can count the number of relevant/irrelevant items (e.g., documents) that are retrievable in the collection and those that are contained in the set of items that are actually for a particular query and relate those numbers in various ways to understand the actual quality of the search results. Especially for top-k keyword search on the Web, it makes sense to only consider the subset of those top-k ranked items, as users tend to only scan through the first few results [97].

When regarding the retrieved and retrievable items as sets, we may separate four types of results: true positives (retrieved and relevant), true negatives (not retrieved and irrelevant), false positives (retrieved but irrelevant), false negatives (not retrieved but relevant), see Figure 2.1. Note that usually non-retrievable (but possibly relevant)

[4] `http://webspam.lip6.fr/`

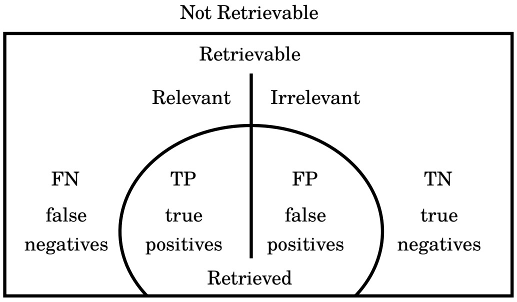


Figure 2.1: The different sets of the binary relevance classification

items are not considered at all. Items may be non-retrievable either because they are not in the collection or non-existent at all (= imaginary). Thus, in our context, *relevant* and *irrelevant* refer to retrievable items only.

Furthermore, since a ranked list of results can also be seen as a model or an estimator for the ideal response, we may also use some standard statistical methods for discrete random variables as an indicator of quality. As no universal measure appears to be in sight, I will use the following, complementing measures.

Precision, Recall and F-measure. Precision is the probability that a retrieved document is relevant; also: $P(n)$ = Precision at rank n (only the top-n results). Recall is the probability that a relevant document is retrieved. The F-measure F_1 is the harmonic mean between Precision and Recall.

$$P = \frac{|TP|}{|Retrieved|} \qquad R = \frac{|TP|}{|Relevant|} \qquad F_1 = 2 \cdot \frac{P \cdot R}{P + R}$$

MAP. Mean average precision. Given a set of queries, we compute the average precision AvP for each query from all $P(r)$ with r having a relevant result (i.e., $rel_r > 0$, i.e. 1 in the binary case), and then average again over all queries.

$$AvP = |Relevant|^{-1} \cdot [\sum_r^n P(r), \forall rel_r > 0] \qquad MAP = |Queries|^{-1} \cdot \sum_q AvP(q)$$

ROC. The *Receiver operating characteristic* relates recall (*true positive rate*, *sensitivity*) to fall-out (*false positive rate*, $1 -$ specificity, FP/Irrelevant). Fall-out is the probability that a non-relevant document is retrieved by the query. Recall and fall-out are computed for all top-p ranks, yielding a curve in ROC's fall-out/recall space. Ideally, we may get a straight line with 100% recall (and thus zero fall-out) – and a diagonal in the worst case (random guess) since any lower curves may be inverted (switching relevant/irrelevant), thus yielding higher scores again. ROC usually is summarized by the area under curve (AUC), which can, for example, be retrieved from the curve's Gini coefficient.

$$AUC = \frac{G_1 + 1}{2}$$

NDCG. Normalized discounted cumulative gain. DCG_p, the discounted cumulative gain at rank p, attributes greater importance to highly ranked items by logarithmically discounting the relevance of lower-ranked ones; it is then normalized by an hypothetical ideal (IDCG) to provide the NDCG. The ideal ranking comprises a list of monotonically decreasing relevance scores (in the case of a binary classification this means that no relevant document is preceded by an irrelevant one).

$$DCG_p = rel_1 + \sum_{i=2}^{p} \frac{rel_i}{log_2 i} \qquad\qquad NDCG_p = \frac{DCG_p}{IDCG_p}$$

Kendall Tau. Given a specific ranking of results, the Kendall τ coefficient measures the association to another ranking. As opposed to putting focus on the absolute position in the ranking, this metric evaluates the overall concordance of the two rankings (i.e., how often does an item i precede another item k in both rankings).

$$\tau = \frac{(\text{number of concordant pairs}) - (\text{number of discordant pairs})}{(\text{number of pairs})}$$

RMSE. The Root Mean Square Error is the square root of the variance between the response values Y and the predicted response values $\hat{Y}$.

$$RMSE(\hat{Y}) = \sqrt{E\left((\hat{Y} - Y)^2\right)}$$

R^2 **Coefficient of Determination.** Similarly to RMSE, R^2 describes the variation of the data that cannot be explained by the model. After a least squares regression, R^2 constitutes the square of the correlation coefficient between the response values Y and the predicted response values $\hat{Y}$. R^2_{adj} is an adjusted variant that punishes additional features/independent variables if they do not improve the model more than by chance (n = sample size, p = number of independent variables).

$$R^2 = \frac{\sum_i(\hat{Y}_i - \bar{Y})^2}{\sum_i(Y_i - \bar{Y})^2} = \frac{Cov(x,y)}{Var(x)Var(y)} \qquad R^2_{adj} = 1 - (1 - R^2)\frac{n-1}{n-p-1}$$

The **Kullback-Leibler (KL) divergence** is a measure for change in information entropy (the additional communication effort necessary) from a probability distribution P to Q. We can treat it as the Information Gain if P is used instead of Q.

$$D_{KL}(P\|Q) = H(P,Q) - H(P) = \sum_i P(i)\log\frac{P(i)}{Q(i)}$$

Mutual Information is the strength of dependence between two random variables X and Y. It is essentially the KL-divergence of the joint probability compared to the product of the individual probabilities, i.e., we expect a higher information gain if X and Y are closely related than if they were independent.

$$I(X;Y) = D_{KL}\left(p(x,y)\|p(x)p(y)\right)$$

Normalized Mutual Information (NMI) measures the mutual dependence of the two solutions by relating their entropies. Several variations of normalization exist. In this thesis I use Strehl's & Ghosh's variant [111], which is defined as

$$NMI(X,Y) = \frac{I(X;Y)}{\sqrt{H(X)\ H(Y)}}$$

.

Chapter 3

Link Structure

In this chapter, we deal with the first problem: How can we – at Web scale – model and utilize the inherent link structure of a Web page (its outgoing and incoming links) for improving search quality?

3.1 PageRank in short

Computing the importance of a web page purely from its structural importance at hyperlink-level (regardless of its content) is a highly active research field since the Web's early times. The most popular algorithm being PageRank [92], which recursively determines the importance of a web page by the importance of all the pages pointing to it. Regarding the link structure aspect of my thesis, I will focus on this particular ranking strategy.

Model: Random Walk with Restart. The main concept behind the PageRank paradigm is the propagation of importance from one Web page towards others, via its out-going (hyper-)links. Each page $p \in P$ (P is the set of all considered pages) has an associated rank score $r(p)$, forming the rank vector $\vec{r}$. Let L be the set of links, where $(s,\ t)$ is contained iff page s points to page t and $L(p)$ be the set of pages p points to (p's outgoing links). Links and pages finally form the web graph $G = (P, L)$. To compute $\vec{r}$, the following iteration step is then repeated until all scores r stabilize to a certain defined residual degree $\delta < \varepsilon$:

$$\forall t \in P : r^{(i)}(t) = (1-\alpha) \cdot \tau(t) + \alpha \sum_{(s,t)\in L} \frac{r^{(i-1)}(s)}{|L(s)|} \tag{3.1}$$

The formula consists of two portions, the jump component (left side of the summation) and the walk component (right side), weighted by α (usually around 0.85). $r^{(i-1)}(s) \cdot |L(s)|^{-1}$ is the uniformly distributed fraction of importance a page s can offer to one of its linked pages t for iteration i. Intuitively, this models a "random surfer" following an outgoing link from the current page (*random walk*) with probability α, which, with probability $1-\alpha$, gets bored and then restarts the process by opening a random page (*random jump*). The main utility of α is to guarantee convergence and avoid "rank sinks" [18].

In fact we may interpret this behavior as a Markov process associated to the web graph, having $\vec{r}$ as the state vector and A (see Eq. 3.2) the transition probability from one page to another. Thus we can write Equation 3.1 in matrix terms as follows:

$$\vec{r} = (1-\alpha) \ \cdot \ \vec{\tau} \ + \ \alpha \ A\vec{r} \tag{3.2}$$

Parallelization. Equation 3.1 represents the linear system representation of this matrix computation, using the Jacobi iterative method. Several improvements for a centralized computation of PageRank have been researched in detail [8, 27, 53, 65, 66, 82, 69, 89]. For example, other stationary iterative solvers may be used, such as the Gauß-Seidel method, which converges two times faster than Jacobi [8] but turned out to not be as efficiently parallelizable as the Jacobi method, since it requires access to the preliminary results of the current iteration (and thus, additional communication). There already are parallel Gauss-Seidel implementations for certain scenarios such as the one described in [70], using block-diagonally-bordered matrices; however, they all admit their approach was designed for a static matrix; after each modification, a specific preprocessing (sorting) step is required, which can take longer than the actual computation. Because the web is highly dynamic, almost 40% of all links change in less than one week [30], disregarding this preparation step would veil the real overall processing time. Steady reorganization of coordinates in a huge link matrix simply imposes an unjustified management overhead.

Existing approaches to PageRank parallelization can be divided into two classes: Exact Computations and Approximations. For *exact computations* of PageRank, the web graph is initially partitioned into blocks of individual pages: grouped randomly [103], lexicographically sorted by page [86, 106, 124] or balanced according to the number of links [49]. Then, standard iterative methods such as Jacobi (Equation 3.1) or Krylov subspace [49] are performed over these pieces in parallel. The partitions periodically must exchange information: Depending on the strategy this can expose suboptimal convergence speed because of the Jacobi method and result in heavy inter-partition I/O (e.g., in [86], computing the rank for a page t requires access to all associated source page ranks $r(s)$ across all partitions).

On the other hand, *approximations* might be sufficient to get a rank vector which is comparable, but not equal to the exact one. Instead of ranking pages, higher-level formations are used, such as the inter-connection/linkage between hosts, domains, server network addresses or directories, which can be orders of magnitudes faster. The inner structure of these formations (at page level) can then be computed in an independently parallel manner ("offline"), as in BlockRank [64], SiteRank [120], the U-Model [20], ServerRank [116] or HostRank/DirRank [37].

Personalized PageRank. The so-called *Personalized* PageRank[92] promises an improvement in ranking with respect to individual interests/facets, which are specified through a set of relevant pages. As opposed to the regular PageRank, the random jumps now only address a given set of pages to which the computation should be *biased.* A unified representation of both approaches (random/biased) is the following:

$$\forall t \in P \; : \; r_B(t) = (1-\alpha) \cdot \pi_B(t) + \alpha \sum_{(s,t)\in L} \frac{r_B(s)}{|L(s)|} \tag{3.3}$$

Here, $\pi_B(t)$ describes the likelihood that the surfer reaches page t from a jump. In the case of regular ("unbiased") PageRank, we treat all pages of the web graph as being contained in B, so: $\pi_P(t) = \tau(t) = |P|^{-1}$, in general it is:

$$\pi_B(t) = \begin{cases} \frac{1}{|B|} & \text{iff } t \in B \\ 0 & \text{otherwise} \end{cases} \tag{3.4}$$

The distribution of scores in the rank vector $\vec{r}_B$ follows a power law (just as the actual numbers of incoming and outgoing links to a page [19]) and thus is scale-free. Therefore multiplying a PageRank score with the keyword-based document relevance score $W(d, q)$ (e.g., based on the $TF \times IDF$ measure) causes important pages (with respect to the given biasing set B) to gain higher scores ρ_B. Note that the number of retrieved documents does not change when switching to another biasing set, since no facet-specific keywords are added. Instead the document ranking is changed:

$$\rho_B(d, q) = W(d, q) \ \cdot \ r_B(d) \tag{3.5}$$

Personalizing PageRank is in fact just a biasing towards an individualized set of scores (it is not anyhow "personal" by definition). As a non-user specific approach, Haveliwala's Topic-Sensitive PageRank[54] makes use of linear combinations of just 16 Personalized PageRank vectors, each biased on one of the 16 top categories of ODP respectively. The weights for the linear combination are derived from each topic's term vector (hence, access to full-text information is required):

$$\vec{r}_{B'} = \sum_{\beta} [\omega_\beta \vec{r}_\beta] \tag{3.6}$$

Other, more recent approaches to generate personalized rankings extend this scheme. Qiu and Cho [96] for example enhance Topic-Sensitive PageRank to contain user specific weights for combining the 16 biased vectors. They learn these weights through machine learning on the user's click history. Aktas et al. [38] have successfully applied Personalized PageRank with respect to domain name features (country and generic TLD topic assignment). Finally, Jeh and Widom [62] and more recently Sarlos et al. [104] have investigated the means to compute Personalized PageRank in a scalable way for a large set of users. They both exploited the idea of decomposing the biasing set into small sets with a single non-zero entry followed by a linear combination of the resulting Personalized PageRank vectors.

3.2 Parallelization and Distribution of PageRank

In this section I introduce a new approach to computing the exact PageRank vector in a parallelized fashion. Exact results are obtained faster than distributed strategies based on the Jacobi method, improving by orders of magnitude over the other algorithms generating exact PageRank scores. I show that the convergence improvements of the Gauß-Seidel method for solving linear systems [8] can also be efficiently applied in a parallelized PageRank scenario, without being restricted to static matrices as in [70]. We can achieve this by modeling the Web graph in a two-dimensional fashion (with the URL's hostname as the primary criterion), thus separating it into reasonably disjunct partitions, which are then used for distributed, incremental web crawling [30] and PageRank computation.

3.2.1 The Two-Dimensional Web

Computing the PageRank vector for a large web graph using a materialized in-memory matrix A is definitely not possible. A common solution is to store the links in a format like "Destination Page ID, Out-degree, Source Page IDs..." (which resembles L). Because pages only link to a few others this results in much lower memory requirements of the link structure, in the magnitude of $\mid L \mid \cdot \ \overline{n}^{\ -1} \cdot c$ bytes ($\overline{n} =$ average out-degree; $c = const.$)

Of course, compression techniques [80] or intelligent approaches to disk-based "swapping" [53, 27, 89] can improve the space requirements even further (e.g. by relying on a particular data order, or on the presence of caches). But with the permanent growth of the web, even such techniques will soon hit memory limits of a single computer, or unacceptably slow down the computation process. See [89] for a thorough discussion of these optimizations.

I thus propose a new, complementing strategy for keeping the web graph and rank information completely in RAM of several networked machines, utilizing a separation between global (host) and local information about each page.

Host-based Link Locality

Bharat et al. [17] have shown that there are two different types of web links dominating the web structure, “intra-site” links and “inter-site” ones. A “site” can be a domain (`.yahoo.com`), a host (`geocities.yahoo.com`) or a directory on a web server (`http://www.geocities.com/someuser/`). In general, we can define a site as an interlinked collection of pages identified by a common name (domain, host, directory etc.), and under the control of the same authority (an authority may of course own several sites).

Due to the web sites' hypertext-navigable nature, it is supposable that a site contains more internal than external links. In fact a high amount of all non-dangling links are *intra-host* or *intra-domain* ($> 93\%$) [64]. This assumed block structure has been visualized by Kamvar et al. [64] using dot-plots of small parts (domain-level) of the ”LargeWeb” graph's link matrix [55]. In these plots, the point (i, j) is black, if there is a link from page p_i to p_j, clear otherwise.

I performed such a plot under the same setting, but on whole-graph scale. The outcome is interesting: a clear top-level-domain (TLD) dominant structure (see Figure 3.1a). For example, the `.com` TLD represents almost 40% of the complete structure and has high connectivity with `.net` and `.org`, whereas the `.jp` domain shows almost no inter-linkage with other TLDs. However, if we only inspect the `.com` domain (see Figure 3.1b, the dot-plot depicts a diagonally dominant structure. The diagonal represents links from target pages nearby the source page (which are *inter-host* pages). Both results are primarily caused by the lexicographical order of URLs (with hostnames reversed, e.g. `http://`*`com.yahoo.www`*`/index.html`).

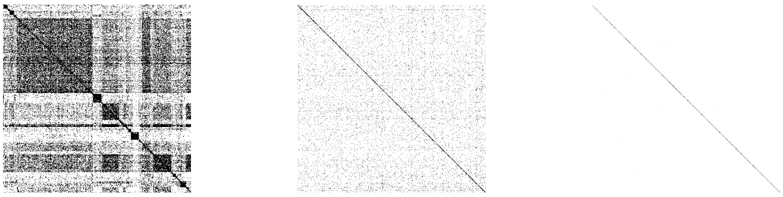
But is this costly *sorting* over all URLs necessary at all? To further analyze the impact of hostname-induced link locality, we can inspect the LargeWeb dot-plot in a *normalized* (histographical) fashion, where a dot's grayscale value depicts the *cumulative percentage* of links in a specific raster cell. In addition, we do not sort the pages lexicographically, but only group them per host and permute all hosts randomly to avoid any lexicographical or crawl-order-dependent relationship between them. The clear diagonal dominance now also becomes visible on whole-graph scale (Figure 3.1c).

From Numbers to Tuples

It should be obvious that the web was already designed to be two-dimensional: Hostnames are namespaces aimed to disambiguate different local contexts (i.e., paths like "`/dir/index.html`").

Previous approaches to web graph partitioning always resulted in having *one* unique ID associated to each page, eventually sorted lexicographically [64, 86, 106] or in crawling order to exploit specific graph properties [89]. Such a single page ID provides a very compact representation of the web graph, which can be visualized in a matrix dot-plot as shown above. But it also requires continuous reorganization (sorting) for newly added or removed pages in the course of incremental crawling. Otherwise, a mixture of hosts along the URL IDs would render a host no longer characterizable by a closed interval of IDs, thereby losing the advantage of link locality. One may introduce gaps in the numbering to reduce the sorting costs, but still, all subsequent pages will have to be renumbered once the gap is filled. In a distributed scenario, this can cause extensive network I/O by repeatedly moving pages from one partition to another.

I therefore propose a different page identification scheme, based on the affiliation of each page to a specific host and independently of pages from other hosts. More specifically, we may use a *tuple* consisting of two independent, positive integers, a HostID (only dependent on the URL's hostname) and a LocalID (identifying the local components – path and query string). This simplifies the addition of new local pages to a specific host, as well as of new hosts, since it avoids any renumbering.



(a) LargeWeb, sorted (b) .com subgraph of (a) (c) LargeWeb, normalized

Figure 3.1: Linkage dot-plots

As an implementation-specific note, it is expected that for current web graphs, it is sufficient to store the tuples as two `uint32` four-byte integers. We then can address a maximum of 4.29 billion hosts and a maximum of 4.29 billion pages per host in 8 bytes. For small hosts, we could even reduce the local part to 16 bits, thereby further cutting down memory footprint.

3.2.2 Partitioned PageRank

Let us now consider the impact of such a partitioning scheme on the PageRank algorithm. I first present an analysis that unifies two of the most common algorithms for solving linear systems, Gauß-Seidel and Jacobi. Then, we will apply this analysis to propose an improved parallel PageRank algorithm, and finally I will discuss several optimization issues.

Unifying Jacobi and Gauss-Seidel

It has been observed that the Gauß-Seidel iteration method compared to the Jacobi method can speed-up PageRank convergence by a factor of 2, as it uses scores of the current iteration as soon as they become available [8]:

$$\forall (s,t) \in L : \quad r^{(i)}(t) = (1-\alpha)\ \tau(t)\ +\ \alpha \left(\sum_{s<t} \frac{r^{(i)}(s)}{|L(s)|} + \sum_{s>t} \frac{r^{(i-1)}(s)}{|L(s)|} \right) \tag{3.7}$$

As opposed to the Jacobi iteration, the Gauß-Seidel variant requires iterating over the links $(s,t) \in L$ in a strictly ascending order. At first glance, this seems to be a major drawback when we want to apply it to a distributed, partitioned web graph.

To clarify the impact of the restriction of link order, we now derive a common base algorithm for both, Jacobi (equation 3.1) and Gauß-Seidel (equation 3.7) algorithms: Let us define an intermediate ranking vector $r^{(i-1,i)}$ that combines the vectors of the previous and the current iteration, depending on the state of a ranked page p in the set of available pages P ($P = P' \cup P''$; $\nexists\, p : p \in P' \wedge p \in P''$; P' contains all pages which have already been ranked for iteration i; P'' contains all other pages, whose scores have not been touched since iteration $i-1$):

$$r^{(i-1,i)}(p) := \begin{cases} r^{(i)}(p) & \text{if } \; p \in P' \\ r^{(i-1)}(p) & \text{if } \; p \in P'' \end{cases} ; \; r^{(i)}(t) = (1-\alpha)\ \tau(t) + \alpha \sum_{(s,t)\in L} \frac{r^{(i-1,i)}(s)}{|L(s)|} \quad (3.8)$$

Under this setting, for the Gauß-Seidel method, $P' = \{\ p \mid p < k\ \}$ and $P'' = \{\ p \mid p \geq k\ \}$, with $k \in \{1, 2, ..., |P|\}$, whereas for the Jacobi method, we have $P' = \emptyset$ and $P'' = P$. Both iteration methods, Jacobi and Gauß-Seidel, can then be simplified to this joint formula:

$$r^{(\star)}(t) = (1-\alpha)\ \tau(t) + \alpha \sum_{(s,t)\in L} \frac{r^{(\star)}(s)}{|L(s)|}, \text{ with } r^{(\star)}(t) = r^{(i-1,i)}(t) \quad (3.9)$$

From Equation 3.8, we know that before each iteration i, $\vec{r}^{(\star)} = \vec{r}^{(i-1)}$ and after the iteration $\vec{r}^{(\star)} = \vec{r}^{(i)}$. The state of $\vec{r}^{(\star)}$ during the iteration then only depends on the order of links $(s,t) \in L$ (the way how P' and P'' are determined). This iteration method has worst-case convergence properties of Jacobi and best-case of Gauß-Seidel, depending on the order of elements, random order vs. strictly ascending order, while always providing the same per-iteration running time as the Jacobi iteration.

We further generalize the impact of the rules for P' and P'': We can argue that if only a small fraction F of all links concerned ($|F| \ll |\ L\ |$) is not in strictly ascending order, the overall convergence speed still remains in the magnitude of standard Gauß-Seidel. In our case, in order to be able to parallelize the Gauß-Seidel algorithm, we will assign inter-host/inter-partition links (about 6%) to this small fraction.

Reformulating PageRank

For such an optimization, let us reformulate our above mentioned unified PageRank equation using our new two-dimensional page numbering scheme. Thus, page variables "p" will be replaced by page tuples $\mathsf{p} = (p_x, p_y)$, with p_x representing the page's

HostID, $host(\mathsf{p})$, and p_y its LocalID, $local(\mathsf{p})$. To account for the separation of inter- and intra-host links, the formula now reads as follows:

$$\begin{aligned}
r^{(\star)}(\mathsf{t}) &= (1-\alpha)\ \tau(\mathsf{t})\ +\alpha\ \left(v_I^{(\star)}(\mathsf{t}) + v_E^{(\star)}(\mathsf{t})\right) \\
v_I^{(\star)}(\mathsf{t}) &= \sum_{(\mathsf{s},\mathsf{t})\in L} \frac{r^{(\star)}(\mathsf{s})}{|L(\mathsf{s})|} \quad \forall\ host(\mathsf{s}) = host(\mathsf{t}) \\
v_E^{(\star)}(\mathsf{t}) &= \sum_{(\mathsf{s},\mathsf{t})\in L} \frac{r^{(\star)}(\mathsf{s})}{|L(\mathsf{s})|} \quad \forall\ host(\mathsf{s}) \neq host(\mathsf{t})
\end{aligned} \tag{3.10}$$

Since $v_I^{(\star)}(\mathsf{t})$ solely requires access to *local* (intra-host) rank portions, it can efficiently be computed from scores stored in RAM. The local problem of ranking intra-host pages is solvable via a fast, non-parallel Gauß-Seidel iteration process. There is no need for intra-host vote parallelization − instead, we parallelize on the host-level, thus necessitating only inter-host communication, which is limited to the exchange of external votes.

This approach produces exactly the same ranks as the original PageRank, while being more scalable than the other parallel PageRank algorithms. This is mainly due to the parallelization of the Gauß-Seidel algorithm, in which we take advantage of the web's host-oriented block structure.

Reaching Optimal Performance

Communication Cost Optimization. While votes between hosts of the same partition (server) can easily be conveyed in RAM, votes across hosts of different partitions require network communication. The gross total for exchanging external votes over the network must not be underestimated. With the LargeWeb graph setup, almost 33 million are exchanged between partitions. For bigger web graphs, this could rise up to a few billion exchanges and can easily lead to network congestion if too much information is transmitted per vote.

As opposed to other approaches, where a vote consisted of target page ID (sometimes along with source page ID) and score, we simply reduce this to transmitting a single value per page (the score), because the link *structure* does not change during the iteration cycle. More generally, the link structure of all the pages that exchange votes between two partitions pages only needs to be determined whenever the graph changes (in the case of incremental web crawling) and then to be sent to the specific target partition. Moreover, the source page does not need to be specified in order to compute the PageRank score, but only the target page ID (see Equation 3.10). Additionally, by grouping the list of target pages by host, we need to transmit each target host ID only once.

Most notably, each partition has to transmit only one single value per target page, not per link to that page, since all votes from local pages that link to a specific page can be aggregated to a single value (surprisingly, this simple but very effective approach did not appear in any previous work):

$$v_E^{(\star)}(\mathsf{t}) = \sum_{\beta \in \Pi} \sum_{(\mathsf{s},\mathsf{t}) \in L_\beta} \frac{r^{(\star)}(\mathsf{s})}{|L|} = \sum_{\beta \in \Pi} v_\beta^{(\star)}(\mathsf{t}) \ \ \forall \ host(s) \neq host(t) \tag{3.11}$$

with Π being the set of partitions containing links towards t, and β each one of these partitions.

Transferring $v_\beta(\mathsf{t})$ (the sum of votes from partition L_β to t) as a single value reduces the network load dramatically. Using this optimization, we see a reduction of vote exchanges by 89% with the LargeWeb graph. Table 3.1 lists the differences between inter-partition links and votes and their quota of all links.

Type	**Amount**	**Percent**
Total Links	601,183,777	100%
Inter-Partition Links	32,716,628	5.44%
Inter-Partition Votes	3,618,335	0.6%

Table 3.1: LargeWeb Inter-Partition links and votes

Computational Load Balancing. In order to keep the convergence behavior of the centralized PageRank in our parallel scenario, inter-partition votes must be exchanged after every iteration (see [86] for a discussion of consequences of not doing so). To keep the overall computation time still low, all intra-partition computations and after that all network communication should terminate isochronously (at the same time). Because intra-partition computation is directly proportional to the number of pages per partition (see Equation 3.10), this either means that all available servers must be equally fast, or the graph has to be at least partitioned adequately to the performance of the servers. Moreover, other slow-down factors could also influence the running time, such as different network throughput rates of the network controllers and system boards (even with the same nominal speed).

A good strategy to load-balancing Parallel PageRank in a heterogeneous environment could be running a small test graph on all new servers, measure computation speeds, and balance the real graph accordingly. In any case, memory overflows due to bad balancing parameters like in [49] are avoided, and no manual interaction to find these parameters is necessary.

3.2.3 Experiments

For the PageRank experiments, I first converted the Stanford LargeWeb graph [55] into the new tuple representation, resulting in 62.8M pages and 601M links distributed over 470,000 hosts with averaged 137.5 pages each (the maximum was 5084 pages per host); the inter-host link percentage[1] is 6.19% (see Table 3.2). I then sorted the available hosts by their page count in descending order and distributed the pages host-wise in a round-robin manner over 8 partitions of equal size ($\frac{1}{8}$ of the graph just fitted into the RAM of the smallest server).

Although the pages-per-host distribution was not strictly exponential, it resulted in an equal page and link distribution (see Figures 3.2, 3.3, 3.4, 3.5). Remarkably, the intra-partition ratio (inter-host links inside the same partition) is negligible, as the inter-partition link rate nearly equals to the inter-host ratio. This means that hosts can arbitrarily be shifted from one partition to another (which is necessary for fast re-balancing with incremental web crawling).

[1] Unfortunately, the last 8 million pages of DNR-LargeWeb could not be converted, since there was no URL associated with them – thus, our numbers slightly differ from the ones in [64].

Type	Amount	Percent
Total	601,183,777	100%
Intra-Host	563,992,416	93.81%
Inter-Host	37,191,361	6.19%
Inter-Partition	32,716,628	5.44%
Intra-Partition	4,474,733	0.74%

Table 3.2: LargeWeb link distribution

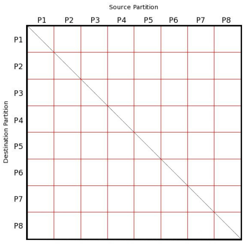


Figure 3.2: Partitioned Dotplot (LargeWeb)

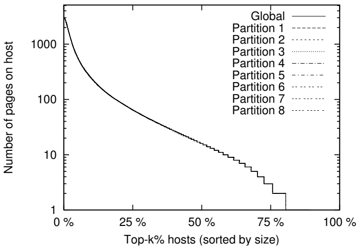


Figure 3.3: Pages per host (8 individual partitions and global)

Implementation

I have implemented Partitioned Parallel PageRank in Java using a P2P-like network with a central coordinator instance. This coordinator is only responsible for arranging the iteration process at partition-level and does not know anything about the rank scores or the link structure (it is much simpler than the coordinator in [124]). Before the computation, all nodes announce themselves to the coordinator, communicating the hosts they cover. The iteration process is started as soon as all nodes are ready. The coordinator then broadcasts the global host structure to all known nodes and instructs them to iterate. Whenever a node's subgraph changes, it sends lists of external outgoing link targets to the corresponding nodes.

For every iteration step, a node will compute its votes using our reformulated PageRank (Equation 3.10); the partition itself is again divided into subpartitions processed in parallel. The nodes then aggregate all outgoing inter-partition votes by target page and send them directly to the other nodes responsible for these target

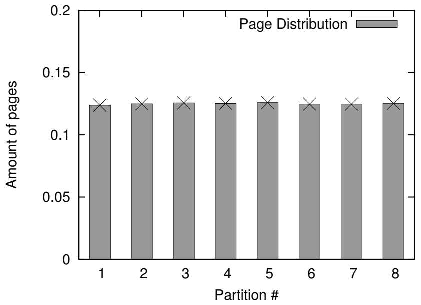


Figure 3.4: Pages per Partition

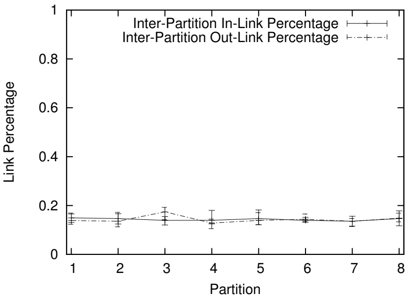


Figure 3.5: Inter-partition links

pages, in the order specified beforehand. Finally, each node reports its local rank status (using the sum and number of its PageRank scores) to the coordinator, in order to compute the global residual δ. As soon as all nodes have succeeded, the coordinator decides whether to continue iterating, by broadcasting another "iterate" command unless the residual reached the threshold ε.

The addition of new pages during incremental crawling may happen at any time. If the addition covers new hosts, the coordinator selects a node according to the current balancing. From then on, this node is responsible for all pages of that host. The assignment is broadcast to all nodes in case that there were dangling links to that (previously uncovered) host.

Results

Most of the experiments have been on four Linux machines, an AMD Dual Opteron 850 2.4 GHz, 10GB RAM ("A"), an Intel Dual Xeon 2.8 GHz, 6GB RAM ("B") and two Intel Xeon 3.0 GHz, 1.5GB RAM ("C" and "D"). They were connected via 100MBit Ethernet LAN and not under load before the experiments. I divided the LargeWeb graph into eight partitions and distributed them among the four servers according to available memory (Machine A holds four partitions, B two, C and D one) and performed unbiased PageRank computations.

I examined the convergence behavior, rank distribution and elapsed time both globally and per-partition. All per-partition results matched almost perfectly with the global counterpart and therefore confirmed our assumptions (see Figure 3.6). The PageRank computation converged below $\varepsilon = 10^{-3}$ after 17 iterations. The entire computation took less than 9 minutes, with only 66 seconds accounted for rank computation, the rest being network I/O. With a Gigabit-Ethernet connection, network communication costs would probably go down to the same magnitude as computation costs. Compared to the running times of a centralized PageRank computation with disk I/O, using our networked servers, parallel PageRank is about 10 times faster per iteration. The recomputation itself (ignoring network transmission) was about 75 times faster.

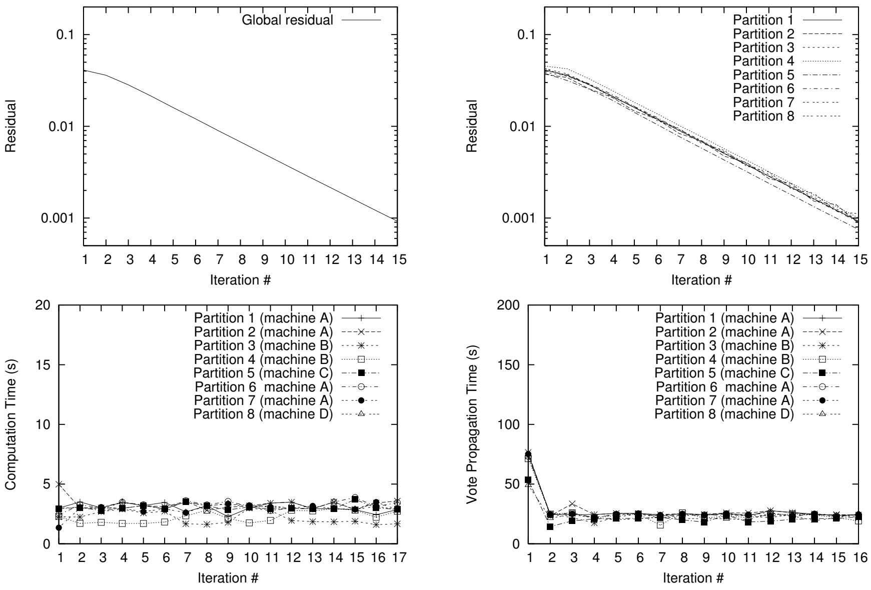


Figure 3.6: Partitioned PageRank convergence, vote calculation and network communication times using 8 partitions on 4 machines; $\varepsilon = 0.001$

3.2.4 Discussion

I have presented an efficient method to perform the PageRank calculation in parallel over arbitrary large web graphs. We accomplish this by introducing a novel two-dimensional view of the web, having the host ID as the only discriminator, as well as by adapting the Gauß-Seidel method for solving linear systems in this scenario. Additionally, I have presented optimizations for the distributed computation, such as vote aggregation and utilizing the partitioning scheme for fast re-balancing in the course of incremental crawling.

Of note, even though my algorithm was initially published in 2006, it may still be regarded one of the most computationally efficient methods [84]. It might be interesting how the algorithm performs on a massive parallel machine and when other PageRank-specific enhancements that reduce convergence time (for example, through less coordination operations), under extensive memory demanding scenarios.

A large-scale application of my algorithm for topic detection follows below.

3.3 Topic-specific PageRank

Full-text search is probably one of the most important facilities to access documents in the Web. Unlike controlled collections such as digital libraries, the Web does not have a rich set of annotations. Consequently, when the user wants to focus her query to a specific subject, she has to reformulate it with additional terms describing her topic of interest. Yet this also implies that the set of possible results is restricted to those documents which *contain* the given query terms. If the user wants for example to find "sales contact" persons in the topic of "Business concerning natural textile fabrics", she has to express all this information as terms. This query augmentation will clearly deprive her from finding most pages containing only the phrase "sales contact" and the name of some textile company.

Since most queries submitted to Web search engines consist only of very few keywords, search results are susceptible to be implicitly biased towards generally popular web sites. This is due to enriching text retrieval methods like TFxIDF with link analysis algorithms as PageRank [92]. A promising approach to solve this dilemma of under- and over-specification was to bias PageRank to favor a specific set of pages, called *biasing set* [54]. In most cases these biasing sets have been selected as subcategories of given large scale taxonomies, such as the Open Directory (ODP)[2].

Although there exist a few prior studies analyzing the properties of such topically biased PageRank [29], many aspects remained unstudied. In this thesis I complete the investigation. We perform a utility analysis for topically biased PageRank and clarify the relation between the parameters of an ODP category (e.g., depth, number of children and siblings, number of pages therein, etc.) and the quality of the resulted biased rankings. I also investigate the correlation between the biased ranking and the generic, non-biased one. Finally, I sketch some applications of biased PageRank which could benefit from our study.

3.3.1 Deeper inside ODP

Setup. I empirically analyzed the quality of the ODP-biased PageRank vectors using both quantitative measures, i.e., Kendall Tau similarity [68], and qualitative ones, i.e., Mean Average Precision (MAP). The testbed was a 9.2M document web graph focused on the ODP catalog, which I have recently gathered using the Heritrix[3] crawler. About 100 biasing (sub-)categories were randomly chosen from four top level categories, namely Business, Computers, Recreation and Sports. This selection process was executed as follows: For each of the four top categories, three subcategories were randomly picked; then, for one of them, again randomly three subcategories were taken and so on, until no deeper levels were available. Almost all paths ended at level 6 (with level 1 being one of the ODP root categories). Finally, Biased PageRank vectors were computed using the pages residing in each of these categories as biasing sets and using my parallel PageRank implementation presented in Section 3.2.

I also selected five queries per category randomly using Google AdWords[4], which suggests commonly used query terms to some specific keywords of interest. Whenever

[2] http://dmoz.org

[3] http://crawler.archive.org/

[4] http://adwords.google.com/

such a query resulted in less than one hundred results within our local index, it was replaced by another one, randomly selected as well. Nevertheless, in most cases several thousands of results were obtained per query. Note that these queries are implicitly focused on each given ODP topic, and thus they should have resulted in rather similar outputs for Non-biased and Biased PageRank.

Finally, I performed searches using the generated queries and Biased PageRank for each associated category, as well as its parent and each of its child categories. Moreover, also unbiased searches (with regular PageRank) were performed for each query. In all cases, the output results were sorted by Lucene[5] TFxIDF-based score multiplied with the specific (Biased) PageRank scores. For the quantitative analysis, the top-30 matches from each result list were compared using Kendall Tau, whereas for the qualitative one, Mean Average Precision was employed on the top-10 results. Three persons evaluated *all* search results, rating them with 1 if they were relevant both to the given query and category, and with 0 otherwise. The MAP scores for each (query, category) pair were averaged over all subjects to obtain a single value per pair. These were then further averaged over all queries, thus calculating a MAP for each category, as used in Figure 3.7.

Results. In order to visualize the results we model the categories as a directed hierarchical graph. Figure 3.7 presents a fragment of that graph corresponding to the top category `/Business`), which is representative for the remaining graph as well. Nodes represent categories and edges between them denote parent–child or child–parent relationships. An edge's width depicts the (averaged) Kendall similarity between the two categories. The thicker it is, the more similar the linked categories are. A node's contour line width represents the ratio between MAP for Biased PageRank and MAP for Non-biased PageRank (marked as "NoBias"). Again, the thicker this line is, the higher is the precision for Biased PageRank when compared to NoBias[6].

We now summarize our results as follows:

- *There is no relationship between the Kendall similarity of Biased and Unbiased PageRank (edge weights) and the category level.* Even though one would expect

[5] `http://lucene.apache.org/`

[6] For /Business/Textiles_and_Nonwovens/Textiles/Fabrics, MAP for NoBias was 0; it is depicted as a dashed line.

lower categories to produce results more similar to each other (as their biasing sets become rather small), this phenomenon does not always occur. More, there are higher level categories whose Biased PageRank vectors are quite similar (e.g., `Textiles` / `Textiles_And_Nonwovens`), although their biasing sets are larger.

- *The size of the biasing set neither correlates with the Kendall similarity, nor with the PageRank quality (in terms of MAP).* Large biasing sets may result in both high and minimal improvements over non-biased PageRank. I thus suspect that a higher correlation might be achieved when comparing the *connectivity* of the pages within each biasing set with MAP. However, if this connectivity is expressed in terms of total amount of out-links, again no correlation occurs.
- *The MAP ratings generally increase until ODP level five, and then drop sharply.* This shows that bottom level ODP categories tend to be *less useful* biasing sets as page amount and connectivity are rather low.
- *MAP is not correlated with the Kendall similarity.*
- *Kendall similarity to Unbiased PageRank almost always tends to 0.* This is quite important, as it shows that biasing *does* have a significant impact on ranking.
- *Kendall similarities between parent and child categories are generally very low* (< 0.2). This indicates that it would be useful to employ more specialized (deeper) categories to bias PageRank on, rather than using the top-level categories only as in previous work.
- *Kendall similarities between sibling categories are generally very low* (< 0.2; see the upper right part of the figure for an excerpt of such similarities). Thus, ODP sibling categories are well defined, being quite distinct from each other.

Practical Applications. It is important to note that biasing PageRank using ODP is highly useful in many applications. To name but a few, it can be employed for (1) Personalized Web Search (i.e., bias on user's topics of interest), (2) Faceted Search (i.e., promote the selected facet by biasing), (3) Automatic Extension of the ODP (i.e., derive new qualitative pages to add into each category), etc.

3.3.2 Discussion

In this section, I presented a quality analysis of Biased PageRank under different, nested categories of the Open Directory taxonomy. It could be shown that the MAP quality of Biased PageRank generally increases with the ODP level, yet it also starts dropping sharply at some point, when the amount and connectivity of the pages contained within that category level are too low. Moreover, biasing on different siblings or on children of a given category generally yields quite different outputs, thus sustaining the usage of more specialized (deeper) categories to bias PageRank on, in order to obtain a better search outcome.

As computing Biased PageRank for all possible ODP categories is still rather time consuming, future work should focus on algorithms based upon these findings to automatically select only those categories which yield search results very different from regular PageRank, while also significantly improving its quality.

3.4 Using PageRank for Faceted Search

It is often difficult to find terms which precisely separate relevant from irrelevant pages, because they are either ambiguous or can at least be seen from different perspectives. While modern search engines may satisfy the average user's needs by focusing on general importance (i.e., a keyword search for "bush" primarily returns pages about *George W. Bush*), as soon as other aspects are concerned, the only way out is to specify additional keywords (i.e., do a search for "bush gardening" to focus on horticultural issues) and hope that these auxiliary terms are not too restrictive. But the more aspects are specified as terms (such as "scientific article" and "British"), chances are high that many highly relevant documents are filtered out because not all terms are matched [57].

The emerging paradigm of Faceted Search [57, 122, 35, 36] may help in this situation. Here, the system automatically determines possible aspects from the result set and presents them to the user, modeled as categories of orthogonal dimensions

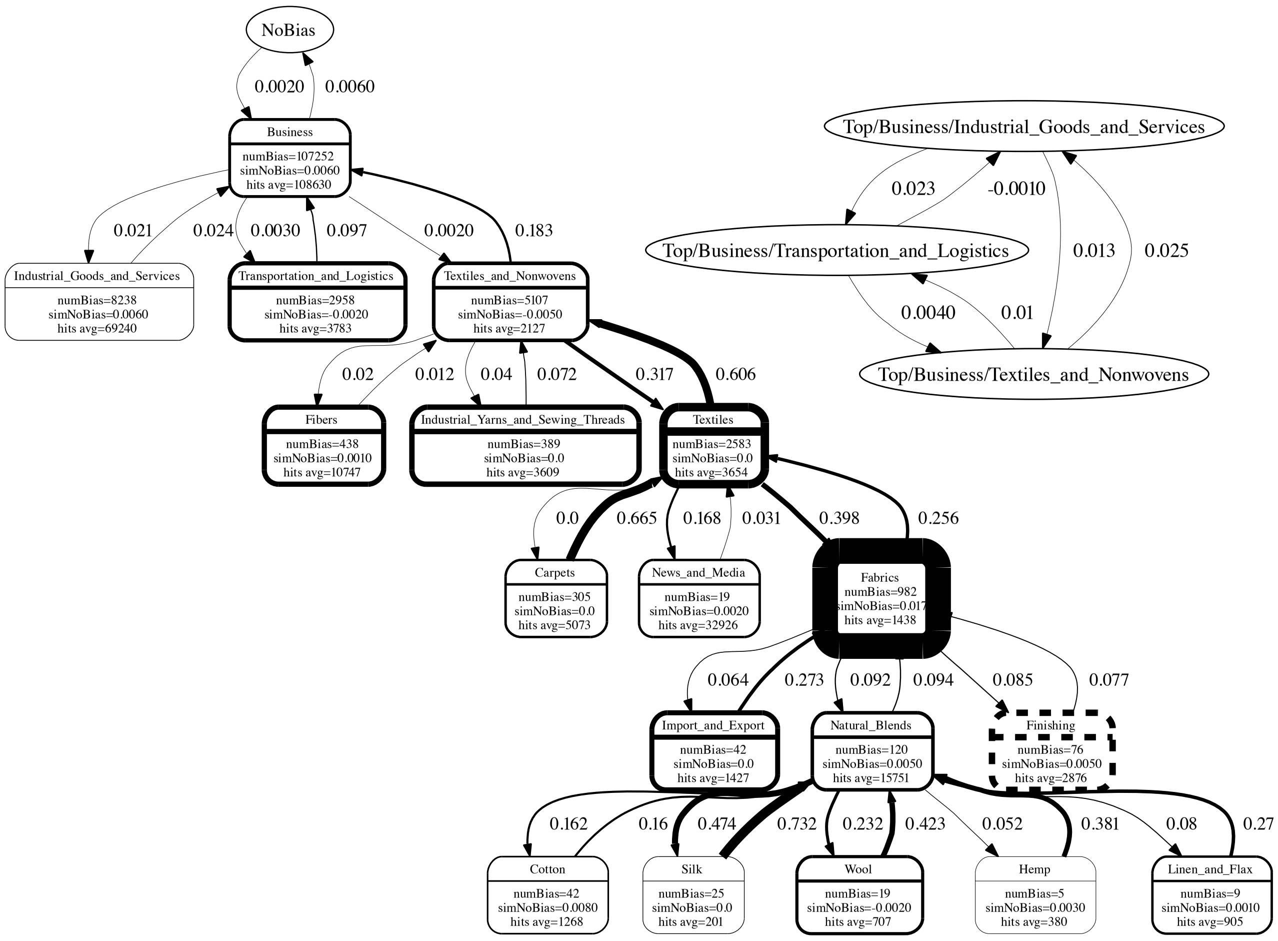


Figure 3.7: Rank Similarities for the “Business” branch of ODP categories

like topic, cultural background or target audience and finally enables the user to iteratively narrow ("drill-down") the search until he is satisfied with the results. These "facets"[7] are represented as metadata, they do not interfere with full-text keywords as opposed to text-based clustering approaches [57]. While faceted search works well in situations where facet metadata is annotated to the documents like in library systems or enterprise search applications (e.g., restaurant search), on dynamic, large scale collections of heterogeneous documents such as the Web, the faceted classification of pages and the identification of facets within search results still are unresolved problems. The lack of annotations and the omnipresence of noise within this collection hamper a direct adoption of enterprise faceted search.

In this section, I focus on the question how we can still utilize the few annotations from taxonomies or folksonomies like ODP or del.icio.us for faceted web search. I present an efficient method for automatically identifying facets in web search results solely by link analysis. First experiments have shown that we get a high precision of the suggested faceted classifications without requiring additional data structures or categorization algorithms.

3.4.1 Relevant Background

Faceted search. Faceted search is a relatively young research area, and thus there exist only few approaches to tackle the problems it raises, especially when generalizing this kind of search for the entire web environment. Facets allow for different views on the result set, which can be obtained through a specialized user interface [122], thus enabling the user to choose different possible starting paths for the exploration of the collection. The Flamenco System[8] for example allows for both searching and browsing an image collection from various perspectives, such as gender, country of affiliation for Nobel prize winners. There exist at least two types of facets: (1) Hierarchical and (2) Flat. The concept of Hierarchical Faceted Categories (HFC) was introduced by Hearst et al. in [58, 122], and it relies on a set of category hierarchies (one per facet),

[7] Note: In this context, I use the term "facet" as a synonym to "aspect" or "category", whereas I call the orthogonal axes "facet dimensions".

[8] `http://flamenco.sims.berkeley.edu/`

built manually in advance. Thus, they classify the documents available in a collection space according to each of the hierarchies separately, producing several navigation paths across the same set of points. Inherently, users will find their sought documents faster, as more routes towards them exist. The main approach we discuss here, i.e. using "page topic(s)" as one facet for categorizing web results, also falls in the HFC type. The second type of facets is a flat one, in which there is no clear relation between the elements generated within the same facet dimension. I am not aware of any prior work creating this kind of facets from within textual web documents. Whereas initially facets were thought of purely independent, orthogonal and predefined dimensions, extensions beyond this basic approach allow the exploration of correlated and dynamic facets [13] and approach an optimal ranking of facet dimensions by modeling the interaction of users with faceted search engines [83].

Large scale taxonomies / folksonomies. One of the largest efforts to manually annotate web pages is the Open Directory Project (ODP)[9]. Over 85,000 editors helped to categorize more than 5 million web sites into almost 600,000 hierarchical categories describing web sites' topics; however, these pages still make far less than 0.1 percent of the publicly accessible web. While in ODP, the taxonomy is clearly split into 16 root categories, such as Business, Computers or Sports, "folksonomical" organized platforms like del.icio.us[10] allow users to annotate arbitrary tags to web pages rather than using a fixed taxonomy [88, 50]. Since both approaches allow to classify pages to more than just one facet (category or tag), faceted search could easily be implemented for the pages annotated in these collections, but not for the whole web graph.

3.4.2 Web Page Classification using Personalized PageRank

Several aspects of a single page may be considered relevant for faceted search; they can be ordered into facet dimensions like thematical coverage, language, age etc. To automatically retrieve such facets, classification algorithms can be applied to receive the set of relevant facets M_p for each page p in the web graph. If possible, a relevance measure is assigned to each facet, resulting in a relevance vector $\vec{m}_p$ whereas each

[9] `http://dmoz.org/`

[10] `http://del.icio.us/`

dimension represents one of all available facets. In classical IR, to compute $\vec{m}_p$ text-based models are used, such as Naive Bayes or Latent Semantic Analysis. For the Web, one can consider the hyperlink structure of the web as another source of classification power. Several approaches exist which either combine link structure and textual information or even attempt to derive a classification from the web graph only [39, 85, 121, 32, 40]. While they may perform very well, in all cases the only purpose of the proposed algorithms is to generate a classification being not related to PageRank; the imposed additional payload (computational and storage requirements) to the search engine system may not be underestimated. In this section, I present an efficient method which re-uses the rank vector of Personalized PageRank for classification, avoiding such "overhead".

The results shown in Section 3.3 support the assumption that Personalized PageRank vectors themselves provide sufficient information for classifying pages towards arbitrary facets. Because of the skewed jump in the PageRank computation, pages within the corresponding biasing set tend to get higher scores than pages outside. Via the link structure also pages being outside but nearby the biasing set (in terms of link hops) gain high scores [92]. Assuming that pages mostly link to *somehow* related pages, there consequently must be a direct relationship between a page's facet-specific PageRank score $r_f(p)$ and its membership $m_p(f)$ to that facet.

$$m_p(f) \propto r_f(p) \tag{3.12}$$

We can treat the values as qualifiable membership to the facet f (member of the set of available facets F); values above the average score $\mu \approx 1.0$ are treated as "positive" membership, values below as "negative" membership (i.e., non-relevant to the facet). One may assume that the membership value is *equal* to the rank value, or that a high rank score automatically means a high relevance to the facet (as suggested in [54]). However, since there also is a fair amount of pages of general importance/interest which may also have a high score [40], a direct-proportional relationship turns out to be too imprecise. I therefore introduce the facet membership *uncertainty* $\alpha_f(p)$ for the page p: the more facets are possible the less reliable the membership statements are.

Since facet dimensions are usually designed to be orthogonal, this uncertainty should depend on the number of facets in the very same dimension dim as the considered facet f. In case that no facets could be determined, the uncertainty is infinite:

$$G_f(p) = \{g \in F \mid dim(g) = dim(f) \ \wedge \ r_g(p) \geq \mu\} \tag{3.13}$$

$$\alpha_f(p) = \begin{cases} \infty & \text{iff } G_f(p) = \emptyset \\ |G_f(p)| & \text{otherwise} \end{cases} \tag{3.14}$$

To account for the exponential nature of the rank score, we relate the score's logarithm to α_f (otherwise, r_f would dominate in this function):

$$m_p(f) = \frac{log\ r_f(p)}{\alpha_f(p)} \tag{3.15}$$

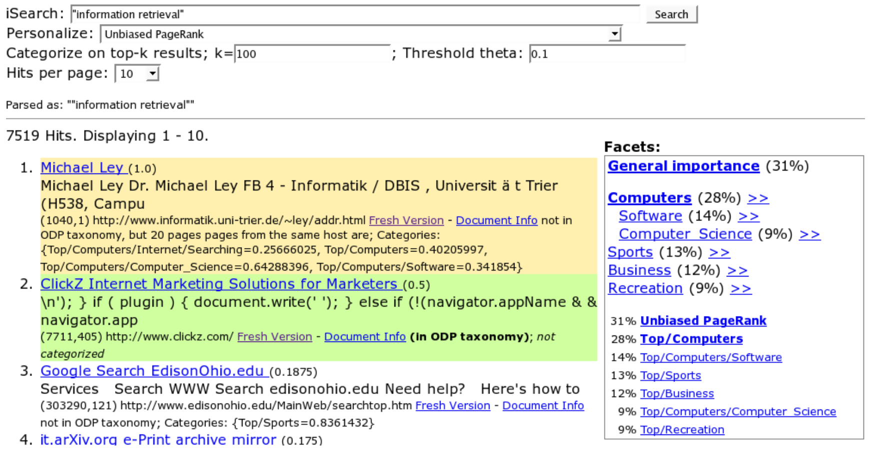
$m_p(f)$ now also *quantitatively* reflects positive or negative membership certainty[11]. Having said that, a value around zero does not implicate that a page does not relate to a specific facet – it is just unclear. This case applies to commonly linked pages like the *Acrobat Reader download* page, the *phpBB bulletin board system* website and others. On the other hand, specialty pages clearly benefit from this classification.

3.4.3 Identifying Facets from Web Search Results

When submitting a text query q on a PageRank-supported web search engine, the top-k of the returned result list represents the k most relevant results for the query terms with respect to general importance (for unbiased PageRank) or to specific facets (for personalized PageRank). For a sufficiently large k, this list could be regarded as a sample set S_q of the whole result set D_q; another approach to determining S_q, taking care of strongly over-represented facets, is shown in [6]. Now, we can compute the facet membership vector $\vec{m_p}$ for all pages $p \in S_q$. From this, we can derive the facet membership vector $\vec{m_{S,q}}$, which represents the facet memberships for the given top-k subset S_q of results for the query q. In other words, this vector represents the facets which are deemed to be contained in the full result set D_q.

[11] We shall avoid the term *probability*; neither $r_f(p)$ nor $\alpha_f(p)$ are bounded.

Figure 3.8: Screen-shot of the prototype



iSearch: "information retrieval"

Personalize: Unbiased PageRank

Categorize on top-k results; k=100 ; Threshold theta: 0.1

Hits per page: 10

Parsed as: ""information retrieval""

7519 Hits. Displaying 1 - 10.

1. Michael Ley (1.0)
   Michael Ley Dr. Michael Ley FB 4 - Informatik / DBIS , Universit ä t Trier (H538, Campu
   (1040,1) http://www.informatik.uni-trier.de/~ley/addr.html Fresh Version - Document Info not in ODP taxonomy, but 20 pages pages from the same host are; Categories: {Top/Computers/Internet/Searching=0.25666025, Top/Computers=0.40205997, Top/Computers/Computer_Science=0.64288396, Top/Computers/Software=0.341854}
2. ClickZ Internet Marketing Solutions for Marketers (0.5)
   \n'); } if (plugin) { document.write(' '); } else if (!(navigator.appName & & navigator.app
   (7711,405) http://www.clickz.com/ Fresh Version - Document Info **(in ODP taxonomy)**; *not categorized*
3. Google Search EdisonOhio.edu (0.1875)
   Services Search WWW Search edisonohio.edu Need help? Here's how to
   (303290,121) http://www.edisonohio.edu/MainWeb/searchtop.htm Fresh Version - Document Info not in ODP taxonomy; Categories: {Top/Sports=0.8361432}
4. it.arXiv.org e-Print archive mirror (0.175)

Facets:

General importance (31%)

Computers (28%) >>
  Software (14%) >>
  Computer Science (9%) >>
Sports (13%) >>
Business (12%) >>
Recreation (9%) >>

31% **Unbiased PageRank**
28% **Top/Computers**
14% Top/Computers/Software
13% Top/Sports
12% Top/Business
9% Top/Computers/Computer_Science
9% Top/Recreation

In order to compute $\vec{m_{S,q}}$ we use the following procedure: For each facet membership value (in each vector $\vec{m_p}$, for all pages $p \in S_q$) which exceeds a certain threshold ϑ, we increment the corresponding value $m_{S,q}(f)$ by one; finally, $\vec{m_{S,q}}$ is normalized to values between 0 and 1.0:

$$m_{S,q}(f) = \frac{|\{p \in S_q | m_p(f) \geq \vartheta\}|}{|S_q|} \tag{3.16}$$

As an improvement to facet diversity, I suggest to compute the membership values not only from one sample set but from multiple PageRank result sets. For example, with the ODP taxonomy, we start with sampling the top-k results from unbiased PageRank as well as all personalized PageRank vectors based on the taxonomy's 16 top categories. If the user has already chosen an ODP-topic as a facet, we take the corresponding child categories instead.

Another way to extend the set of facets, which can also be applied to non-hierarchical facets, is to start from a custom seed of facets and then iteratively determine relevant facets using the formula above and extend this seed by newly detected facets D (see algorithm 3.1).

Algorithm 3.1 Iterative identification of facets

```
$\lambda$ $=$ Membership threshold for newly discovered facets
$D$ $\leftarrow \emptyset$
$D'$ $\leftarrow$ Facet seed
$i_{max}$ $\leftarrow$ Maximum number of iterations
$i$ $\leftarrow 0$
while $i < i_{max} \wedge D' \setminus D \neq \emptyset <$ do
    $i \leftarrow i + 1$
    $D \leftarrow D \cup D'$
    $D' \leftarrow \{g \in F \mid m_{q,D}(g) \geq \lambda\}$ ;
        $m_{q,D}(f) = \frac{\sum_{d \in D} |\{p \in S_{d,q} | m_p(f) \geq \vartheta\}|}{\sum_{d \in D} |S_{d,q}|}$
end while
```

The user can now select from the facets given in D for the refinement of his search. If more than one facet is selected, personalized PageRank is used based on the linear combination of the facets' rank vectors, as described in Section 3.4.1. As opposed to Topic-Sensitive PageRank, it is not necessary to derive the combination weights from textual document statistics. Instead, we may propose predefined weights on the level of facet dimensions (e.g., prefer topic over origin), which can be altered by the user; facets of the same dimension may safely be equally balanced.

3.4.4 Implementation and Evaluation

Right now, the facet detection implementation only covers one dimension, the topical membership of page, based on the ODP taxonomy; the system could easily be extended to more dimensions, domain name features and type of information source (academia, business, government, media, private etc.), though. A screen-shot of the prototype is depicted in figure 3.8. The search engine prototype is based upon the Lucene[12] information retrieval software library, which I extended to hold multiple PageRank scores per document. As a search basis the 9.2 million documents crawled for the ODP experiment (Section 3.3) was used.

[12] http://lucene.apache.org/

Any keyword search is conveyed using unbiased PageRank by default; however the user can manually choose any available biasing set. The search results are classified on-line using equation 3.15; since the classification is fuzzy and the operation can be performed quickly (no access to textual data is required), I decided not to store this information statically in the index. For the top-k results ($k = 100$ per default), we identify the contained facets having a membership certainty of at least ϑ (0.1 per default). Both parameters can be adjusted by the expert user. The detected facets are presented next to the document descriptions; facets based on ODP topics are shown both as a tree as well as a sorted list, along with the facet membership value from algorithm 3.1. The user can choose from these facets by clicking on the corresponding facet name (facets with a membership of at least 25% are highlighted). The search is then repeated using the corresponding personalized PageRank.

Unbiased PageRank is treated as a special facet. It may also receive a membership value. Intuitively, this facet can be regarded as general importance. If no other facet has been retrieved for the keyword query, I conclude that we lack the proper Personalized PageRank. From this perspective, the membership to “general importance” can be interpreted as uncertainty as well.

I performed a preliminary evaluation of the facet detection algorithm. I determined a set of phrase queries which are relevant to specific ODP categories (33 in the test set, 5 keywords each) and searched for these phrases with our system (i_{max} was 1). In order to avoid falsified results, I did not distill these queries from my own data set, but utilized Google’s AdWords service[13] (we randomly selected 5 suggested queries; an excerpt is shown in Table 3.3). Then, with the help of two colleagues, I compared the category recommendations from AdWords with the topical facet recommendations from the test system. Exact phrase search returned results in 155 out of the 165 cases. Only in 13 cases, the algorithm did neither retrieve the desired category nor any ancestor category (this makes a precision of 91.6%).

[13] `https://adwords.google.com/`

Table 3.3: Sample keywords from Google AdWords

Business: information technology; market share; strategic planning; supply chain management; swot analysis
Business/Textiles and Nonwovens/ Textiles/ Carpets: rugs; floor covering; persian carpet; oriental weavers; hardwood floor
Computers: laptops; workstations; flat screen; second hand computers; midwest micro
Computers/Internet/Searching: search the web; web browser; front crawl; search engines; internet searching
Recreation: recreation jobs; parks and recreation; nude recreation; recreation and leisure; lake mead recreation
Recreation/Travel/Travelogues: travelogue game; travel diary; travelogues; travel adventures; travel phots
Sports: nfl; athletics; stadiums; formula one; basketball
Sports/Soccer/UEFA/England/Women: women soccer uk; football girls; ladies soccer; girls fc; manchester ladies

3.4.5 Discussion

I presented an approach to drill-down from hierarchical faceted search and to simply re-order web search results according to a specific category within a facet using Personalized PageRank. The current implementation covers output categories for the "topic" facet, thus enabling users to on-the-fly switch to result rankings according to PageRank biased on some Open Directory topic. I have also proposed a new simple technique to infer the most relevant topics associated to a user query. Experimental results have shown this approach to yield precise identification of facets.

Further work may explore different variations of the membership function to improve precision and deeper inspection of the search results to improve recall. The latter may strongly benefit from computing membership values at index time (offline) instead of online at search time. Then, storing those categories with a sufficiently high membership certainty directly in the index allows to quickly enumerate them at runtime for the whole result set (just as in classic Faceted Search).

Of note, a strikingly similar approach (using “merit values” instead of membership certainties) has been filed for U.S. patent by a third party, shortly after my initial publication in 2006, and been granted in 2009.[14] It remains to be seen what one may conclude from the potentially subtle differences.

[14] United States Patent US7493330, `http://www.freepatentsonline.com/7493330.pdf`

Chapter 4

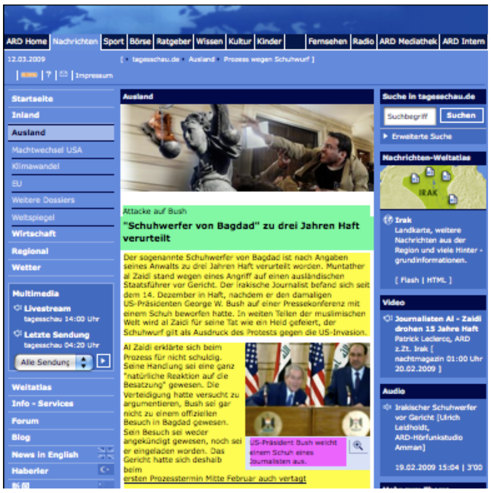
Text Structure

In this chapter, we deal with the second problem: How can we – at Web scale – model and utilize the inherent textual structure of a Web page for improving search quality?

4.1 The Block-Level Nature of Web Text

Compared to the early times of the Web, where individual HTML documents more or less represented one textual document [14], identifying and retrieving distinct information elements has now increasingly become difficult. Besides the *main content* (e.g., an article) modern web pages also contain a bouquet of other textual elements such as navigation menus, advertisements, user comments, text ads, snippet previews of related documents, legal disclaimers etc. (see Figure 4.1) Separating (= segmenting) these distinct elements and eventually classifying them into relevant and non-relevant parts is essential for high-quality results. When examining a Web page, humans can easily distinguish the main content from these other text portions, which are mainly meant to augment the full-text [47]. These additional text segments provided seem only be partially useful or probably even counterproductive for search and classification; the common solution to the problem is simply erasing template content or at least ignoring it.

A number of approaches have been introduced to automatize this distinction, using a combination of heuristic segmentation and features. In this thesis, I approach the



Ausland

Attacke auf Bush

"Schuhwerfer von Bagdad" zu drei Jahren Haft verurteilt

Der sogenannte Schuhwerfer von Bagdad ist nach Angaben seines Anwalts zu drei Jahren Haft verurteilt worden. Muntather al Zaidi stand wegen eines Angriff auf einen ausländischen Staatsführer vor Gericht. Der irakische Journalist befand sich seit dem 14. Dezember in Haft, nachdem er den damaligen US-Präsidenten George W. Bush auf einer Pressekonferenz mit einem Schuh beworfen hatte. In weiten Teilen der muslimischen Welt wird al Zaidi für seine Tat wie ein Held gefeiert, der Schuhwurf gilt als Ausdruck des Protests gegen die US-Invasion.

Al Zaidi erklärte sich beim Prozess für nicht schuldig. Seine Handlung sei eine ganz "natürliche Reaktion auf die Besatzung" gewesen. Die Verteidigung hatte versucht zu argumentieren, Bush sei gar nicht zu einem offiziellen Besuch in Bagdad gewesen. Sein Besuch sei weder angekündigt gewesen, noch sei er eingeladen worden. Das Gericht hatte sich deshalb beim ersten Prozesstermin Mitte Februar auch vertagt

US-Präsident Bush weicht einem Schuh eines Journalisten aus.

Figure 4.1: A typical modern Web page with large navigation and related material (content highlighted)

problem from a Quantitative Linguistic perspective. I consider three key application areas for web page segmentation: *(1) De-duplication.* Identical content information may be presented using different web page layouts. *(2) Content Extraction.* Besides the obvious benefits for Web-based news clipping etc., removing template noise might also increase classifier performance. *(3) Keyword-based Web search.* A page should be regarded less relevant to the query if the matched term only occurs in a template segment.

4.2 Web Page Segmentation

Until now, the segmentation problem has mainly been addressed by analyzing the DOM (Document Object Model) structure of an HTML page, either by rendering and visual analysis or by interpreting or learning the meaning and importance of tag structures in some way, both using heuristic as well as formalized, principled approaches. However, the number of possible DOM layout patterns is virtually infinite, which inescapably leads to errors when moving from training data to Web-scale. The actual retrievable unit – namely *text* – has only partially been investigated for the purpose of web page segmentation. Whereas it has been analyzed on the level of semantics and on term-level, a low-level pattern analysis is still missing.

Attempts to Web page segmentation consider a variety of methods from different aspects. Most commonly, the structure of the web page (i.e., the DOM tree) is analyzed, in order to mine block-specific patterns, for example to separate and remove template elements from the actual main content. Bar-Yossef and Rajagopalan [11] identify template blocks by finding common shingles, similar to Gibson et al. [47] who also considers element frequencies for template detection. Debnath et al. compute an inverse block frequency for classification [33]. In [24], Chakrabarti et al. determine the "templateness" of DOM nodes by regularized isotonic regression. Yi et al. simplify the DOM structure by deriving a so-called Site Style Tree which is then used for classification [123]. Vieira et al. present an approach to template removal by identifying common DOM subtrees from a sample set and removing these structures from the whole collection [114]. Kao et al. separate blocks of DOM subtrees by comparing the entropies of the contained terms [67]. Vision-based approaches add information gained after rendering the DOM, such as Cai et al.'s VIPS algorithm [22], Chen et al.'s approach to tag pattern recognition [28] as well as Baluja's [10] method using decision tree learning and entropy reduction. Chakrabarti et al. approached the web-page segmentation problem from a graph-theoretic perspective [25]. As shown by Cai et al. [23] and more recently by Fernandes et al. [43] the resulting segment structure can also be used for improving keyword-based search. Finally, Fauzi et al. focus on segmenting pages to extract images and their surrounding context [42].

In this section, I will

1. define an abstract block-level page segmentation model which focuses on the low-level properties of text instead of DOM-structural information,

2. concretize this abstract model: the key observation is that the number of tokens in a text fragment (or more precisely, its *token density)* is a valuable feature for segmentation decisions. This allows us to reduce the page segmentation problem to a 1D-partitioning problem,

3. present the Block Fusion algorithm for identifying segments using the text density metric,
4. present an empirical analysis of my algorithm and the block structure of web pages and evaluate the results, comparing with existing approaches.

4.2.1 Problem Discussion

Segmentation as a Visual Problem

It is surprising how different the visual representation and the corresponding HTML document structure can be across different websites. Not only the use of different layouts contributes to this situation, but also the fact that there are versatile ways to model an identical layout, e.g. by varying between semantic and visual markup (`<EM>` vs. `<I>`), misusing `<TABLE>` structures for positioning non-tabular elements as well as completely neglecting HTML semantics for the layout. The latter has become very popular due to the use of CSS across most Web 2.0 websites, where tags usually are just `<DIV>` elements. This situation makes web page segmentation a non-trivial task. On Web-scale, rule-based or trained algorithms working on DOM-level are, due to the extreme heterogeneity of HTML style, susceptible to failure. On the other hand, vision-based approaches naturally have a higher complexity since the layout must be rendered (like in a browser) prior to analysis, which might be too slow to be incorporated into the Web crawling and indexing cycle.

Although we are examining the problem of web page segmentation from a textual perspective, there is a clear relationship to image segmentation from the field of Computer Vision: any DOM-level algorithm has to bear comparison with image recognition approaches, which span from k-means pixel clustering over histogram mode seeking and graph-partitioning to greedy region merging strategies [110]. In fact, we can draw a parallel from Shi's normalized cuts graph partitioning technique [110] to the recent work of Chakrabarti et al. [25], for instance. An example of a graph-independent approach is Haralick's and Shapiro's Region Growing [110]. Region Growing essentially is a greedy merging strategy; starting from one position in the image (e.g., top left corner), the grower iteratively fuses neighbored regions to larger ones (i.e., distinct pixels to sets of adjacent pixels). Under the assumption that the pixels are independent and uniformly distributed, the similarity of a region to another one is quantified by the deviation from the average *intensity* of that region – regions are merged if the deviation is insignificant. An application for Region Growing is text-block extraction from scanned newspaper images, where the algorithm is also known as Block Growing [26, 7].

Segmentation as a Linguistic Problem

In the field of Quantitative Linguistics, distributions of linguistic units such as words, syllables and sentences have been widely used as statistical measures to identify structural patterns in plain text documents, in particular for identifying subtopics [56] as well as for discovering changes of writing style [2] – both can be regarded as a special form of segmentation. In this discipline, it is a generally accepted assumption that the probability of a given class x in the corresponding unit's distribution is univariately dependent on the probability of the neighboring lower class $x-1$ [52]:

$$P_x = g(x)\, P_{x-1} \tag{4.1}$$

For example, when a text is segmented into blocks of almost the same size, it is believed that the class distribution of term frequencies (occurrence probabilities) is negative hypergeometric (*Frumkina's law* or *law of text blocks*), which has been validated for various languages [16]:

$$P_x = \frac{\begin{pmatrix} -M \\ x \end{pmatrix}\begin{pmatrix} -K+M \\ n-x \end{pmatrix}}{\begin{pmatrix} -K \\ n \end{pmatrix}} \;,\; x = 0, 1, 2, ..., n \tag{4.2}$$

Taking this into account for segmentation, an obvious strategy is to examine the statistical properties of *subsequent* blocks with respect to their quantitative properties.

In [56], for example, Hearst presents an algorithm which discovers sequences of adjacent paragraphs that belong to a topical unit within the document; which paragraphs get assigned to a particular subtopic is decided by a neighbored-block comparison based on term-frequency and cosine similarity.

Besides such an analysis of documents on the term-level, there are further interesting quantitative properties to consider. The distribution of document lengths follows the well-known Zipf distribution [34]. It might be reasonable to consider this distribution for segmenting intra-document text portions as well. Zipf's law states that the occurrence frequency of objects of a particular class is roughly inversely proportional to the corresponding rank of the class:

$$y = C\ x^{-b} \tag{4.3}$$

Another efficient quantum is *sentence length.* According to Altmann [2, 15], the creation of sentences is a stochastic process which follows a rhythm based on certain synergetic properties, i.e. the sentence lengths change along with the text flow. For analyzing changes in writing style he thus recommends not to compare random samples of a document but consecutive sentences instead. He concludes that also the occurrence probability of a particular sentence length x is a function of $x-1$ (yielding a hyperpascal distribution):

$$D_x = \frac{P_x - P_{x-1}}{P_x} \tag{4.4}$$

Segmentation as a Densitometric Problem

Coming back to the problem of *web page* segmentation, it is questionable whether the particular use of one specific HTML formatting style yields better signals for finding the "right" segmentation than another one. It is obvious that the *absence* of element tag information is a strong indicator for the segmental unity of a text portion. I consider such text portions *atomic*. Could then perhaps the sheer *presence* of any element tag already be a sufficiently good signal for segmentation? While there are a few tags which separate by high chance (heading tags such as `<H1>`) and some which usually do not separate (the anchor text tag `<A>`), the majority of elements has unclear effects to segmentation. Thus, we may simply model a web page as a series of text portions (non-segmentable*, atomic blocks*) interleaved by a sequence of one or more opening or closing element tags, regardless of their meaning. I call such a sequence a *gap*. This simplifies the discussion to distinguishing the gaps which separate two segments and gaps which do not. Non-separating gaps may be discarded, resulting in larger text segments (*compound blocks*). While we can always *a priori* define certain tag-based rules for this decision finding, I focus on analyzing the blocks' inherent textual properties for this purpose.

Most likely, a segment gap is caused by a change in the text flow, e.g. from a list of short phrases (navigational text portions like "Home", "What's new", "Contact us") over a sequence of full sentences (for the main content) back to short phrases or one-sentence blocks for the page footer (e.g., "Copyright (c) 2008 by ... All rights reserved"). This setting is similar to the analysis of *writing style* by comparing sentence lengths (see Section 4.2.1). Due to the lack of proper sentences in template elements, it is difficult to define "sentence" in the web page scenario. Instead, we may substitute sentence length by *text density*, i.e. the number of words within a particular 2-dimensional area. Text density has been defined by Spool et al. in the field of Web Usability as the ratio between the total number of words in a block and the height of the rendered and printed block in inches [108]; a similar notion is known in Computer Vision, that is the *intensity* of an image region [110]. I transfer this concept to HTML text. The counterpart of a pixel in HTML is *character data* (the atomic text portion), an image region translates to a *sequence* of atomic text portions, which I also call *block* here. To determine a text block's "height", we word-wrap its text (not its rendered representation) at a constant line width $w_{\max}$ (in characters). The resulting block b_x's density $\rho(b_x)$ could then be formulated as follows:

$$\rho(b_x) = \frac{\text{Number of tokens in } b_x}{\text{Number of lines in } b_x} \tag{4.5}$$

This definition of text density has the elegant property that – except tokenization – no lexical or grammatical analysis needs to be performed. Given a proper wrapping width, it is supposed to serve as a discriminator between sentential text (high density) and template text (low density). I propose $w_{\max} = 80$. This is the traditional screen width of monospaced terminals and seems to fit the definitions of an English sentence: Assuming an average word length of 5.1 characters[1], we can write a maximum of $\frac{80}{5.1+1} = 13.1$ separate words (tokens) per line, which roughly covers one medium-sized sentence; obviously, the absolute maximum is 40 one-character tokens per line. It makes sense to exclude the last line of a multi-line block for the computation, since it would falsify the actual density when averaging if it is not completely filled to $w_{\max}$. Given the set of tokens T contained in the set of wrapped lines L covered by a block b_x, we can reformulate Equation 4.5 as follows.

$$T'(b_x) = \{t \mid t \in T(l),\ l_{first}(b_x) \leq l < l_{last}(b_x)\}$$

$$\rho(b_x) \quad = \quad \begin{cases} \frac{|T'(b_x)|}{|L(b_x)|-1} & \left|L(b_{x)}\right| > 1 \\ |\, T(b_x)\, | & \text{otherwise} \end{cases} \tag{4.6}$$

Now the density of a multi-line block is not influenced by the number of additional tokens (i.e., doubling the number of tokens leads to almost double the number of lines, which gets normalized again; see Equation 4.6). However, having only a few words (like "Contact us") still leads to a much lower density value, as expected.

While the text density measure does not consider lexical or grammatical properties of sentences at all, its role as a surrogate for sentence length may be well justified. Altmann [2] supports this by the rationale that *language* itself does actually not care about the existence or clear boundaries of particular lexical or grammatical units and that such units are rather an orthographical convention of the speech community. What seems more important than a proper definition of "sentence" is the measure of the units enclosed by the sentence (words, syllables, characters). The unit used for text density is the *token*, which basically is a variant of the (also diffuse) notion of "word"; in our case it is any contiguous sequence of non-whitespace characters, simplified to the set of contained literals and digits.

[1] An overview of language-specific word lengths can be found at `http://blogamundo.net/lab/wordlengths/`

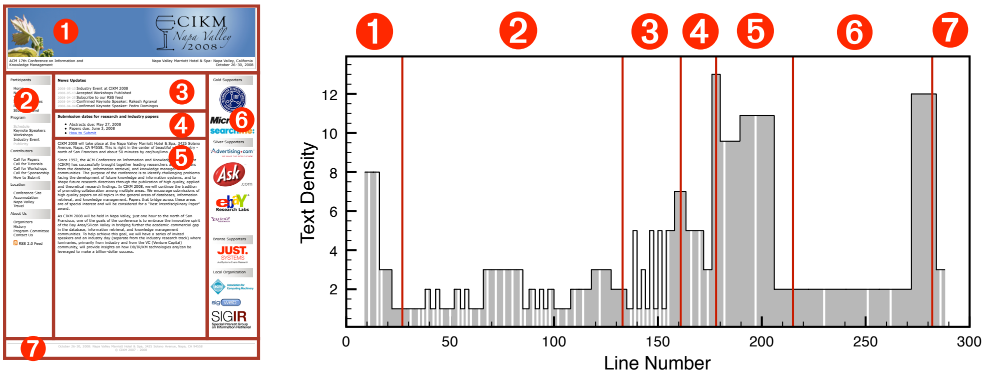


Figure 4.2: Visual vs. Densitometric Segmentation (expected results)

Segmentation as a 1-Dimensional Problem

The task of detecting block-separating gaps on a web page ultimately boils down to finding neighbored text portions with a significant change in the *slope* of the block-by-block text density. In Figure 4.2, we see the desired segmentation[2] of the CIKM 2008 welcome page (`http://cikm2008.org/`), using both visual as well as densitometric boundaries. In the diagram, the density of the atomic text blocks is depicted as grey bars, HTML markup is indicated as white stripes and the expected segmentation boundaries are indicated as red vertical lines. Apparently, apart from the expected spikes, the distribution of text density appears to be a fairly good signal for textual similarity as well as for identifying full-text segments (block #5).

4.2.2 The Block Fusion Algorithm

As it turns out by the preceding discussion of the segmentation problem, we can essentially transfer parts from the perspective of Quantitative Linguistics as well as of Computer Vision to our setting. Due to Altmann's findings about the length dependence of neighbored sentences within the text flow and my corresponding findings

[2] Indisputably, there is no such thing as *the* segmentation, since segments may be considered at different granularities.

on the text density, a greedy strategy seems a plausible algorithmic approach; besides being deterministic, an at least near-optimal result is likely. If we now indeed consider *text density* as being interrelated to the notion of *pixel intensity*, we may consider adopting the Block Growing strategy from image processing to. To avoid confusion with the pixel-based methods, I call this token-based method *Block Fusion*. The decision when to combine (fuse) two adjacent blocks now is made by comparing them with respect to their text densities instead of pixel intensities. We may define this slope delta between two adjacent blocks x and y as:

$$\Delta\rho(x, y) = \frac{|\rho(x) - \rho(y)|}{\max(\rho(x), \rho(y))} \tag{4.7}$$

If the slope delta is below a certain threshold $\vartheta_{\max}$, we assume that the blocks belong to one segment and should therefore be fused. “To fuse” here means joining the lines of the two blocks x and y to a new block z, such that z spans from the first line of x to the last line of y. After this, x and y are replaced by z. As with the Block Growing strategy, we can iteratively continue with this operation until no pair of neighbored block exists which satisfies the threshold constraint.

In addition to that, we might also consider the following extension to this simple fusion strategy. As we can see from the example density distribution of the CIKM web page (Figure 4.2), there are some adjacent segments with alternating densities of 1.0/2.0/1.0, 1.0/5.0/1.0 etc. (this is the section about important dates – the dates are enclosed by `<SPAN>` tags, which create gaps). This may lead to high slope deltas close to 100% and therefore to less fusions than expected. I conclude that the surrounding blocks *dominate* the enclosed one. My suggestion is to smooth these alternations by adding the following condition to the Block Fusion algorithm: if the text densities of the predecessor and successor of a block are identical and higher than its own density, all three blocks are fused. Of course, we will validate this heuristic against the plain strategy. See Algorithm 4.1 for a common representation of both strategies, BF-PLAIN and BF-SMOOTHED.

The computational complexity of Block-Fusion is trivial. Assuming we have N atomic blocks on a page, the cost per iteration is $c \cdot (N - 1)$ comparisons ($c = 1$ for

Algorithm 4.1 The Block Fusion algorithm (*plain/smoothed*)

```
Require: $B \Leftarrow$ The set of (initially atomic) blocks which partition the lines $L$
repeat
    loop $\leftarrow false$
    for all $b_i \in B$ with $i > 1$ do
        if $\rho(b_{i-1}) = \rho(b_{i+1}) \wedge \rho(b_i) < \rho(b_{i-1})$ then
                                                        ▷ Only checked for BF-smoothed
            $b_{i+1} \leftarrow \{l \in L|\ l_{first}(b_{i-1}) \leq l \leq l_{last}(b_{i+1})\}$
            remove $b_{i-1}$
            remove $b_i$
            $i \leftarrow i + 1$                                        ▷ Skip $b_{i+1}$
            loop $\leftarrow true$
        else if $\Delta\rho(b_{i-1}, b_i) \leq \vartheta_{\max}$ then
            $b_i \leftarrow \{l \in L|\ l_{first}(b_{i-1}) \leq l \leq l_{last}(b_i)\}$
            remove $b_{i-1}$
            loop $\leftarrow true$
        end if
    end for
until loop $= false$
```

BF-plain, $c = 2$ for BF-smoothed) and a maximum of $N - 1$ fusions per iteration occur. Because the iteration stops as soon as zero fusions occurred, the worst case that may occur is a single fusion per iteration (convergence is guaranteed). The total number of operations for a maximum of k iterations until convergence therefore is:

$$(N - 1) + (N - 1 - 1) + \cdots + (N - k - 1) = O(N)$$

Two variables may influence the quality of the segmentation: the threshold $\vartheta_{\max}$ and the input blocks B. Regarding $\vartheta_{\max}$ I believe that this threshold is not document-specific but rather depends on the average style of the document class and its inherent quantitative properties. In Section 4.2.4 we determine an appropriate threshold value from a random sample of web documents. According to our definition of the simple block-gap model (Section 4.2.1), B describes the sequence of textual portions of the original HTML document. Whenever one or more opening or closing element tags are encountered, a new block is created, consisting of the plain text that is surrounded by markup; each block's text is initially word-wrapped by w_{max} characters (the wrapping does not change in the course of fusion).

Apart from special HTML tags whose nested character elements do not contribute to the text of the page (like `<SCRIPT>`, `<OPTION>` etc.) and the `<A>` tag, which I regard as a core feature of hypertext markup and therefore do not consider a gap before or after this tag, we do not respect the element tag's semantic meaning or expected visual effect – a `<H1>` tag produces the same type of gap as a `<B>` tag, for example. Intuitively, we could of course claim that `<H1>` does indeed have a stronger impact on segmentation than a `<B>` tag, but this would again lead to heuristic, rule-based or DOM-structural approaches. For the evaluation, we will consider such a rule-based extension of the BF-smoothed algorithm (which I call Bf-rulebased for simplicity) that employs a set of specific *gap-enforcing* and *gap-avoiding* tags ($T_{ForceGap}$ and T_{NoGap}). Given the set of tags $T(x, y)$ between two segments x and y, to support this extension we have to change the slope delta function from $\Delta\rho(x, y)$ to $\Delta\rho'(x, y)$:

$$\Delta\rho'(x,y) = \begin{cases} +\infty & T(x,y) \cap T_{ForceGap} \neq \emptyset \\ -\infty & T(x,y) \subseteq T_{NoGap} \\ \Delta\rho(x,y) & \text{otherwise} \end{cases} \tag{4.8}$$

I consider the following gap-enforcing tags ($T_{ForceGap}$) as a good choice for the rule-based approach: `H1-H6`, `UL`, `DL`, `OL`, `HR`, `TABLE`, `ADDRESS`, `HR`, `IMG`, `SCRIPT`[3] (basically a subset of HTML block-level elements tags). For T_{NoGap}, I consider the following tags: `A`, `B`, `BR`, `EM`, `FONT`, `I`, `S`, `SPAN`, `STRONG`, `SUB`, `SUP`, `U`, `TT` (a subset of HTML inline element tags). When $\vartheta_{\max} = \infty$, this approach simply segments the document after every occurrence of a tag $\in T_{ForceGap}$ regardless of $\Delta\rho'$ or T_{NoGap} (Block Fusion has no effect in this case; I call this special variant JustRules). Lower values of $\vartheta_{\max}$ represent a trade-off between markup-based and density-based segmentation. The examination of the effects of $\vartheta_{\max}$ are part of this evaluation.

[3] Occurrences of `SCRIPT` likely indicate a gap.

4.2.3 Experimental Evaluation

To demonstrate the stability and effectiveness of the density-based Block Fusion strategy, I employed two standard test collections: Webspam UK-2007[4] and the Lyrics dataset used in [25]. Despite its name the Webspam UK-2007 collection is a good snapshot of the U.K. Web, roughly consisting of 106 million pages from 115,000 hosts. Several hosts have already been classified as *spam/non-spam*.

From this non-spam fragment (356,437 pages) I randomly picked 111 web pages coming from 102 different websites and manually assessed these documents to define a comparable segmentation. These manual results were then compared against the following different clustering strategies:

1. WordWrap. Simply take all text of a page and wrap it after $w_{\max} = 80$ characters; every line is a segment.

2. TagGap. Every text portion between any tag (except `A`) is a segment.

3. BF-plain, BF-smoothed and BF-rulebased. As described in Section 4.2.2.

4. JustRules. As described in Section 4.2.2.

5. GCuts. As described in [25]. I did not implement this algorithm. Yet, a comparison of clustering performance scores is justified since both datasets comprise randomly chosen web pages of various kind and are of the same size.

Statistical Properties of Web Page Text

First of all, we need to validate the assumptions on the actual quantitative linguistic properties of textual web page content. One can assume that text density as defined in Equation 4.6 is a surrogate for sentence length. It should therefore also yield the same characteristic distribution (or at least one which satisfies Equation 4.1). To derive distinct classes i from the text density quotient of neighbored blocks, I use the following assignment in accordance with Eq. 4.4.

[4] `http://www.yr-bcn.es/webspam/datasets/uk2007/`

Of note, adjacent blocks with the same density are regarded as one block, i.e. as a contiguous "sentence" which has been mistakenly separated:

$$X[i] = \left\lceil \frac{\Delta\rho(b_{x-1}, b_x)}{\rho(b_x)} \right\rceil \quad \forall \Delta\rho(b_{x-1}, b_x) \neq 0 \tag{4.9}$$

I used the manually created segmentation of the 111 web pages from the Webspam-UK2007 test collection and computed the class frequencies. Then the Altmann-Fitter[5] was applied to automatically determine one or more possible fits out of more than 200 supported discrete distributions for the given input data. The most significantly fitting probability distribution was the *negative hypergeometric* (Equation 4.2 with $K = 2.30454$, $M = 0.10989$, $n = 17$), having $\chi^2 = 14.2394$, $P(\chi^2) = 0.3572$, $C = \chi^2/\sum F(i) = 0.0061$, $d.f. = 13$ and is rated by the Altmann Fitter as a "very good fit". See Figure 4.3 for a graphical comparison; raw results are shown in Figure 4.12.

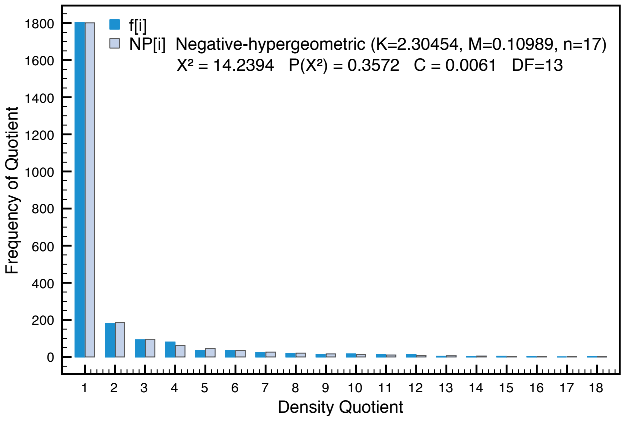


Figure 4.3: Probability Distribution of the Text Density Quotient of Adjacent Blocks

[5] http://www.gabrielaltmann.de/

While this differs from the initially assumed hyperpascal distribution, the general assumption (Equation 4.1) still holds and seems to abide by Frumkina's law. In fact, Vulanovic and Köhler assume [115] that Frumkina's law can be applied not only on term-level but to all types of linguistic units. It can now be shown that this is at least the case for the distribution of text density quotients between adjacent blocks, coming to the conclusion that text density may indeed function as a surrogate for sentence length.

More, we also find that the distribution of the number of tokens in a segment abides by Zipf's law. This has already been shown on document-level [34], and it is just consistent to also find these properties on intra-page level. I was able to model the segment-level word lengths of our manually segmented documents by $y = 1.086 \cdot x^{-0.7028}$, with $\chi^2 = 256.555$ and a root mean square error (RMSE) of 0.013.

4.2.4 Segmentation Accuracy

Metrics. In order to quantify the accuracy of the segmentation computed by Block Fusion, I employed the two cluster correlation metrics *Adjusted Rand Index* (AdjRand) and *Normalized Mutual Information* (NMI) used in [25]. Both metrics determine the agreement between two clustering methods on a particular dataset, using a value between 0 (no agreement) to 1 (perfect agreement). The corresponding label vectors hold the information to which segment a particular token belongs to. AdjRand is Hubert's & Arabie's normalized extension to the Rand measure, which basically relates the number of agreements to the number of disagreements between the two given clusterings [61]. NMI measures the mutual dependence of the two solutions by relating their entropies (see Section 2.3). Knowing that segmentation should follow Zipf's law on token-level, we can also measure and depict the consistency of a particular segmentation solution with this law. A deviation from the expected distribution is regarded a segmentation failure.

Results. Assume that for each variant of Block Fusion there is an optimal setting for the threshold $\vartheta_{\max}$ that is pre-determined by the underlying linguistic regularities. I probed Block Fusion using different settings for this threshold using our sample document set: for each candidate threshold, the average AdjRand and NMI scores were computed, retrieved by a document-level comparison of the segmentations. The results are shown in Figures 4.4 and 4.5; it also shows the average number of resulting blocks for each setting as a reference. For BF-PLAIN and BF-SMOOTHED there seems to be an optimal threshold at $\vartheta_{\max} \approx 0.38$ for our sample document set, whereas any threshold between 0.3 and 0.4 seems reasonable. Starting with $\vartheta_{\max} = 0.4$, the accuracy decreases and finally drops dramatically with $\vartheta_{\max} \geqq 0.6$. See Figures 4.9, 4.10 and 4.11 for the corresponding visual and densitometric representation.[6]

I verified that the determined thresholds $\vartheta_{\max}$ are not particularly document-specific – we get almost the same optimal threshold for two random halves of the test set. For BF-RULEBASED the optimum is $\vartheta_{\max} \approx 0.6$. This means that the heuristically determined gap-enforcing tags do indeed contribute to the quality of segmentation, but the text densities do as well. The results for all applied clustering strategies are depicted in Table 4.1. Block Fusion clearly improves over WORDWRAP and TAGGAP. Interestingly, the scores of BF-PLAIN and BF-SMOOTHED are almost identical to GCUTS [25], which is a surprising achievement for a markup-agnostic approach. At last, BF-RULEBASED in fact outperforms any other approach. While it is close to the quality of JUSTRULES (whose accuracy confirms the effectiveness of the heuristic segmentation rules for the evaluated dataset), it also shows that our heuristics were not perfect and Block Fusion was able to improve them. Finally, I also examined the impact of $w_{\max}$ to the accuracy (see Figure 4.7). It appears that this word-wrap boundary is stable for widths between 80 and 110. This confirms the assumption on the relation between language-specific average sentence length and line width. Theoretically, we could optimize it to $w_{\max} = 90$, but this would only increase accuracy by less than 0.01 on average. Finally, Figure 4.6 shows a log-log plot of block-level tokens counts for all considered algorithms (except GCUTS). All Block Fusion-based approaches as well as JUSTRULES and the manual segmentation expose

[6] The short segments seen in Figure 4.10 could not be fused by BF-Smoothed, because the smoothening criterion $\rho(b_{i-1}) = \rho(b_{i+1})$ was not met. I heuristically found the improved criterion $\rho(b_{i-1}) \leqq 5 \wedge \rho(b_{i+1}) \leqq 5$ which indeed fuses the segments correctly, while improving the accuracy scores only by ca. 0.02. We may therefore consider this improvement as insignificant and omit it from the proposed solution.

the typical straight line known from Zipf distributions. As expected, TagGap and WordWrap obviously do not show this behavior. This means that Block Fusion is indeed able to transform the tag-induced segmentation to a segmentation which resembles the same statistical properties as the expected ones.

Performance

Since Block Fusion is designed as an iterative algorithm, we should consider the iteration behavior in terms of *average accuracy error* $(1 - accuracy)$ – I expect this error to monotonously decrease per iteration, just as the number of remaining blocks. Figure 4.8 reveals that most of the error gets removed already after the first iteration. Even though on average more blocks are fused during the following iterations, these fusions do not contribute to improving accuracy. Block Fusion achieves this performance because it can fuse an arbitrary number of preceding atomic or compound blocks with similar density in one iteration (see Algorithm 4.1). Notably, for the used test data, the total processing time per page was only 15ms on a standard laptop.

	AdjRand	NMI	# Blocks
WordWrap	0.25	0.59	25.0
TagGap	0.43	0.65	69.43
Bf-plain	**0.60**	**0.75**	27.72
Bf-smoothed	**0.62**	**0.76**	19.77
Bf-rulebased	**0.79**	**0.87**	21.24
JustRules	0.78	0.84	17.64
(GCuts)	(0.60)	(0.76)	-

Table 4.1: Achieved average Accuracies

	C	b	χ^2	Error
Bf-plain	1.04024	0.74899	98.15	0.00438
Bf-smoothed	1.03643	0.73334	87.38	0.00538
Bf-rulebased	1.47937	0.67028	649.36	0.02096
JustRules	1.08526	0.70280	256.56	0.01372

Table 4.2: Zipf Distribution Parameters

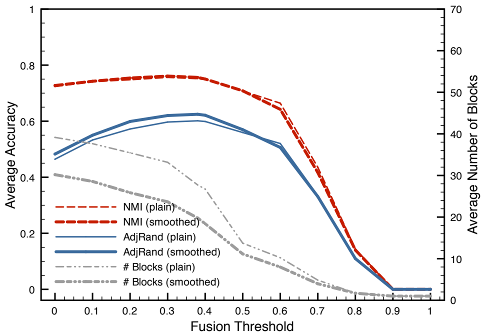


Figure 4.4: Optimizing $\vartheta_{\max}$ (BF-PLAIN/SMOOTHED)

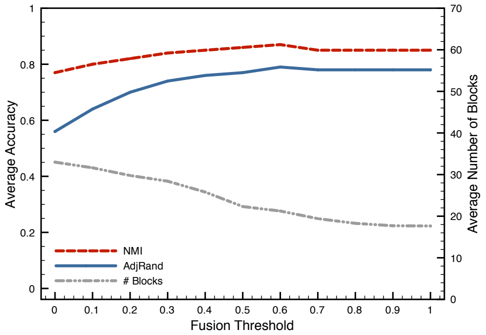


Figure 4.5: Optimizing $\vartheta_{\max}$ (BF-RULEBASED)

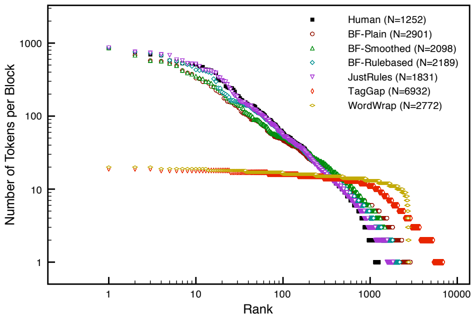


Figure 4.6: Validation of Zipf's Law on Block Level

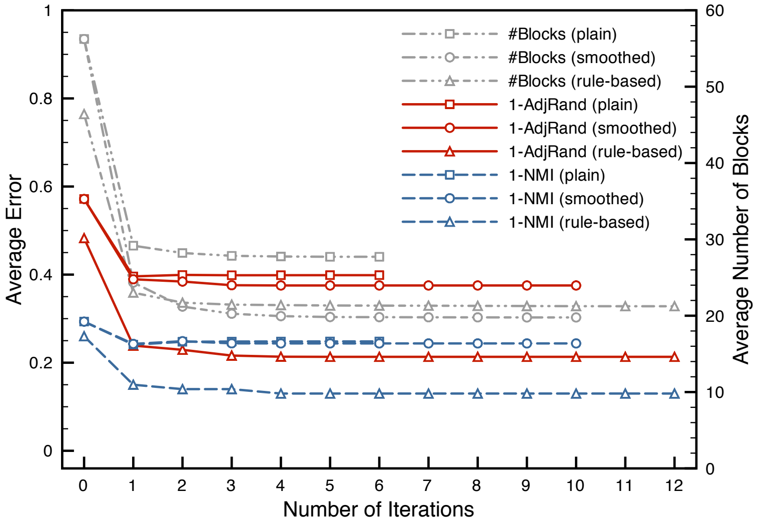


Figure 4.8: Iteration Behavior

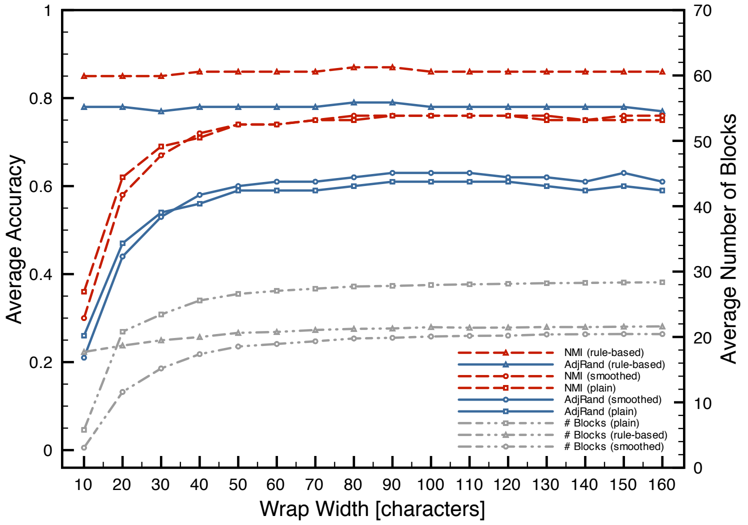


Figure 4.7: Impact of $w_{\max}$ on Average Accuracy

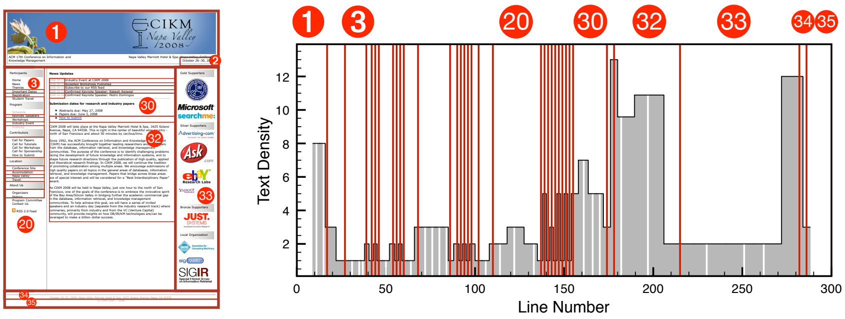
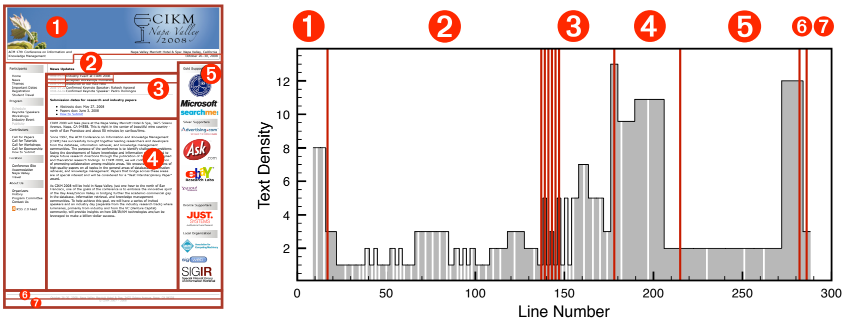


Figure 4.9: Visual vs. Densitometric Segmentation (BF-PLAIN)



See Footnote 6 for an explanation of the short segments between lines 131 and 147.

Figure 4.10: Visual vs. Densitometric Segmentation (BF-SMOOTHED)

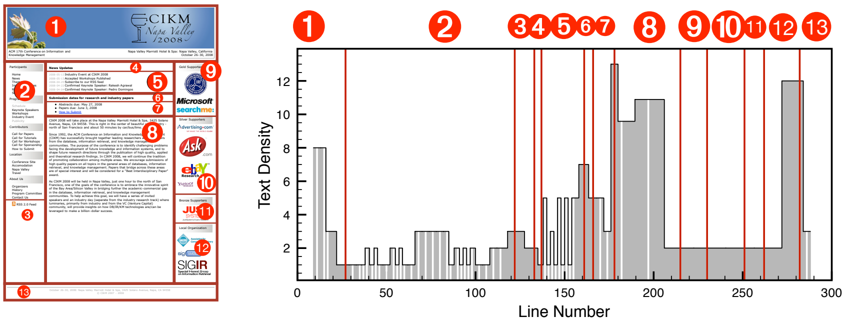


Figure 4.11: Visual vs. Densitometric Segmentation (BF-RULEBASED)

4.2.5 Application to Near-Duplicate Detection

Setup. Finally we now quantify the usefulness of the segmentation for the purpose of near-duplicate detection. Again we compare the results from Block Fusion against [25], where the LYRICS dataset was used to evaluate the accuracy of detecting web pages with the same content but different appearance. The dataset consisted of 2359 web pages song lyrics by six popular artists (ABBA, Beatles, BeeGees, Bon Jovi, Rolling Stones and Madonna), taken from the three websites `absolutelyrics.com`, `seeklyrics.com` and `lyricsondemand.com`. The six artists were deliberately chosen to minimize the effect of false-positives on the evaluation caused by cover songs. As I was unable to acquire the original dataset, I crawled the three websites again using the same setup, resulting in 6982 web pages (which is likely to be a superset of the initial crawl by Chakrabarti et al.). By matching artist and title, 1082 songs have been determined that appear on all three websites (i.e., on 3246 web pages). In addition to that, 3246 other web pages have been randomly chosen from the three websites (1082 for each). This setup allows a relatively clean comparison between the true-positive and true-negative rates of a de-duplication algorithm. To determine what a near-duplicate is and what is not, the same heuristic was used as in [25]: For each page of a pair of candidate pages, the tokens of the largest text segment are used to create 8 shingle fingerprints using the min-hash algorithm, with a window of 6 tokens. A pair of pages is regarded a near-duplicate if the pages share at least 50% of the shingles. The largest text segment simply is determined by counting the number of enclosed tokens; in our setup, segments containing at least 50% hyperlinked textual content are discarded since they are likely not to contain the main content despite their length.

Results. The resulting true positive/negative scores corresponding to each algorithm (including a comparison to the text as a whole, FULLTEXT) are shown in Table 4.3. JUSTRULES is the narrow winner with respect to finding duplicates, but all Block Fusion variants perform equally well for detecting non-duplicates and significantly perform better than GCUTS, even the simplest variant BF-PLAIN.

	True Duplicate Pairs	True Non-Duplicate Pairs
TOTAL	3246	3246
FULLTEXT	19.9%	96.3%
WORDWRAP ($w_{\max} = 80$)	5.4%	76%
TAGGAP	16.9%	88.5%
Bf-plain ($\vartheta_{max} = 0.38$)	72.2%	100%
Bf-smoothed ($\vartheta_{max} = 0.38$)	73.1%	100%
Bf-rulebased ($\vartheta_{max} = 0.6$)	86.3%	100%
JUSTRULES	89.4%	100%
(GCUTS)	(61.7%)	(99.9%)

Table 4.3: Duplicate Detection Accuracy

4.2.6 Discussion

The problem of web page segmentation can be seen from a quantitative linguistic point of view as a problem of identifying significant changes of particular statistical properties within the considered text. As demonstrated, an effective property is *token-level text density*, which can be derived from vision-based measures. This text density follows the same fundamental linguistic law (Frumkina's Law) as many other linguistic units. In addition to that, the distribution of the expected number of tokens in a segment follows Zipf's law. The proposed algorithm for web page segmentation, built upon the region growing strategy known in Computer Vision, performs significantly better than the state-of-the-art graph-theoretic algorithm, as the experimental evaluation on large real-world data sets demonstrates.

The presented approach is orthogonal to existing work and considers new and complementary aspects to solve the segmentation task. As shown by the rule-based Block Fusion hybrid, a more sophisticated combination of other strategies and the Block Fusion algorithm promises further improved segmentation quality. Since the considered linguistic properties seem to be mostly language-independent, the next logical step is to evaluate these findings on a multilingual corpus. In particular, we need to discuss the influence of the wrapping parameter $w_{\max}$ and threshold $\vartheta_{\max}$ on different languages. Further work should also find an explanation of the discovered statistical behavior from a purely linguistic perspective. It would also be particularly interesting to investigate the use of the presented techniques in other areas of Information Retrieval, including block-level ranking, block-level link analysis and block-level classification.

```
-- ALTMANN-FITTER 2.1 --              X[i]   F[i]     NP[i]
Result of fitting                        1   1802  1800.9989
                                         2    180   184.9156
Input data: hist-1.dat                   3     92    95.4882
Distribution: Negative hyper-            4     80    62.2025
geometric (K,M,n)                        5     34    44.5585
                                         6     36    33.5436
Sample size: 2334
Moments:                                 7     24    25.9808
                                         8     18    20.4556
M1 =  1.8106  M2 =  4.4963
                                         9     14    16.2396
M3 = 34.2793  M4 =356.4193              10     16    12.9186
Best method is Method 1 of 2            11     12    10.2396
Parameters:                             12     12     8.0391
K = 2.30453585151999                    13      4     6.2069
M = 0.109889153462268                   14      2     4.6669
n = 17, DF =13                          15      4     3.3651
χ^2 = 14.2394                           16      2     2.2639
                                        17      0     1.3385
P(χ^2) =  0.3572
                                        18      2     0.5779
C =  0.0061
```

Figure 4.12: Altmann-Fitter Results

4.3 A Densitometric Classification of Web Templates

Utilizing the segment-level text density metric presented in the previous section, this section covers an analysis of the structure of a large, representative Web corpus. Through a densitometric classification, we find that Web content exposes two classes of text, covering full-text and navigational information respectively. I show that this structure corroborates recent findings from the field of Quantitative Linguistics. Finally, the findings are applied to template removal.

4.3.1 Theoretical Background

Quantitative Linguistic Text Theory

Several observations have been made which corroborate the theory that natural language obeys the same principles as many other psychobiological and natural phenomena, namely the class of *power laws* [90]. George K. Zipf pioneered this model by his *principle of least effort,* which he said was inherent in human behavior [125]. Numerous empirical observations confirm the hypothesis that the creation process of language, in particular text (spoken or written language), follows particular probabilistic regularities, which have been subsumed by statistical laws, in particular Zipf's law (the frequency of an object, e.g. a term, is inversely proportional to its rank), Frumkina's law (when dividing text into passages of words, the frequency of a particular linguistic entity follows the negative hypergeometric distribution) and the Menzerath-Altmann law (the longer a linguistic construct, the smaller its constituents). The organization of text has been observed and successfully modeled statistical as *urn trials* at the level of various linguistic units such as *phoneme*, *word*, *sentence*, *text segment* etc. and for several features such as *frequency*, *length, repeat rate*, *polysemy* and *polytextuality.*

The levels of language seem to be strongly interdependent (cf. Menzerath-Altmann law). Köhler modeled this system as the so-called synergetic language control circuit and showed that it seems applicable to any linguistic level or aspect [72]. He postulated so-called language system requirements, amongst others the requirements of secure/reliable information transfer, leading to redundancy, and the requirements of economy, incorporating the principle of least effort, with its aspects like minimization of effort for encoding, decoding, memory capabilities/context-independence, ambiguity and so on. Köhler found that the system requirements mutually influence the variability of the system's properties in cooperating and in competing ways; considering Zipf's theories, these requirements may be called synergetic "forces" [71, 72].

Any attempt to corroborate the established laws and models (or possibly to reject them) requires a quantitative, empirical analysis. Quantification is really not the aim, but a means to understanding the structures and processes of text and language [51]. The required statistical analysis has to be performed using an appropriate text or corpus, otherwise one would neglect/hide the language-immanent heterogeneity [3]. Once a representative baseline corpus is established, further analytical explorations can be attempted such as stylometric approaches to assign (with a given probability) a particular author to a specific document (or to exclude an author from consideration), a genre (newspaper text, political statement, scientific work etc.) or a readability score (e.g., boulevard news vs. legal articles) to a particular article, using scores like type-token-ratio, verb-adjective-ratio, vocabulary richness and so on [112, 113, 95].

If one is able to closely fit a previously discovered distribution (e.g., negative hypergeometric, hyperpascal, negative binomial etc.) to the data, this contributes to corroborating the theory. Recently, Wimmer and Altmann presented a unified representation of many existing linguistic hypotheses [118, 51], a logical extension of Köhler's synergetic approach. They derive a common representation of the aforementioned distributions and relations by discussing the relation between a linguistic variable Y and another independent variable X which shapes the behavior of Y (i.e., also its rate of change, *dx*, which in turn is controlled by the aforementioned synergetic forces). Relations between X and Y for example are: polytextuality/polysemy, polysemy/length and also rank/frequency.

The relationship between X and Y can be seen as an infinite series of the form

$$\frac{dy}{y} = (a_0 + \frac{a_1}{x} + \frac{a_2}{x^2} + \frac{a_3}{x^3} + \cdots)\, dx \tag{4.10}$$

(with $a_0, a_1, a_2, \ldots$ being constant factors of the acting forces). The solution of 4.10 yields

$$y = C\, x^{a_1} e^{-a_0 x} exp(-\sum_{i=1}^{\infty} \frac{a_{i+1}}{x^i}) + d \tag{4.11}$$

(with C and d being normalization parameters) which actually is a generalization of the commonly used form of the Menzerath-Altmann law

$$y = C\, x^{a_1} + d \tag{4.12}$$

This regularity was also discussed for discrete variables, in particular non-negative probability distributions with probability mass functions $\{P_0, P_1, \dots\}$ of the form $P_x = g(x)\, P_{x-1}$. The discrete equivalent of the continuous model (Equation 4.10) is:

$$P_x = (a_0 + \frac{a_1}{x} + \frac{a_2}{x^2} + \frac{a_3}{x^3} + \cdots)\ P_{x-1} \tag{4.13}$$

From this recurrence formula many well-known distributions observed in the field of linguistics can be derived, including the Katz/Kemp-Dacey-hypergeometric families of distributions [117], whose limiting cases are (amongst others) the geometric, the Poisson, the hyperpascal and the negative-hypergeometric (including its limiting cases binomial and negative-binomial) distributions; all of them have already been discussed and empirically found for particular linguistic units.

Relation to the Web

It would be surprising if the findings made on "plain text" would not be valid for text on the Web. I have shown in Section 4.2 that the discussed laws can be applied successfully to segment web pages into blocks of text. For conducting the segmentation, the block-level text density measure $\varrho(b)$ was introduced, derived from the pixel-based text density of Computer Vision-based approaches and transformed to token-level. Basically, it counts the number of tokens $|T(b)|$ in a particular text block b divided by the number of lines $|L(b)|$ covered after word-wrapping the text at a fixed column width $w_{\max}$ (the empirically estimated optimal value for English text is between 80 and 90 characters). Due to the side-effect of having an incompletely filled last line after wrapping, the latter is not taken into consideration unless it is the only line in the segment:

$$T'(b) = \{t \,|\, t \in T(l),\ l_{first}(b) \le l < l_{last}(b)\}$$

$$\varrho(b) \quad = \quad \begin{cases} \frac{|T'(b)|}{|L(b)|-1} & |L(b)| > 1 \\ |\,T(b)\,| & \text{otherwise} \end{cases} \tag{4.14}$$

The actual segmentation algorithm is based on the merge-only strategy *Block Fusion* presented in Section 4.2.2. Adjacent text fragments of similar text density (interpreted as "similar class") are iteratively fused until the blocks' densities (and therefore the text classes) are distinctive enough. Using various settings, including a rule-based approach, it was shown that the resulting block structure closely resembles a manual segmentation.

Even though text density was derived from concepts of Computer-Vision, it appears that the exposed behavior of $\varrho(b)$ in text is similar to existing linguistic measures. In particular, the ratio between the text densities of neighbored blocks follows the negative-hypergeometric distribution, corroborating Frumkina's law (see Section 4.2). Further details about the density measure are discussed in Section 4.3.2.

4.3.2 Corpus-Level Pattern Analysis

Setup

We again conduct our analysis on the Webspam UK-2007 test collection[7] (as for the segmentation problem in Section 4.2). Since spam pages tend to be automatically generated, may not necessarily obey the laws of natural language and could skew our results, we also again focus on the non-spam part consisting of 356,437 pages, with 316,448 documents containing extractable text.

Since we are trying to understand the distinction between templates and main content, we perform a statistical classification on segment-level, under the assumption that each segment is sufficiently homogeneous (i.e., either template *or* main content). As a manual segmentation appears infeasible at corpus-scale, we employ the BlockFusion segmentation algorithm, *BF-RuleBased* in particular, which was shown to have a segmentation accuracy in terms of normalized mutual information (NMI) of 0.87 (Table 4.1); BF-RuleBased is the most effective variant of the BlockFusion family – the segmentation boundaries are usually at the HTML block-level elements `H1-H6`, `UL`, `DL`, `OL`, `HR`, `TABLE`, `ADDRESS`, `HR`, `IMG`, `SCRIPT` but they may also exist between two neighbored segments which expose noticeably different text densities.

[7] `http://www.yr-bcn.es/webspam/datasets/uk2007/`

Density vs. Token Length

Text density is a particularly useful measure when analyzing the Web's quantitative structure. It does not depend on the notion of "sentence", which we could hardly define for the Web's content: many portions of text simply do not contain sentences, nor anything meaningful which could be separable by full stop (this is especially true for template text). As for the text density a relationship to existing linguistic measures was already shown above, we may assume that $\varrho(b)$ indeed is an adequate linguistic measure, too. Under the aspect that many linguistic measures obey the principles expressed by the Menzerath-Altmann law, we verify whether this law also holds for the text density. Actually, text density seems to follow this law *per definitionem:* the higher a text density, the shorter contained tokens *must* be on average. Thus, a strong relationship to average token length is likely.

First, we analyze the measures "average token length" and "text density" separately. For both measures, we compute per-document averages, normalize the scores to a maximum of 1.0 and sort them in decreasing order. Finally, we fit Equation 4.12 to them. We quantify the goodness of the fit by the correlation coefficient R^2 (the square of the correlation between the response values and the predicted response values) and the root-mean-square error RMSE.

Indeed, for both measures, *average token length* and *text density*, a high correlation can be observed. For average token length, we achieve $R^2 = 0.9335$; RMSE = 0.00002 with $a_1 = 0.51, c = 4.096 \cdot 10^{-5}$, $d = -1$. For the text density, with $a_1 = 7.5 \cdot 10^{-7}$, $c = 2.79 \cdot 10^4$, $d = -1$ the goodness of fit is $R^2 = 0.9654; \text{RMSE} = 0.0154$. Of note, in order to fit the rank sequence for average token length, the top 100 documents which had a very high average score had to be omitted; the skewness was caused by very long tokens (the largest average token length encountered was 65.026). Second, we analyze the ratio of text density to average token length. As above, we normalize and sort the values; we then fit Equation 4.11 to them. With the parameters $a_0 = 0$, $a_1 = 0.024$, $a_2 \cdots a_\infty = 0$, $c = 0.712$, $d = -0.98$, the resulting goodness of fit is $R^2 = 0.97, \text{RMSE} = 0.0029$. The three rank sequences are depicted in Figure 4.13.

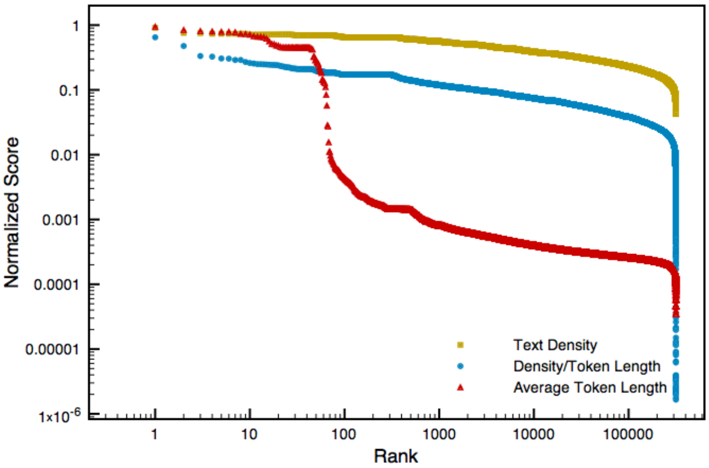


Figure 4.13: Text Density / Token Length Ranks

I conclude that the text density measure (in combination with a good segmentation strategy) can be well integrated into the established theories. Moreover, it appears to be less susceptible for noisy data than the average token length.

The Beta Distribution Model

To reduce the impact of errors caused by a too fine-grained segmentation, we examine the amount of text (= number of tokens) contained in segments of a particular text density ϱ. We can model this histographically by rounding the density to the nearest integer $\varrho'(b) = [\varrho(b)]$ (according to [118] switching between a continuous and a discrete representation as needs arise is valid under these circumstances, also see Equations 4.10 and 4.13).

Figure 4.14 depicts the retrieved token-level count/density distribution for the whole corpus. Apparently, two modal scores are visible, at $\varrho' = 2$ and $\varrho' = 12$ respectively. This indicates at least two classes of text within the corpus. The superimposition of different classes ("strata") of text is already known in linguistics, from a theoretical perspective it may even be the normal case, even though empirically a separation may not seem necessary [3]. To confirm the presence of multiple classes we need to find a corresponding distribution function. As we have to visible modal scores, the distribution function which is to be retrieved is expected to be a combination of two individual distributions, for which we may chose the Beta distribution:

$$\begin{aligned} f_{\text{beta}}(x, a, b) &= \frac{1}{B(a, b)} x^{a-1}(1-x)^{b-1} \\ B(a, b) &= \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)} = \int_0^1 t^{a-1}(1-t)^{b-1}\, dt \\ \Gamma(x) &= \int_0^\infty t^{x-1}\mathrm{e}^{-t}\mathrm{d}t = x \cdot \Gamma(x-1) = (x-1)! \end{aligned} \tag{4.15}$$

The Beta has several advantageous features. First, the Beta distribution is very generic in the sense that it allows a parameterization of the curve's skewness both to the left ($a < b$) and to the right ($a > b$), having the mode at $(a-1) \cdot (a+b-2)^{-1} \cdot x_{\text{norm}}$. These parameters seem necessary in our case (see the varying token counts for $\varrho' = [1; 3]$ and $\varrho' = [10; 12]$ respectively). Second, the distribution is continuous, which allows us to describe the probability of tokens covered by a particular density ϱ, not only the approximation ϱ'. Third, it is already known in Quantitative Linguistics. Generally, the Beta distribution seems to fit very well to linguistic data. For example, we could fit it to the distribution of English sentence lengths in the standard Brown Corpus[8] with $R^2 = 0.996$, RMSE $= 8.87 \cdot 10^{-4}$ having $a = 1.926, b = 6.356, c = 0.013, x_{\text{norm}} = 79$. In particular, Altmann and Burdinski [5] have derived the discrete negative hypergeometric (or: Beta-binomial) distribution using it, which in turn abides by the Menzerath-Altmann Law. Of note, f_{beta} has also been applied to histogram-based image segmentation [1], which is a remarkable fact because the text density metric and the accompanied BlockFusion algorithm inherit the notion of density as well as the block-merge strategy from Computer Vision, too (Section 4.2.1).

Using two Beta distributions, we may now attempt a fit as follows:

$$\begin{gathered} c \cdot [p_1 \cdot f_{beta}(x, a_1, b_1) + p_2 \cdot f_{beta}(x, a_2, b_2)] \\ x \in [0; 1], \;\; x = \varrho \cdot \frac{1}{2} w_{max} \end{gathered} \tag{4.16}$$

[8] http://people.scs.fsu.edu/~burkardt/m_src/prob/english_sentence_length_pdf.m

Unfortunately the fit was unsatisfactory. I was able to fit a *curve* to Equation 4.16 ($p_1 = 0.65$; $p_2 = 0.55$) but no *distribution* (i.e., with $p_1 + p_2 = 1$). A combination of *three* Beta distribution yields a fairly good distribution fit ($p_1, p_2, p_3 = \frac{1}{3}$; $a_1 = 64.08, b_1 = 147.9$; $a_2 = 2.596, b_2 = 32.33$; $a_3 = 10.7, b_3 = 30.45$; $c = 0.025$ with $R^2 = 0.944, RMSE = 0.0031$):

$$f(x) \quad = \quad c \;\cdot\; [p_1 \cdot f_{beta}(x, a_1, b_1) + p_2 \cdot f_{beta}(x, a_2, b_2) + p_3 \cdot f_{beta}(x, a_3, b_3)] \quad (4.17)$$

However, we can (and therefore must) further simply the distribution to a combination of two beta distributions and the normal distribution, with which we achieve an almost perfect fit ($R^2_{.} = 0.998$, RMSE $= 0.0021$ for $a_1 = 68.03$, $b_1 = 132.5$; $a_2 = 4,034$,$b_2 = 54,49$; $c = 0.015$, $d = 0.64$; $e = 78.87$, $f = 7.834$, $\mu = 28.65$, $\sigma^2 = 6.489$; x scores (densities) have been normalized by $x_{\text{norm}} = 36$ to $[0:1]$ before fitting):

$$\begin{aligned} f(x) \;=\; & c \cdot (d \cdot f_{beta}(x, a_1, b_1) + (1-d) \cdot f_{beta}(x, a_2, b_2)) \\ & + (1-c) \cdot \varphi_{\mu,\sigma^2}(e \cdot x + f) \quad (4.18) \end{aligned}$$

The parameters a_1, a_2, b_1, b_2 define the skewness and the location of the mode of the Beta distribution. c and d are weights, e and f are normalizing constants.

I conclude that the distribution of text densities can be divided into two *fuzzy* classes C_1 and C_2; the transition from C_1 to C_2 follows the normal distribution, which means that for blocks with particular densities it is rather undetermined to which class the contained text belongs. Moreover, from the distribution parameter d I conclude that C_1 roughly covers one third of the tokens enclosed in the corpus and C_2 covers two thirds. Figure 4.14 depicts this fit as well as its three parts; we see that for $5 \leq \varrho' \leq 10$ the normal distribution dominates. Notably, these classes are not visible at token level (see Figure 4.15); the average token length appears to follow the (unimodal) Beta distribution $y = c \cdot f_{\text{beta}}(x/x_{norm}, a, b)$ with $a = 40.53, b = 2612, c = 0.002458$, $x_{\text{norm}} = 358$ ($R^2 = 0.9876, \text{RMSE} = 0.003$). This supports the assumption that text density and average token length are measures at different linguistic levels.

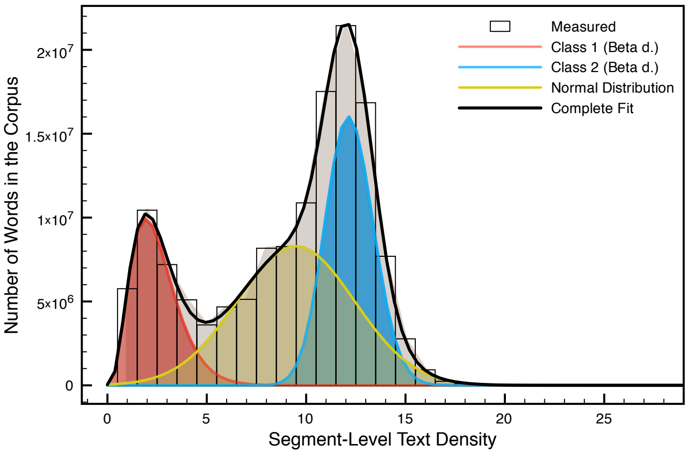


Figure 4.14: Density Distribution Model

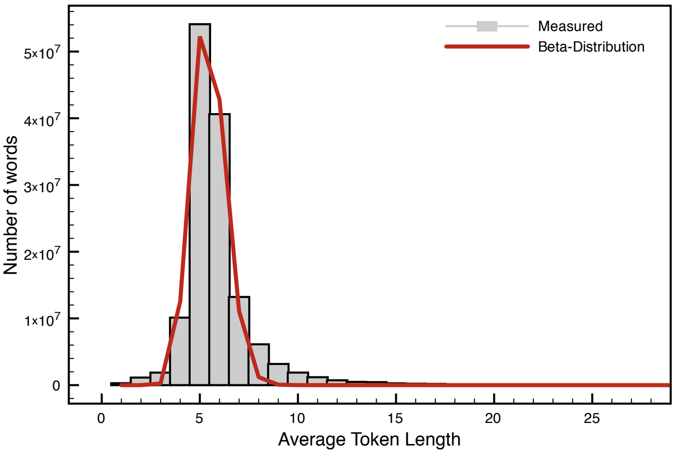


Figure 4.15: Average Token Length

Term Typicality

To make a statement on the meaning of the determined two classes, the content of these classes, that is the *term vocabulary*, needs to be analyzed. If the two classes are different, then the contained token vocabulary should also expose noticeable differences. To prove this, we first divide the corpus text into two partitions π_1 and π_2; π_1 only contains blocks with densities $\varrho' \leq 8$ and π_2 with $\varrho' \geq 9$ ($\varrho' = 8$ is the boundary point of the two discrete beta distributions). Second, we analyze the token distributions of the two partitions.

As we want to express the peculiarities of the two classes C_1 and C_2, which are roughly represented by π_1 and π_2, we compare the partition-specific term document frequencies. We expect that terms which are *typical* for C_1 appear much more often in π_1 than in π_2, and vice versa. We examine this relationship by computing the corresponding document frequency ratio; the normalized ratio follows a power law distribution of the form $y = c\,(x/(1-x))^{-a_1}$ with $a_1 = 0.39$ and $c = 0.01$ ($R^2 = 0.9468, \mathrm{RMSE} = 0.0034$, see Figure 4.16).

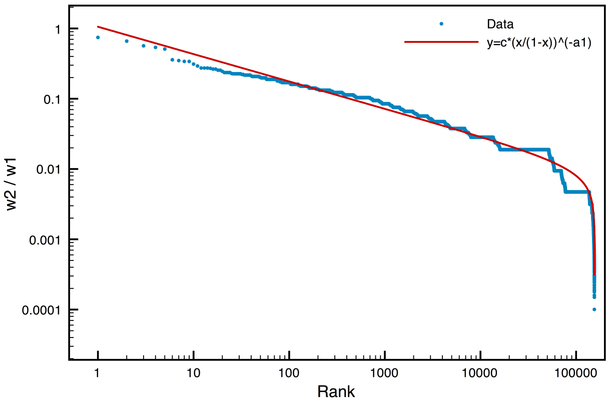


Figure 4.16: Document Frequency Ratio

This type of power law distribution has recently been discussed by Lavalette [81] and Popescu [94] as a generalization of Zipf's law. In our case, we can interpret the ratio $x/(1-x)$ as the combination of two Zipfian subsets, a top-ranked and a bottom-ranked one, which mutually influence the curve (i.e., since our document set is finite as much smaller than the imaginary full set, the observed frequencies drop faster than in the optimal case, again exponentially to be precise). In fact the document frequencies of the considered terms apparently are Zipfian, too, and for both partitions enough typical terms exist.

To avoid over-interpreting the impact of rarely occurring terms, we limit our analysis to terms with a collection-wide document frequency $w_{1\cup 2}$ of at least 100. For these terms, we compute the *term typicality* $\varepsilon(t)$, which we define as the logarithmic ratio of the corresponding document frequencies w_1, w_2 of the examined term t in the two partitions. The ratio is normalized by the logarithm to base $N+1$ with N being the number of documents in the corpus (i.e., the maximum document frequency):

$$\varepsilon(t) = \log_{N+1} \frac{w_2(t)+1}{w_1(t)+1} \tag{4.19}$$

The resulting values are in the range of $[-1;+1]$. The absolute score is the degree of typicality, the sign indicates the direction of typicality (-1 means the term clearly belongs to class 1, $+1$ states that the term clearly belongs to class 2). In our setup, of the 2938 terms with $w_{1\cup 2} \geq 100$, 589 terms (20%) expose a term typicality $\varepsilon \leq -0.05$ (i.e., C_1) and 1255 terms (42.7%) a term typicality of $\varepsilon \geq +0.05$ (i.e., C_2). Table 4.4 shows the top-20 typical terms for C_1 and C_2 respectively. As one can see, C_1 terms are very likely to appear in template blocks, whereas C_2 terms are more likely for full-text.

The "Full Stop" Criterion

I argued that template text usually contained no full stop. This clearly is an observation which needs to be empirically analyzed for the whole corpus. A strong

Rank	Term	ε	Term	ε	Rank	Term	ε	Term	ε
1	memberlist	-0.37	option	0.32	11	videophone	-0.30	pension	0.23
2	usergroups	-0.34	van	0.31	12	stocked	-0.30	creed	0.22
3	headcovers	-0.33	liability	0.29	13	brvbar	-0.30	their	0.22
4	accesskey	-0.33	rd	0.29	14	landscaper	-0.29	these	0.22
5	changelog	-0.33	gloucester	0.28	15	prater	-0.29	adverse	0.21
6	thimbles	-0.31	income	0.28	16	upchurch	-0.29	director	0.21
7	notifications	-0.30	provider	0.27	17	sge	-0.29	michael	0.21
8	tuskers	-0.30	tea	0.26	18	barebone	-0.29	sku	0.21
9	gnomes	-0.30	settings	0.25	19	dr.who	-0.29	double	0.21
10	qed	-0.30	cheap	0.24	20	turntables	-0.29	accident	0.20

Table 4.4: The top-20 typical terms in segments with $\varrho' \leq 5$ (comparison between segment frequency ≥ 100 and < 100)

Rank	Term	ε	Term	ε	Rank	Term	ε	Term	ε
1	sitemap	-0.33	spelled	0.51	11	faq	-0.26	helped	0.31
2	bookmark	-0.29	thousands	0.36	12	miscellaneous	-0.26	majority	0.30
3	accessibility	-0.29	temporarily	0.35	13	jun	-0.26	reached	0.30
4	misc	-0.29	gave	0.34	14	basket	-0.26	despite	0.30
5	skip	-0.28	tried	0.33	15	gmt	-0.26	incorrectly	0.30
6	shipping	-0.28	aimed	0.33	16	wed	-0.26	hundreds	0.30
7	polls	-0.28	seem	0.32	17	faqs	-0.25	themselves	0.30
8	affiliates	-0.27	eventually	0.31	18	currency	-0.25	although	0.30
9	username	-0.27	unfortunately	0.31	19	homepage	-0.24	whether	0.29
10	thu	-0.27	obvious	0.31	20	checkout	-0.24	we'll	0.29

Table 4.5: The top-20 terms for π_1 and π_2

correlation between the feature “segment contains full stop” and the class relationship would support this assumption.

Let us proceed as follows. We partition the corpus' text into segments which contain at least one full stop (π_F) and those without (π_N). I simply define “full stop” as a dot character (`.`) which immediately follows a white-space terminated sequence of at least two letters, except for a few known abbreviations (`vs.`, `DC.`, `Inc.`, `Ltd.`, `No.`, `VAT.` and `Jan.` to `Dec.`). First, we analyze the histographical distribution of tokens enclosed by segments of particular text densities (as in Section 4.3.2), for both partitions π_F and π_N separately and compare to the overall distribution. The histograms are depicted in Figure 4.17. Even though segments with and without full stop exist for all present text densities, the two strata are clearly visible. Again, I was able to fit the Beta distribution $y = c \cdot f_{\text{beta}}(x, a, b)$ to each of the two partitions with high correlation. For the non-full stop part, we get $R^2 = 0.91, \text{RMSE} = 0.015$ with $a = 1,394$, $b = 10,75$, $c = 0.02749$. For the full stop part, we get $R^2 = 0.94, \text{RMSE} = 0.014$ with $a = 30.15, b = 61.54, c = 0.024$. The R^2 values are not as good as for the overall fit (see Figure 4.14) as we ignore the impact of the inter-class normal distribution in this case. Of note, the modes of the two Beta distributions (1.4 and 11.7) are almost identical to the ones found for the classes C_1 (1.99) and C_2 (12.15).

Using Weka[9], I computed the expected KL-Divergence provided by the "full-stop" feature at segment-level. Indeed, text density has a fairly high information gain for predicting the occurrence of a full stop (0.711), which is substantiated by a classification accuracy of 91.4% using a simple linear classifier.

Finally, the amount of tokens enclosed in π_N ($4.18 \cdot 10^7$ or 69%) and in π_F ($9.50 \cdot 10^7$ or 31%) is in line with the ratio between C_1 and C_2 determined by the Beta distribution fit described in Section 4.3.2.

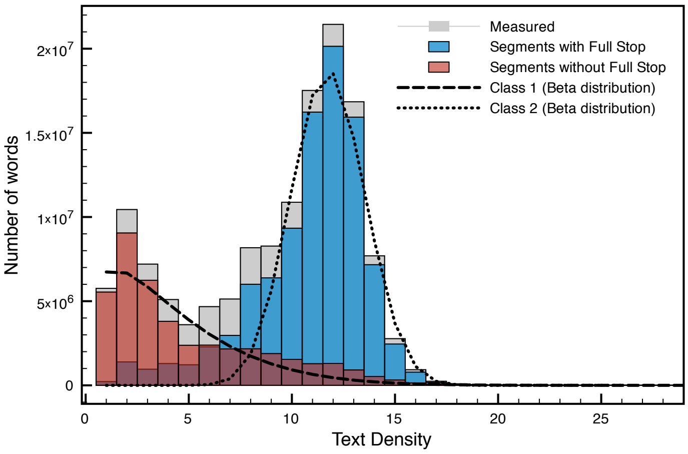


Figure 4.17: Full Stop as a simple partitioning criterion

[9] http://www.cs.waikato.ac.nz/ml/weka/

4.3.3 Template Removal

We now investigate the correlation of the discovered properties of Web text to a baseline strategy for template detection.

Baseline

As shown in [47], the frequency of a segment hash in the collection is a good measure for detecting templates, especially those which are regarded static "boilerplate" text; hashing is not very effective for rarely occurring template segments and for those which contain dynamic, context-sensitive (e.g., time and date) or random text.

For each text segment in our corpus, we normalize the text blocks and create a hash fingerprint as follows: First, the text is converted into lowercase. We remove the month and week day tokens (January-December, Jan-Dec, Monday-Sunday, Mon-Sun) as well as AM and PM and any URL found in plaintext. Then any character except Latin letters are replaced by whitespace, which we normalize to a single space between any remaining token. If the original text contained at least one date token, we add `$DATE$` as a special indicator token. If the original text contained at least one URL, we add `$URL$`; the two indicators are meant to help avoiding hash conflicts of actually different strings. Finally, we compute the SHA1 digest for the remaining text, whose hexadecimal representation is the text's fingerprint.

Evaluation

The above process results in 1,717,039 distinct fingerprints. Interestingly, the sequence of segment frequencies seems to follow Lavalette's extension to the Zipf law (see Section 4.3.2) of slope $a_1 = 0.7408$ with $R^2 = 0.9329$, RMSE= 14.2665 (unnormalized). The curve is depicted in Figure 4.18. This may again be due to the fact that basically two classes of text exist in the corpus, those which are very frequent (boilerplate templates) and those which are not frequent (non-boilerplate content).

Next, we investigate the token-level distribution of frequent templates. We consider segments which have a fingerprint frequency of at least 10. 38,634 of such segments exist, representing 28% of the tokens in the corpus ($3.8 \cdot 10^7$ out of $1.37 \cdot 10^8$), see Table 4.6 for the most frequent ones. The corresponding token distribution again can be fitted to a combination of two Beta distributions and the normal distribution (Equation 4.18) with $R^2_. = 0.9966$, RMSE $= 0.0026$ having $a_1 = 106.6$, $b_1 = 162.6$; $a_2 = 5.348, b_2 = 64.35$; $c = 0.0204$, $d = 0.4821$; $e = 81.69$, $f = 2.045$, $\mu = 23.81$, $\sigma^2 = 9.264$; x scores (densities) have been normalized by $x_{\text{norm}} = 31$ to $[0 : 1]$ before fitting). From d we see that the two Beta-distributed parts are almost equally important to the distribution (48.2% vs. 51.8%), this also correlates with the ratio between templates with full-stop $\pi_{F,T}$ and without $\pi_{N,T}$ (47% to 53%); see Figure 4.19.

As we can see, the relative amount of detected template content in class C_1 is much higher than in class C_2. $\pi_{N,T}$ represents 63% of the tokens covered by segments with $\varrho'(b) \leq 5$, whereas $\pi_{F,T}$ only represents ca. 17% of the tokens for $\varrho(b) \geq 6$ and 20% for $\varrho(b) \geq 9$. An analysis of the remaining segments with $\varrho'(b) \leq 5$ which are not covered by $\pi_{N,T}$ using the term-typicality measure (Table 4.5) and a random-sample manual inspection (Table 4.7) indicates that no significant structural difference exists between the *detected* boilerplate templates and the remainder of text with a text density of 5 or less. Some segments appear to be part of a headline or closing words of a letter, which may indicate a sub-optimal segmentation caused by the BlockFusion algorithm (with a segmentation accuracy of 80% this was expectable). I conclude that the part of C_1 with $\varrho'(b) \leq 5$ represents template text with a high probability and could simply be removed right after the segmentation without requiring a global (fingerprint frequency-based) strategy. The removed content represents 23% of all tokens in the corpus and 84% of the tokens detected by the baseline strategy.

4.3.4 Discussion

Stratification of Web text

Web text exposes two prominent classes (strata) of content; the stratification can be seen best at segment-level when analyzing the ratio between text density and token count. Each class can be modeled using the Beta distribution with a fuzzy transition between them following the normal distribution. The proportions of the two classes (in tokens) roughly is 1 : 2. This classification is found when inspecting the text's densities (or possibly sentence lengths), not when comparing lower levels of text (e.g. token lengths).

As we have found, the textual contents of the two classes significantly differ from each other, in notation (sentences vs. non-sentential text) as well as in terminology. With regard to the linguistic model, we can interpret the class with a low text density average (C_1) as a class that is of *navigational* nature (i.e., allowing a quick, economic perception of provided or related content), whereas the class with a high text density average (C_2) describes content of *descriptive* nature (i.e., supplying the reader with the subject matter's details at the cost of higher syntactic complexity).

Application to Template Removal

47% of the tokens which were classified as template content by the baseline strategy are covered by segments with a text density of $\varrho'(b) \leq 5$. The partition represents 23% of the tokens in the corpus and 84% of tokens detected by the baseline. The obvious strategy to obtain a cleaner text collection is to segment each document using the BlockFusion algorithm and to remove all segments which have a maximum text density of 5. As opposed to the fingerprint baseline strategy, no site-level or global information is necessary for this pruning operation.

Next Steps

Having found a well-grounded model for template content, it would now be interesting to see whether machine learning techniques could further improve the classification task, especially by adding more features for this classification, for example the number of links in a segment. Of course, we also need to measure the impact of our template removal strategy to search (in terms of Precision and Recall). Finally, we should investigate how well the model matches to machine-generated spam content.

I will discuss the first two directions in Section 4.4; the spam detection problem is regarded future work.

Frequency	Segment	Frequency	Segment	Frequency	Segment
33,349	Home	4,231	What's New?	2,943	site search
27,841	Search	4,204	Skip navigation	2,941	Quantity
14,897	Contact Us	4,190	Events	2,933	Introduction
10,101	Links	4,131	Features	2,903	Not Found
9,747	Back	4,073	Publications	2,892	Quick Search
8,517	News	4,045	The document has moved here.	2,874	Sitemap
7,937	About Us	3,949	Tell A Friend	2,791	Services
6,800	Information	3,942	Main Menu	2,771	Accessories
6,696	Site Map	3,874	Products	2,760	You are not logged in.
6,573	Login	3,669	Price:	2,717	Shopping Cart
6,433	Categories	3,560	Terms & Conditions	2,713	About
5,400	Contact	3,517	FAQ	2,634	Print this page
4,814	Advanced Search	3,465	This object may be found here.	2,590	profile
4,775	Help	3,417	Checkout	2,586	Jump to:
4,762	Object Moved	3,344	Newsletter	2,562	Accessibility
4,760	Log In	3,308	Privacy Policy	2,551	Description
4,736	Back to top	3,231	Skip to content	2,509	Please try the following:
4,366	top	3,190	home page	2,500	Technical Information (for support personnel)
4,360	Register	3,132	Reviews	2,496	Quick Find
4,237	Latest News	3,041	Navigation	2,479	Contact Details

Table 4.6: The most frequent Segments

Frequency	Segment	Frequency	Segment
2	in Abbeyview, Dunfermline.	1	Electricians in Tyne & Wear
1	What synthetic methods are used?	1	media assistance mapping
1	How do you read Braille?	3	Home > IPOD / MP3 > Other [...]
2	'B' Team photo	1	Ford Mustang 67
1	Carlisle The Border City Our Price : £ 2.99 more...	2	Summer Barbecues
1	Brian Stone 11 May 05 Wheatear Bakewell [...]	1	cheaptickets airline cheapticket
1	New Congregational Chapel Independent Chapel [...]	1	Cheers, Kevan.

Table 4.7: Rare Segments with $\rho'(b) \leq 5$

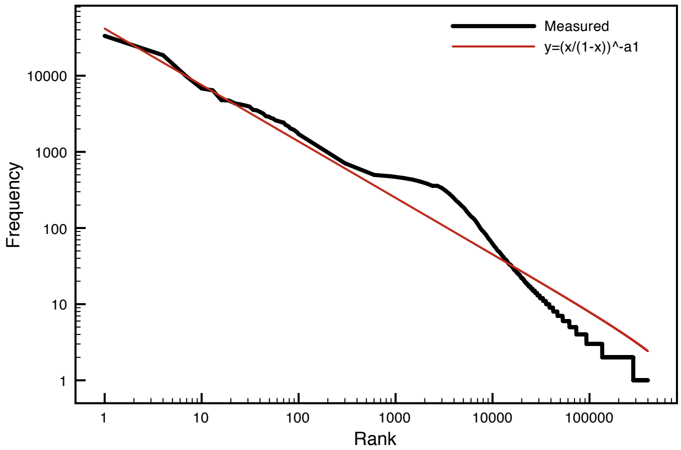


Figure 4.18: Segment Frequency

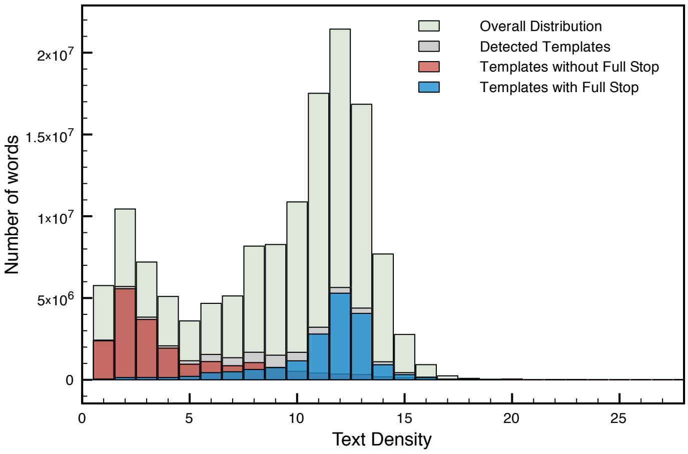


Figure 4.19: Templates detected by Fingerprinting

4.4 Boilerplate Detection using Shallow Text Features

In this section, I report on an analysis of the most popular features used for boilerplate detection on two corpora. I show that a combination of just two features - *number of words* and *link density* - leads to a simple classification model that achieves competitive accuracy. The features have a strong correspondence to stochastic text models introduced in the field of Quantitative Linguistics. Moreover, I show that removing boilerplate content based on these features significantly improves precision on the BLOGS06 benchmark, at almost no cost. Finally, I give a statistical linguistic interpretation of the observations made.

4.4.1 Related Work

Boilerplate and template detection are strongly related to the more generic problem of web page segmentation (see Section 4.2). Approaches to boilerplate detection typically exploit DOM-level features of segments by means of handcrafted rules or trained classifiers, or they identify common, i.e., frequently used segments or patterns/shingles on a website [10, 11, 24, 28, 33, 47, 114, 123]. Using a combination of approaches, Gibson et al. quantify the amount of template content in the Web (40%-50%) [47].

The CleanEval competition [12] aims at establishing a representative corpus with a gold standard in order to provide a transparent and comparable platform for boilerplate removal experiments. The evaluated algorithms mainly apply machine learning techniques for the classification [12]. For instance, NCleaner [41] utilizes a trained n-gram based language model, and Victor [109] employs a multi-feature sequence-labeling approach based on Conditional Random Fields, similar to the approach of Gibson et al. [48]. Another CleanEval contestant, BTE, determines the largest contiguous text area with the least amount of HTML tags and marks it as “full text” [45, 44]. The heuristic is based on the observations that the *tag density* within boilerplate text is higher than within fulltext content and that main content usually is longer than boilerplate text. A similar approach, which uses an n-gram model plus several HTML-based heuristics, mainly focusing on extracting the main content of news articles, has recently been presented by Pasternack et al. [93] and also evaluated against CleanEval, apparently with high accuracy. We analyze a representative set of features used by these approaches for automatic boilerplate classification.

One driving motivation for boilerplate text detection is to improve web search and mining, similar in spirit to simple stop-word removal. Viera et al. [114] introduce an approach based on detecting common subtrees in a few sample pages similar to [123] and observe that clustering and classification accuracy can be improved significantly by removing such common subtrees. Fernandes et al. [43] measure the importance of blocks by a combination of average inverse site frequency of terms in a block, as a measure for block commonality, and the similarity of a block with other blocks on the same page. By weighting terms by their block importance they significantly improve accuracy over the baseline Okapi BM25. I show that densitometric features, which can be computed efficiently online, without resorting to global frequencies, also significantly improves retrieval accuracy.

4.4.2 Web Page Features

Feature Levels

Many features that can be used for the classification of Web page segments have already been described [123, 60, 48, 109]. It is generally expected that the combination of several features can be used to identify text fragments as *headline*, *full text*, *enumeration*, *navigation*, *disclaimer notice* etc., which can then be separated into content and boilerplate text. The number of potential dimensions for this task is huge: text-based strategies like n-gram models can result in tens of thousands of relevant features, which apparently makes the classifier susceptible to overfitting to the contents and layouts of a particular subset. In search of a domain independent, Web-scale solution, I will avoid these token-level features altogether.

Features may be extracted at four different levels: Individual text blocks (elements), the complete HTML document (a sequence of one or more text blocks plus structural information), the rendered document image (the visual representation as in a Web browser) and the complete Web site (i.e., the collection of documents which share a common layout and wording). While the former two levels can apparently be examined for each document locally, the latter two require external information, such as images and CSS definitions for the rendering process and, in order to statistically determine site-level templates and boilerplate text, a sufficiently large number of pages from the same website.

Using features from the two external levels may be highly beneficial to the classification accuracy iff the corresponding data is available. However, there are two major drawbacks. First, rendering pages for classification is a computational expensive operation. Second, template statistics need to be learned separately for each site, they usually cannot be re-used for another website layout. Moreover, it is questionable whether such models are then domain independent (or trained for the news domain only, for instance). I therefore disregard these levels except for one reference feature: the frequency of the text in the whole corpus. Using this feature we can identify phrases commonly used in boilerplate.

Structural Features

Many approaches for Web page segmentation and intra-document text classification utilize structural features in Web pages, such as individual HTML tags (headline, paragraph, anchor text link, image, etc.) or sequences/nested subtrees of HTML tags as well as the presence of particular CSS classes and styles. Of note, the more CSS is used, the less important the semantics of an HTML tag becomes – it is perfectly legal to only use DIV tags and describe the "semantics" of a particular division using style-sheet classes. Unfortunately, CSS classes and sequences of HTML tags are inherently site- and document-specific. Moreover, to fully interpret these rules one has to essentially render the page.

As we want to avoid site-specific signals (which may lead to over-fitting to a particular data set or domain) as well as a costly rendering of pages, I will only examine the following structural features: The presence of a particular headline tag (`H1`, `H2`, `H3`, `H4`, `H5`, `H6`), a paragraph tag (`P`), a division tag (`DIV`) and the anchor text tag (`A`) as an HTML element that encloses a particular text block.

Shallow Text Features

Because boilerplate detection does not inspect text at the *topical* level but rather at the *functional* level, I do not consider the bag of words as classification features. An evaluation at token-level may provide skewed results that describe a particular domain only. Instead, we examine shallow text features at a higher, domain- and language-independent level, which have been discussed in the field of Quantitative Linguistics: *Average word length* (in our definition words are white-space delimited character sequences which at least contain one letter or digit), *average sentence length* (the sentence boundaries are identified by a simple pattern-based heuristic checking for the presence of full stops, question or exclamation marks as well as semicolons) and the absolute number of words.

Another important source for the classification task is the local context, i.e., the absolute and relative position of a text block in the document. If the segmentation granularity is high, it is likely that full-text is followed by full-text and template is followed by template. Moreover, when there is a significant amount of boilerplate text, the main content usually is surrounded by boilerplate (header, footer, left-navigation, right-navigation etc.), not vice versa (i.e., even if the very last text block contains a sentence, if it is a copyright or disclaimer notice, it is regarded boilerplate).

We will also examine a few heuristic features: the absolute number of words that either start with an uppercase letter or are completely upper-case as well as the ratio of these words compared to the total number of words and the ratio of full stops to the overall number of words, the number of date/time-related tokens and the number of vertical bars "`|`" (these characters can sometimes be found in navigational boilerplate text). Moreover, we also compute the link density (called *anchor percentage* in [48]), as the number of tokens within an `A` tag divided by the total number of tokens in the block; for this computation we do not regard the `A` tag as a block separator.

Besides the link density measure, I will also evaluate the text density of each particular block. The text density measure $\varrho(b)$ is used as defined in Equation 4.6 (see Section 4.2.1 for details).

4.4.3 Classification Experiments

Goals and Approach

The goal of this section is to analyze the introduced features for boilerplate detection.

The overall approach is simple: Web pages are segmented into atomic text blocks, which are then annotated with features and on this basis classified into content or boilerplate. Atomic text blocks are sequences of character data which are separated by one or more HTML tags, except for `A` tags – in order to compute the link density.

To train and test classifiers for various feature combinations we start from a known text domain: news articles on the Web. The domain is large and diverse because numerous independent sources contribute to it, is readily available for analysis (e.g. from a news search engine) and the structure is well-understood: Usually one news article (consisting of one or more headlines, the article body and supplemental information like fact boxes, image captions etc.) is surrounded by the standard layout of the publisher's web site (linked headlines and teasers to other news articles, related or not, advertisements, copyright notices etc.). In some cases, the publishers also allow users to comment on the article, comments then appear on the page nearby the article.

For our evaluation, a representative subset of news articles from different sites with different layouts was labeled according to the observed text types (boilerplate/content as well as other classes like headline, user comments etc.) The labeled set is then split into a training and a test set (using a 10-fold cross validation) and fed into a classifier (we will use decision trees and linear support vector machines) to measure the accuracy of the approach. To analyze domain independence, we also evaluate the classifiers against datasets from other domains.

Datasets and Gold Standard

The evaluation is performed on two datasets, a news collection for training and testing and a cross-domain collection for validation.

News Collection. The news collection consists of 621 manually assessed news articles from 408 different web sites. The news articles were sampled randomly from a larger crawl of 254,000 articles from 7,854 web sites which we acquired by monitoring the Google News search engine during the first half of 2008. Using a custom-built crawler, I monitored the news headlines of six different English-speaking Google News portals (USA, Canada, UK, South Africa, India, Australia) and four categories (World, Technology, Sports, Entertainment) and fetched the full text HTML of the corresponding linked articles. The ranked distribution of articles per web site apparently is power-law distributed (maximum number of articles per host: 3774, average: 32.38, median: 5). The top-5 hosts are *ap.google.com*, *afp.google.com*, *reuters.com*, *iht.com*, *news. bbc.co.uk*; at the break-even between rank and frequency (200) is *en.rian.ru* whereas sites like *financeasia.com* and *photonicsonline.com* appear at the bottom. In the examined subset, the maximum number of articles per host is 12 (*news.com.au*) whereas the average and median are 1.52 and 1 respectively. I will use the term "GoogleNews" to describe that subset. In the following sections we focus on this collection except for the frequency of text blocks, which is computed from the complete crawl.

Using a Web browser based text annotator, for each HTML page in the GoogleNews set seven human assessors labeled[10] sequences of text as either *headline*, *fulltext*, *supplemental* (text which belongs to the article but is not fulltext, such as image captions etc.), *user comments*, *related content* (links to other articles etc.). Unselected text is regarded *not content* (boilerplate). The labels were then stored at the level of individual text blocks (i.e., any character sequence that is not interrupted by an HTML tag, except the `A` tag, as described in Section 4.4.2).

The labeling was performed visually in the web browser using a custom-built annotation tool where the assessors only had to select a particular region of text by point-and-click. In the dataset, these block-level labels are stored as nestable HTML SPAN-tags with a specific CSS class at the DOM level just above the elements' text. As they can be easily removed again from the data, this guarantees that the original HTML structure is preserved in its entirety, as opposed to the plain-text approach favored elsewhere [12].

[10] Every page was assessed only once, as no significant inter-assessor disagreement is expected in this context.

Class	# Blocks	# Words	# Tokens
Total	72662	520483	644021
Boilerplate	79%	35%	46%
Any Content	21%	65%	54%
Headline	1%	1%	1%
Article Full-text	12%	51%	42%
Supplemental	3%	3%	2%
User Comments	1%	1%	1%
Related Content	4%	9%	8%

Table 4.8: Class-Distribution in the GoogleNews set

The distribution of classes at three different levels is depicted in Table 4.8; we count the number of text blocks, words and unfiltered tokens (including non-words) separately. The raw data and the gold standard of the annotated subset of Google-News are available online.[11]

Cross-Domain Collection. The CleanEval collection is a benchmark corpus created particularly for the eponymous boilerplate removal competition of the ACL Web-as-Corpus community [12]. The collection consists of 798 raw HTML pages randomly sampled from Web search engines, from which 733 pages have already been manually assessed and split into a training set of 58 documents and a test set of 675 documents. The assessment was performed as follows. After converting the HTML document into plain text, all boilerplate text has manually been removed and remaining text has been structurally labeled as *paragraph*, *headline* or *list element*. The raw data and the gold standard are available online.[12] Unfortunately, because the assessors worked with plain text that has been derived from a Browser-rendered representation of the documents, the gold standard cannot directly be used for an analysis at HTML level. I will nevertheless evaluate my approach against this benchmark to allow a comparison to other CleanEval contestants.

[11] L3S-GN1 collection `http://www.L3S.de/~kohlschuetter/boilerplate`

[12] `http://nlp.fi.muni.cz/~xpomikal/cleaneval/`

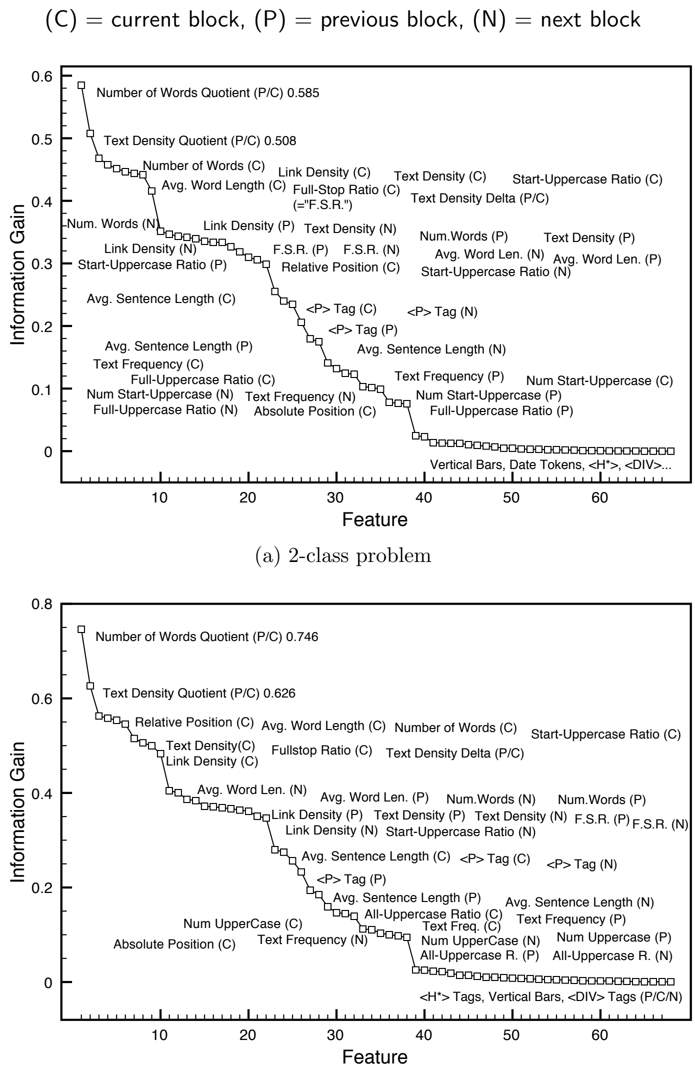
Evaluation

Training and Testing on GoogleNews. As we see from Table 4.8, the class distribution is strongly dominated by *Boilerplate* and *Article Full Text*; the other four classes quantitatively play a minor role. The class *User Comments* was only assessed to quantify the amount of comments text compared to the remaining full text; for the purpose of boilerplate detection we treat comments as main content. Because of the strongly skewed distribution of the initial six text classes, we evaluate a two-class problem (boilerplate vs. content) and a four-class problem (boilerplate vs. full-text/comments, headline, supplemental) separately. The four-class problem generally is more difficult to solve, so we may expect lower accuracies here.

Using Weka, we examine the per-feature information gain and evaluate machine-learning classifiers based on Decision Trees (1R and C4.8) as well as Support Vector Machines (SMO in particular). We measure classification accuracy by Precision, Recall, F_1-Score, False Positive Rate and ROC Area under Curve (AuC); all scores are normalized based on the number of words in a block, i.e., large blocks are weighted higher than small blocks. Figure 4.20 shows the features in decreasing order of their information gain.

Generally, very simple features like *Relative Position*, *Average Word Length* and *Number of Words* of the current block appear to be strong indicators for class membership. For the sake of clarity, let us test the classification accuracy of these highly ranked features separately. Interestingly, the text-flow capturing variants of these features (*Number of Words Quotient*, *Text Density Quotient* (TDQ), *Relative Position*), which relate the value of the current block to the previous one, provide the highest information gain, indicating that the intra-document context plays an important role; this is also substantiated by the classification results. Table 4.9 presents the evaluated algorithms and the achieved accuracies along with the number of features (dimensions) used and the number of leaves for all decision-tree-based algorithms.

The classification baseline is the *ZeroR* classifier, which in our case always predicts *Article Full-text* (4-class problem) and *Content* (2-class problem). Due to the class weights this results in an ROC area-under-curve of less than 50%.



(a) 2-class problem

(b) 4-class problem

Figure 4.20: Per-Feature Information Gain for the GoogleNews collection

The *1R* classifier determines the feature with the least error rate and partitions the corresponding numeric values to derive simple classification rules (the number of partitions equal the number of leaf nodes in a decision tree). 1R over all features resulted in a simple rule with an acceptable accuracy: Any block with a text density less than 10.5 is regarded boilerplate. I analyzed the 1R partitioning also for the features *Average Sentence Length*, *Average Word Length*, *Link Density* and *Number of Words* and got similar (slightly lower) accuracies. However, *Average Word Length* is fairly unsuitable for classification, as 1R generates many partitions between average word length 4 and 5 which alternate between *Boilerplate* and *Article Content*.

We get promising results from the C4.8-based decision-trees. In order to avoid overfitting, the algorithm has been configured to only consider leaves matching at least 1000 instances. By using all the 67 available features (including features from the previous and next blocks) we get a remarkable ROC AuC of 98% for the 2-class problem and 96.9% for the 4-class problem; we also achieve similar results using an SMO support-vector machine with a linear kernel. Moreover, by applying reduced-error pruning I was able to simplify the decision tree to only use 6 dimensions (2 features each for current, previous and next block) without a significant loss in accuracy (ROC AuC 96.9% for the 2-class problem), see Algorithms 4.2 and 4.3.

Application to CleanEval and Re-Validation. To test the domain-independence of the determined classifiers, I applied the two simplified C4.8 classifiers to the CleanEval collection. I evaluated the 2-class problem (boilerplate vs. content of any kind, called *TO* in CleanEval) for the classifier that has been trained for the GoogleNews collection and one that has been trained on the CleanEval training set. In the latter case, the decision rule was even simpler: accept all blocks that have a minimum text density of 7 and a maximum link density of 0.35.

Because the CleanEval collection only provides assessments and the algorithmic output at text-level, we cannot directly reuse the setup we used for the GoogleNews evaluation. The CleanEval initiative provides their own accuracy measure which is

Algorithm	**Dim**	**Precision**		**Recall**		**F_1-Score**		**FP Rate**		**ROC AuC**		**Leaves**	
ZeroR (baseline; predict "Content")	0	40.7	35.0	63.8	59.2	49.7	44.0	63.8	59.2	49.0	48.9	–	–
Only Avg. Sentence Length	1	78.5	72.4	67.9	66.6	68.0	65.2	21.4	22.3	73.3	72.1	2	4
C4.8 Element Frequency (P/C/N)	3	77.7	70.9	76.2	73.2	73.8	69.8	38.3	34.7	70.9	69.8	9	4
Only Avg. Word Length	1	80.2	77.4	77.0	74.8	77.5	73.5	19.5	20.0	78.8	77.4	2	178
Only Number of Words @15	1	86.7	80.9	86.7	84.9	86.7	82.8	15.5	15.5	85.6	84.7	2	2
Only Link Density @0.33	1	88.5	81.7	87.8	83.8	87.4	81.4	19.2	22.5	84.3	80.7	16	2
1R: Text Density @10.5	1	87.8	81.4	87.9	85.4	87.9	83.4	14.3	15.3	86.8	85.0	2	2
C4.8 Link Density (P/C/N)	3	91.1	83.7	91.1	87.4	91.0	85.4	12.1	14.3	94.2	90.8	37	32
C4.8 Number of Words (P/C/N)	3	91.1	87.6	90.8	89.3	90.9	87.6	8.9	1.0	94.7	94.6	40	48
C4.8 All Local Features (C)	23	92.9	88.7	92.9	89.9	92.9	88.7	8.7	10.6	96.6	95.7	102	72
C4.8 NumWords + LinkDensity, simplified	**6**	**92.2**	**84.8**	**92.2**	**88.9**	**92.2**	**86.8**	**10.1**	**12.1**	**95.7**	**93.3**	**8**	**7**
C4.8 Text + LinkDensity, simplified	**6**	**92.4**	**84.7**	**92.4**	**88.8**	**92.4**	**86.7**	**8.5**	**11.4**	**96.9**	**93.2**	**8**	**12**
C4.8 All Local Features (C) + TDQ	25	92.9	89.2	93.0	90.3	92.9	89.1	8.3	9.4	97.2	96.1	109	78
C4.8 Text+Link Density (P/C/N)	**6**	**93.9**	**89.3**	**93.8**	**91.0**	**93.9**	**89.5**	**6.7**	**8.4**	**97.6**	**96.1**	**51**	**45**
C4.8 All Local Features (P/C/N)	64	95.0	91.3	95.0	92.4	95.0	91.4	5.5	7.1	98.1	96.7	105	98
C4.8 All Local Features + Global Freq.	**67**	**95.1**	**91.5**	**95.0**	**92.5**	**95.1**	**91.6**	**5.4**	**6.8**	**98.0**	**96.9**	**99**	**125**
SMO All Local Features + Global Freq.	67	95.3	92.4	95.3	93.2	95.3	91.9	5.4	6.8	95.0	94.0	-	-

Table 4.9: Weka Classification Accuracies for the Google News Collection (2-class/4-class problem)

Algorithm 4.2 Densitometric Classifier

```
currLinkDensity <= 0.333333
| prevLinkDensity <= 0.555556
| | currTextDensity <= 9
| | | nextTextDensity <= 10
| | | | prevTextDensity <= 4:  BOILERPLATE
| | | | prevTextDensity > 4:  CONTENT
| | | nextTextDensity > 10:  CONTENT
| | currTextDensity > 9
| | | nextTextDensity = 0:  BOILERPLATE
| | | nextTextDensity > 0:  CONTENT
| prevLinkDensity > 0.555556
| | nextTextDensity <= 11:  BOILERPLATE
| | nextTextDensity > 11:  CONTENT
currLinkDensity > 0.333333:  BOILERPLATE
```

Algorithm 4.3 NumWords Classifier

```
currLinkDensity <= 0.333333
| prevLinkDensity <= 0.555556
| | curr_numWords <= 16
| | | next_numWords <= 15
| | | | prev_numWords <= 4:  BOILERPLATE
| | | | prev_numWords > 4:  CONTENT
| | | next_numWords > 15:  CONTENT
| | curr_numWords > 16:  CONTENT
| prevLinkDensity > 0.555556
| | curr_numWords <= 40
| | | next_numWords <= 17:  BOILERPLATE
| | | next_numWords > 17:  CONTENT
| | curr_numWords > 40:  CONTENT
currLinkDensity > 0.333333:  BOILERPLATE
```

based upon a weighted Levenshtein Edit Distance at token-level [12]. The computation is expensive and also not essential for this task. I confirmed that the scores can be approximated well with the much simpler bag-of-words token-level F_1 score (like in the GoogleNews setup, except that class weights are not taken into account).

As our scores therefore slightly differ from the ones in [12], I re-evaluated the available results of three CleanEval contestants (BTE, Victor, NCleaner) and also added the heuristics by Pasternack et al. [93] (in two flavors, the unigram model trained on CleanEval and the trigram model trained on a news corpus) to the set of competitors, as well as a baseline ("Keep all text") and a classifier solely based on the feature with the highest information gain: number of words; we mark every block with at least 10 words as content.

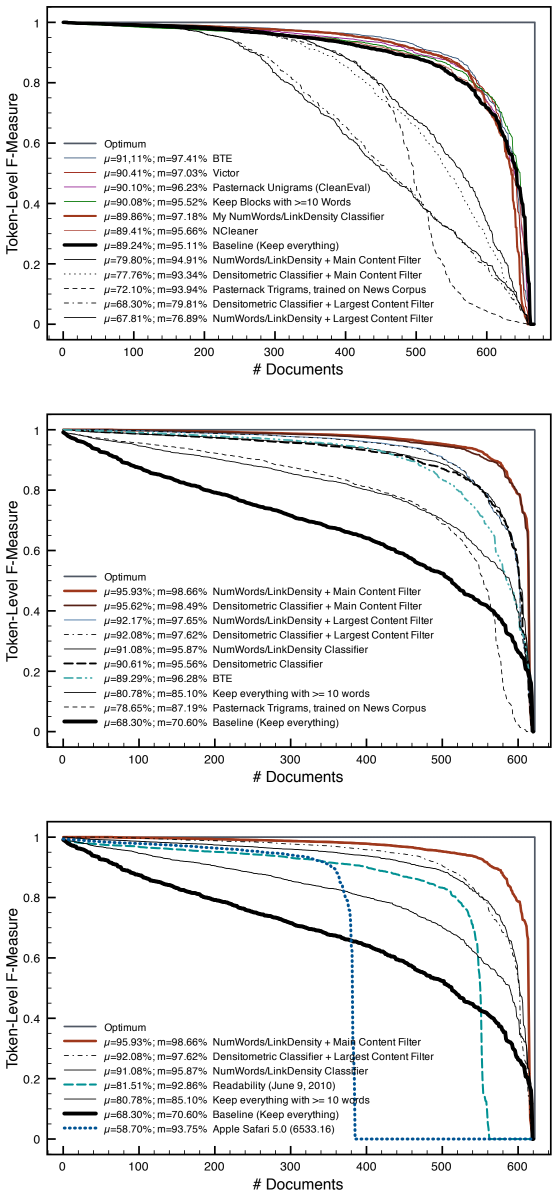
The average (μ) and median (m) as well as the ranked accuracy for each evaluated strategy are depicted in Figure 4.21a. We see that the two flavors of the Pasternack heuristic drastically differ in terms of accuracy. We assume that the algorithm needs proper training to succeed for a particular corpus, and the trigram model from the news domain was not generic enough for the CleanEval dataset.

Additionally, to understand how far heuristic additions could further improve the classification, I extended the two decision tree classifiers downstream with handcrafted rules. In one extension, we only take the content block with the highest number of words (*Largest Content Filter*). In another extension, we add rules that are specific for news (*Main Content Filter*): It extends Largest Content Filter by removing any text that is below a clearly identifiable comments section (a block solely containing one out of 11 indicator strings like "User comments:" etc.) and above a clearly identifiable title (derived from the HTML document's `TITLE` value). As we can see from Figure 4.21a, for the CleanEval collection these modifications resulted in much worse results than the baseline. On the other hand, the very simple strategy to keep all blocks with at least 10 words (as well as our NumWords/LinkDensity classifier) performed just as good as the Pasternack unigram and the NCleaner setups that have specifically been trained for CleanEval.

Ultimately, we see (Figure 4.21a) that basically keeping all text – i.e., not removing anything – would be a good strategy, being only marginally worse than the apparently best solution (BTE)! This leads to the question whether there were failures in the assessment process, whether the collection is comparable to the GoogleNews collection or at all appropriate for the purpose of boilerplate detection.

I repeated this evaluation for the GoogleNews collection, computing accuracy scores in the same way using the same algorithms (BTE as the alleged winner for CleanEval, Pasternack Trigrams as a supposedly mediocre strategy and the algorithms introduced here). For the Pasternack algorithm, I used the Web service provided by the authors; unfortunately, there was no unigram implementation available. Figure 4.21b shows the corresponding results.

We see that the baseline for GoogleNews is much lower than for CleanEval; all tested algorithms perform differently and are usually better than the baseline, except for the Pasternack strategy, which under-performed in a few cases. Overall its performance is lower than expected, given the fact that it has been trained on news articles. I can only assume a bug in their implementation or high overfitting towards a particular subset of news sites. The strategy to just keep everything with a minimum of 10 words did not work very well either, although better than the Pasternack



(a) CleanEval corpus

(b) GoogleNews corpus

(c) GoogleNews (ctd.)

Figure 4.21: Performance of Boilerplate Detection Strategies

trigrams and, on average, improves the baseline by 18.3%. BTE is on par with the two simplified classifiers (using *text density* and *number of words* respectively); it is a little bit better for the median but worse on average. The classifier based on the number of words per block and its link density yields improve the baseline by 33.3%.

In the end, the use of the two heuristic filters (*Largest/Main Content Filter*) can further improve the detection accuracy for the GoogleNews dataset to an almost perfect average F_1 score of 95.93% (this is a 40% relative improvement over the baseline). Even though we see that these algorithms failed for CleanEval, we expect them to work generically for the news domain. On the other hand, we see that both, BTE and our two simplified classifiers work quite well for both collections.

Of note, our classifier is far more efficient than BTE. It runs in linear time, whereas BTE has a quadratic upper bound. Furthermore, it can return more than a single piece of text. BTE's assumption that only one block (the largest having the least tag density) completely covers the main content seems not to hold for all cases: compared to the densitometric classifier, BTE only achieves a suboptimal accuracy in our news experiment. This discrepancy may be due to the fact that the CleanEval collection actually contains less boilerplate text than all other collections we examined (only very little words are in blocks with a text density lower than 11, see Figure 4.22).

As a last remark, I also compared my approach against two heuristics that have not been discussed in the academia yet but are recently becoming popular, the "Readability" browser bookmarklet[13] and Apple's Safari Reader[14] (which is based upon a modified version of Readability). Both techniques indeed perform boilerplate removal, particularly for re-rendering the Web page's text to improving reading ease. As shown in Figure 4.21c, these approaches are surprisingly subpar compared to all other algorithms except Pasternack's trigram algorithm. Of note, Safari Reader at all delivered in only about two third of all documents results; in all other cases the functionality was marked as "not available". Obviously, there is much room for improvement.

[13] `http://lab.arc90.com/`

[14] `http://apple.com/safari`

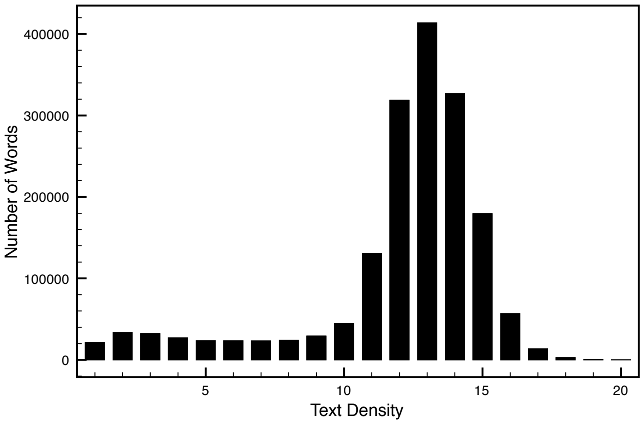


Figure 4.22: CleanEval Text Density Distribution

4.4.4 Quantitative Linguistic Analysis

Text Density vs. Number of Words

In all cases but one, the classifier using *Number of Words per Block* performed slightly better than the classifier using *Text Density.* Also it seems sufficient for a good classification. To get a better understanding why this strategy performs so well, we need to analyze the created decision tree rules (see Algorithms 4.2 and 4.3). We see that the two classifiers do not differ for the link density-specific rules; if the text block consists of more than 33% linked words, it is most likely boilerplate, unless the block is surrounded by long/dense text blocks.

Actually it is likely that both measures, *text density* as well as *number of words* describe the same fundamental principle of text, which however is more visible through text density than through the plain number of words. As the absolute number of words theoretically is unbounded (the longest block in the GoogleNews collection consisted of 1122 words), yet dominated by boilerplate (79% of all blocks, see table 4.8), a straight visual distinction between boilerplate and content is hard to spot (Figure 4.23; the “content” part is magnified for clarity). Also, the rules generated for the number-of-words classifier are difficult to interpret. It is unclear, for instance, why

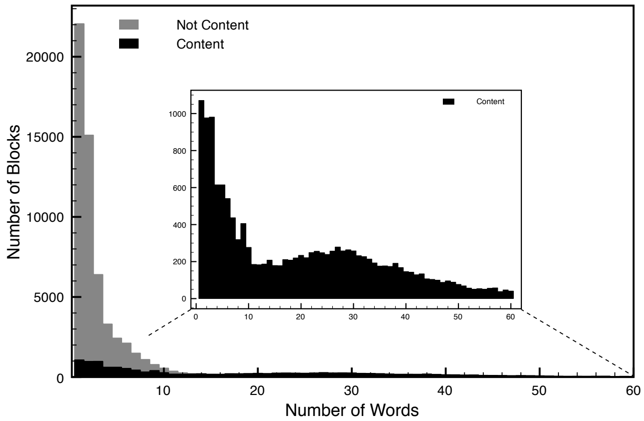


Figure 4.23: Number of Words Distribution (GoogleNews)

everything exactly above *40* words is regarded full text and not already at *30* etc. However, if we look at the very same data using the text density metric (rounding the density to the nearest integer $[\varrho(b)]$), we can clearly identify three modes of a mixed distribution (see Figure 4.24), which are represented in the classifier.

In Section 4.3 I showed for a representative Web corpus (Webspam-UK 2007, ham-part) that the distribution of words in blocks with a particular text density can effectively be modeled as a combination of two beta distributions and a normal distribution. Each beta distribution is assumed to represent one class of text, "full-text" content (complete sentences) and "template text" (incomplete sentences); the normal distribution acts as a fuzzy transition between them. In fact, I was able to apply the same model to the GoogleNews collection data, with a very high goodness of fit ($R^2 = 0.997$, RMSE $= 0.0213$). As opposed to my initial results on an unlabeled corpus, we now have manually labeled annotations, so this hypothesis can finally be examined at a higher level of confidence.

Indeed, the main article's full text, as well as the user comments and, to some degree, supplemental text, can basically be described as blocks with a text density $\geqq 10$ (in fact, the 1R algorithm suggested this split, achieving a ROC AuC of 86.8%). The remaining blocks with a lower density almost completely describe boilerplate

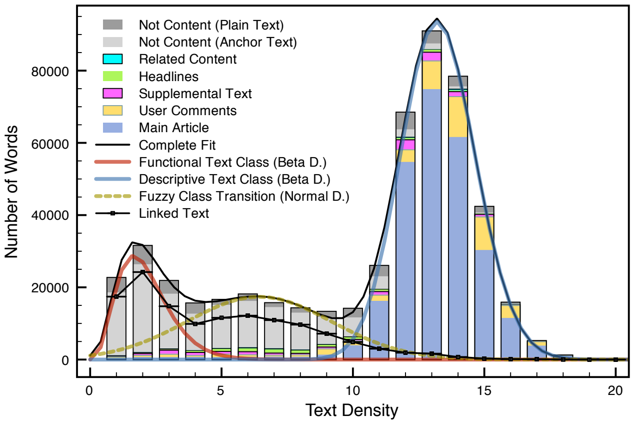


Figure 4.24: Text Density Distribution by class (GoogleNews)

text (headlines appear at text density 4 and higher; some “supplemental text” may expose an even lower text density). Moreover, the fuzzy class seems to be strongly dominated by linked text (hypertext), which might explain why the addition of the link density feature significantly improves the classification accuracy.

Obviously, the text density measure helps us to visualize the mixed distribution in a compact way (much better than the absolute number of words – compare Figure 4.23), even though it appears that for the actual purpose to separate and to classify the two types of text (template and fulltext) the absolute *number of words* per block are sufficient, and thus to be preferred (Occam's Razor).

Actually we can approximate the density distribution for visualization purposes solely using the number of words as follows. From the definition of text density (Equation 4.6) we see that two cases are differentiated: wrapped text (i.e. covering more than one line) and unwrapped text (i.e. only one line). If the line is wide enough (we used 80 characters), all densities below a certain number of words λ describe one-line blocks (except for the unusual case where blocks contain very long words), or combinations thereof. In order to reach a line wrap boundary, a certain number of words need to be written, and thus a large text density score indicates a larger number of words. In fact, the difference of the number of words contained in blocks with at least $\lambda = 11$ words (345.175) to the number of words contained in blocks with a text density of at least 11 (358.428) is insignificant (3.8%).

Stochastic Text Model

The fairly clear separation between short boilerplate and longer content blocks with respect to text density suggests a simple generative process: First, let us find a sufficiently good model for the overall process of generating words. We can see the creation process of text blocks as a Shannon random writer [105].

Imagine the author decides with some probability to write a word or to finish the current block and proceed to the next one. This essentially is a first-order Markov process with two states, T (*add another word to the text*) and N (*skip to the next block*); see Figure 4.25a. The probability of staying in the same state is always the complementary probability of moving to the other state. As we can regard subsequent newlines as a single operation, we have $P_N(N) = 0$ and thus $P_N(T) = 1$ (after skipping to a new block, always at least one word is written).

The state transitions from T can be modeled as a simple Bernoulli trial. Consider the transition to N as *success* (p) and the emission of another word as failure $(1-p)$. The probability that there are k failures (for $k = 0, 1, 2, 3, ...$) before the first success is $Pr(Y = k) = (1-p)^k p$.

Coming from state N means we already have emitted one word, so the actual probability for emitting x words ($k-1$ failures) in the simple scenario then is

$$Pr(Y = x) = (1-p)^{x-1} \cdot p = P_T(T)^{x-1} \cdot P_T(N) \qquad (4.20)$$

which is the 1-displaced geometric distribution; it has extensively been discussed in the field of Quantitative Linguistics [4]. While there are more sophisticated, better matching models for describing this process, the geometric distribution is a good starting point, particularly at corpus level where the individual variations of certain authors become indistinct.

Applied to the GoogleNews corpus, for $P_T(N) = 0.3145$ we achieve a goodness of fit of $R^2_{adj} = 96.7\%$ with a root mean square error ($RMSE$) of 0.0046.

Let us now add the two different types of text that we have discovered in our evaluation, *short* and *long* text. We extend the simple random writer by transforming the state T into two separate states S (print a word of short text) and L (print a word of long text), see Figure 4.25b. As L and S replace T, the probabilities to arrive at L or S coming from N must sum up to $P_N(T) = 1$. Once in the state L or S, we again have a Bernoulli trial: either continue producing words (all of long or short text respectively, no intra-block mixtures) or terminate and go back to state N. As we expect from short text to terminate quickly after a few words and from long text to terminate after a higher number of words, we require that $P_S(N) \gg P_L(N)$.

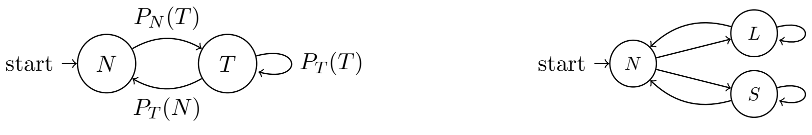
In this mixed scenario, the probability density distribution therefore is:

$$\begin{aligned} Pr(Y = x) = P_N(S) \cdot \left[P_S(S)^{x-1} \cdot P_S(N)\right] + \\ + P_N(L) \cdot \left[P_L(L)^{x-1} \cdot P_L(N)\right] \end{aligned} \tag{4.21}$$

Applying this model to the GoogleNews data results in a higher goodness of fit of $R^2_{adj} = 98.81\%$ with $RMSE = 0.0027$ for $P_N(S) = 1 - P_N(L) = 0.7586$, $P_S(N) = 0.3968$ and $P_L(N) = 0.04369$, which supports our assumption of the mixture, even if the geometric distribution only is a rough approximation. As the geometric distribution's expected value is defined as $E(x) = p^{-1}$ (short text has its mean at $0.3968^{-1} = 2.52$, long text at $0.04369^{-1} = 22.89$) and the determined probability $P_N(S) = 76\%$ is close to the assessed 79% (amount of blocks classified as boilerplate, see Table 4.8), we may attribute a large extent of short text to the *Boilerplate* class and most long text to the *Content* class (see Figure 4.23).

Linguistic Interpretation

The observed compound distribution can be regarded not of arbitrary nature but of a stochastic, quantitative linguistic one, implying that actually two different classes (strata) of text are embedded in the Web content. In the field of Quantitative Linguistics it is generally assumed that text creation process can be modeled as *urn trials* at the level of various linguistic units such as *phoneme*, *word*, *sentence*, *text segment* etc. and for several shallow features such as *frequency*, *length, repeat rate*, *polysemy* and *polytextuality* [115]. Even though it is still unclear by which exact parameters this process is driven, we can model it as a Bernoulli process. Through empirical experiments and simple stochastic concepts, I have shown that this model can be applied to describe the process of content creation on a Web page.



(a) Simple

(b) Stratified

Figure 4.25: Random Writer Models

While we cannot explain the reasons for choosing short or long text at some particular point in any given document, we can interpret the statistically observed behavior at corpus level (which is the level of granularity we are interested in, see the problem statement in Section 1.2): When composing a Web page, an author chooses with some probability whether she wants to write a sequence of actual full sentences or navigational elements. The choice surely depends on the context (hence we observe an improvement when the features from the previous and next block are taken into account, and this is why the geometric distribution does not fit perfectly). The use of full sentences usually means the author wants to make a more or less complex statement which needs grammatical constructs, long explanations etc. Extensive coding (= many bits) is required because the author (sender) does not expect that the audience (receivers) understand the information without explanation. This kind of text (*long text*) therefore can be regarded of *descriptive* nature (i.e., supplying the reader with the subject matter's details at the cost of higher syntactic complexity, just like the full text of this thesis). The second kind of text (*short text*), grammatically incomplete or simple sentences, consisting of only a few words, is used whenever a quick, economic coding is possible, i.e. when the audience is expected to perceive and understand the encoded information without large effort (= few bits), e.g., “Contact us”, “Read more”. Such text is often used for headlines and navigational text (one kind of boilerplate). We can therefore regard the latter form of text of *functional* nature. While there are noticeable exceptions, it appears that, at least for the Web, there is a strong correlation between *short text* and *boilerplate text* as well as between *long text* and *content text*, which explains why the simple classification works so well.

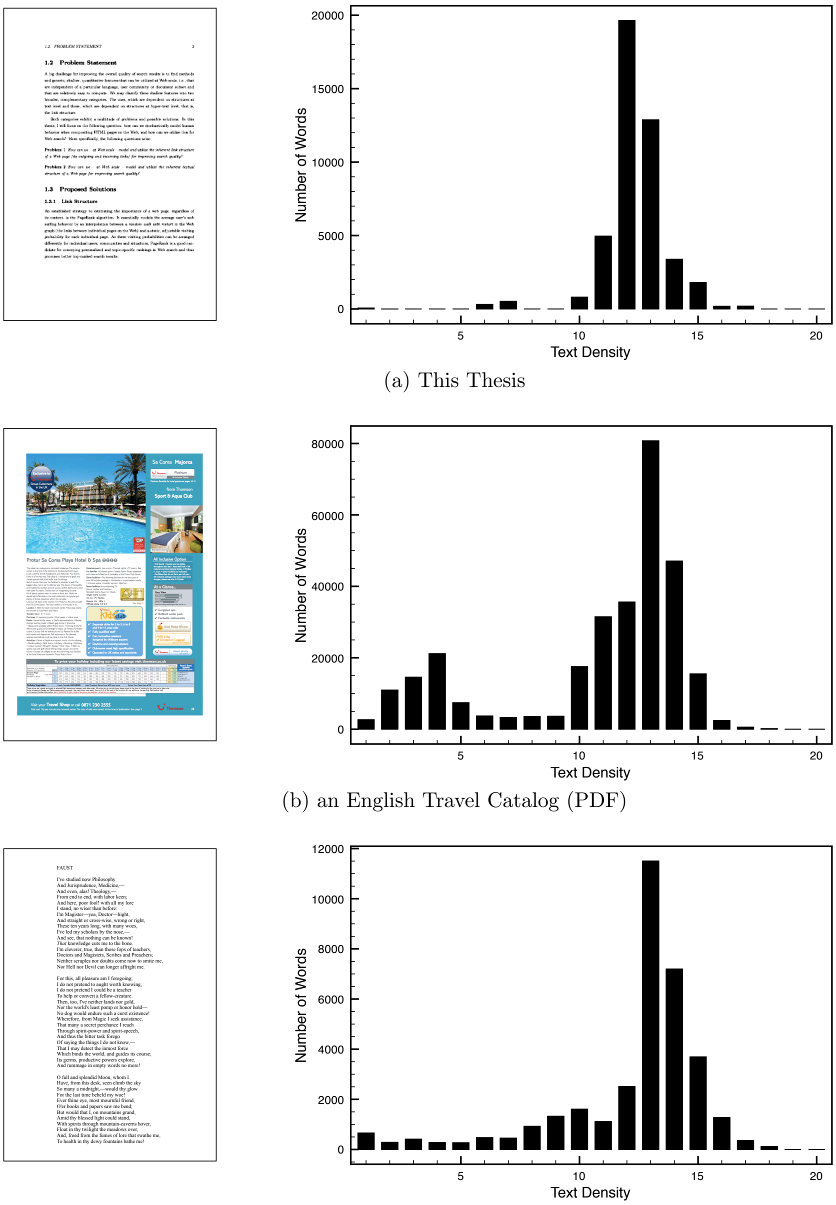
These two strata of text can be visualized in a compact form through the text density measure because it is mostly irrelevant how many words an author spends within an individual text. As soon as she writes complete, non-trivial sentences (i.e., more than ca. 10 words in English) the produced text most likely falls into the *descriptive* class. Text density exactly provides this value-limiting boundary. By word-wrapping text at a predetermined line width (which is dependent upon the average sentence length in characters) and dividing the number of words by the number of lines, we literally "construct" this two-fold distribution and thus can better visualize what was already present in the raw number of words. An incomplete sentence will never wrap to more than one line (in this case text density equals to the number of words), whereas text consisting of complete sentences will always wrap, be averaged to the "typical" number of words in a sentence and encoded as a density value of a rather limited range. This limited range can then be better visualized histographically, as demonstrated.

To the best of my knowledge, the distinction between short and long text has not been discussed in the context of Quantitative Linguistics so far. This is probably because the amount of short text in the previously analyzed "offline works" is almost negligible (this also holds[15] for the present paper, see Figure 4.26a, or Project Gutenberg's online version of Goethe's Faust[16] 4.26c). On the other hand, we may find higher amounts of short text in brochures, tabloids etc, as for example in a travel catalog[17] (see Figure 4.26b). An in-depth analysis of such content needs to be conducted as future work.

[15] The texts have been converted to HTML in order to compute text density as defined here.

[16] http://www.gutenberg.org/files/14591/14591-h/14591-h.htm

[17] Thomson Summer Collection 2011, http://www.thomson.co.uk/



1.2. PROBLEM STATEMENT 3

1.2 Problem Statement

A big challenge for improving the overall quality of search results is to find methods and generic, shallow, quantitative features that can be utilized at Web-scale, i.e., that are independent of a particular language, user community or document subset and that are relatively easy to compute. We may classify these shallow features into two broader, complementary categories. The ones, which are dependent on structures at text level and those, which are dependent on structures at hyper-text level, that is, the link structure.

Both categories exhibit a multitude of problems and possible solutions. In this thesis, I will focus on the following question: how can we stochastically model human behavior when compositing HTML pages on the Web, and how can we utilize this for Web search? More specifically, the following questions arise:

Problem 1 *How can we – at Web scale – model and utilize the inherent link structure of a Web page (its outgoing and incoming links) for improving search quality?*

Problem 2 *How can we – at Web scale – model and utilize the inherent textual structure of a Web page for improving search quality?*

1.3 Proposed Solutions

1.3.1 Link Structure

An established strategy to estimating the importance of a web page, regardless of its content, is the PageRank algorithm. It essentially models the average user's web surfing behavior by an interpolation between a random walk with restart in the Web graph (the links between individual pages on the Web) and a static, adjustable visiting probability for each individual page. As these visiting probabilities can be arranged differently for individual users, communities and situations, PageRank is a good candidate for conveying personalized and topic-specific rankings in Web search and thus promises better top-ranked search results.

(a) This Thesis

(b) an English Travel Catalog (PDF)

FAUST

I've studied now Philosophy
And Jurisprudence, Medicine,—
And even, alas! Theology,—
From end to end, with labor keen;
And here, poor fool! with all my lore
I stand, no wiser than before:
I'm Magister—yea, Doctor—hight,
And straight or cross-wise, wrong or right,
These ten years long, with many woes,
I've led my scholars by the nose,—
And see, that nothing can be known!
That knowledge cuts me to the bone.
I'm cleverer, true, than those fops of teachers,
Doctors and Magisters, Scribes and Preachers;
Neither scruples nor doubts come now to smite me,
Nor Hell nor Devil can longer affright me.

For this, all pleasure am I foregoing;
I do not pretend to aught worth knowing,
I do not pretend I could be a teacher
To help or convert a fellow-creature.
Then, too, I've neither lands nor gold,
Nor the world's least pomp or honor hold—
No dog would endure such a curst existence!
Wherefore, from Magic I seek assistance,
That many a secret perchance I reach
Through spirit-power and spirit-speech,
And thus the bitter task forego
Of saying the things I do not know,—
That I may detect the inmost force
Which binds the world, and guides its course;
Its germs, productive powers explore,
And rummage in empty words no more!

O full and splendid Moon, whom I
Have, from this desk, seen climb the sky
So many a midnight,—would thy glow
For the last time beheld my woe!
Ever thine eye, most mournful friend,
O'er books and papers saw me bend;
But would that I, on mountains grand,
Amid thy blessed light could stand,
With spirits through mountain-caverns hover,
Float in thy twilight the meadows over,
And, freed from the fumes of lore that swathe me,
To health in thy dewy fountains bathe me!

(c) Goethe's Faust (English translation by Bayard Taylor)

Figure 4.26: Text Density Distributions

4.4.5 Retrieval Experiments

Setup. In this section we quantify the impact of boilerplate detection to search. The obvious assumption here is that boilerplate not only is another sort of text, it may also deteriorate search precision, particularly in those cases where keywords match "related articles" text or other keywords that are non-relevant to the actual main content. To evaluate this hypothesis for a representative scenario, we examine yet another domain of Web documents: Blogs.

Blogs are particularly relevant for this task because we may expect many links from one blog page to other blog entries, being topically or temporally related, and those links often include headlines and teaser texts of the referenced item. Moreover, a TREC reference collection already exists, containing 3 million permalink documents retrieved from 100.000 different feeds, along with test queries (consisting of one to five words each) and document assessments at (TREC'06 Blog Track [91]) which were mainly used for measuring opinion retrieval performance. They used graded relevance scores (non-relevant, topically relevant and three levels indicating positive, negative and mixed opinions).

I will use this collection for the evaluation. I indexed the BLOGS06 collection using the Lucene IR library. Separate parallel indexes were created for document blocks with a particular number of words or a particular text density; this allows a selection of permitted ranges at runtime without reindexing. If our assumption holds, one can expect an improvement of the search precision when only words of the *descriptive* (= long) text class are considered for search.

Evaluation. We perform 50 top-k searches (with $k = 10$) for the queries defined in the TREC'06 Blog Track and evaluate Precision at rank 10 (P@10; binary relevance as in the TREC'06 benchmark results) as well as NDCG at rank 10 (graded relevancies as in the TREC'06 assessments). Using the standard Lucene ranking formula we perform 50 searches from queries predefined in the TREC'06 Blog Track and count the number of documents in the top-10 results that have been marked relevant by the TREC assessors. We repeatedly issue the queries for a sliding minimum text density between 1 and 20 and a sliding minimum number of words from 1 to 100 respectively (a minimum of 1 word equals to the baseline). As we only remove *short*

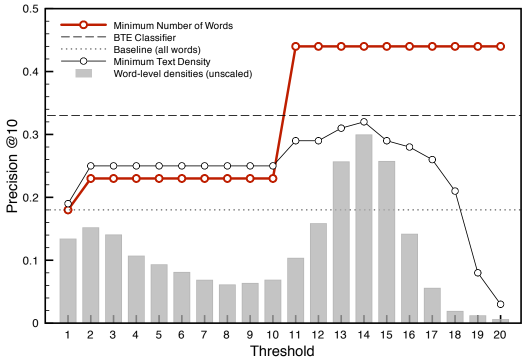


Figure 4.27: BLOGS06 Search Results

text, there is no need for a *maximum* bound. We benchmark the performance of the BTE algorithm for this task and compare $P@10$ as well as $NDCG_{10}$ to our solution. Finally we compare the results to the P@10 scores reported from TREC'06.

Using the sliding minimum *Text Density* we can significantly improve precision (the baseline results in $P@10= 0.18$; $NDCG_{10} = 0.0985$): At the minimum threshold of 14 (with slightly lower values for the surrounding densities between 11 and 20) we get $P@10 = 0.32$ and $NDCG_{10} = 0.1823$, which is almost equal to the scores of the BTE algorithm ($P@10 = 0.33$ and $NDCG_{10} = 0.1627$). For the simple sliding minimum *Number of Words*, we achieve a remarkable accuracy of $P@10 = 0.44$ and $NDCG_{10} = 0.2476$ for any minimum threshold between 11 and 100 words (an improvement by 144%/151% over the baseline and 33%/52% over BTE). I did not examine higher thresholds for practical reasons; at some point, of course, the precision would drop again because of lacking input. Figure 4.27 depicts the results for $P@10$; the $NDCG_{10}$ curves expose identical behavior.

In a direct comparison with the BLOGS06 competition, our results are of course somewhat lower since the strategy does not do opinion mining at all. However boilerplate removal seems to be strongly beneficial for this purpose: we can still compete with the lower 4 of the 16 contestants. One can therefore expect that the addition of our strategy to the opinion mining pipeline further increases accuracy.

4.4.6 Discussion

In this section, I presented a simple, yet effective approach for boilerplate detection using shallow text features, which is theoretically grounded by stochastic text generation processes from Quantitative Linguistics.

I have shown that textual content on the Web can apparently be grouped into two classes, *long text* (most likely the actual content) and *short text* (most likely navigational boilerplate text) respectively. Through a systematic analysis I found that removing the words from the short text class alone already is a good strategy for cleaning boilerplate and that using a combination of multiple shallow text features achieves an almost perfect accuracy. To a large extent the detection of boilerplate text does not require any inter-document knowledge (frequency of text blocks, common page layout etc.) nor any training at token level.

I analyzed my boilerplate detection strategies on four representative multi-domain corpora (news, blogs and cross-domain) and also evaluated the impact of boilerplate removal for document retrieval. In all cases we achieve significant improvements over the baseline and accuracies that withstand and even outperform more complex competitive strategies, which incorporate inter-document information, n-grams or other heuristics.

The costs for detecting boilerplate are negligible, as it comes down simply to counting words.

Chapter 5

Conclusions and Future Work

In this chapter, I summarize my findings about the Web's links and text structure and, as a conclusion, formulate a unified model describing the observed patterns of human behavior.

5.1 Summary

In the present thesis, I have discussed the quantitative power of the Web's inherent structures for the purpose of improving search quality. In particular, I focus on link structure and text structure, and on the possible stratification of these structures.

With respect to link structure, I have shown that the well-known PageRank algorithm can efficiently be parallelized and distributed by separating internal and external links (at host-level) and by aggregating cross-partition link importance, thus reducing the inter-processor communication to only 0.6% of the original load in the examined Web corpus; the algorithm can be regarded state of the art [84].

As a direct consequence of being able to quickly compute PageRank vectors for large graphs, I examined the possibility of using topic-specific PageRank on deeper levels of the ODP Web catalog. I could show that biasing works well until around level 5, at deeper levels the PageRank vectors are not significantly different from each other, thus indicating a real-word upper boundary and, at the same time, a good indication that using biases deeper than just level 1 indeed makes sense. Consequently, I evaluated the use of these Topic Sensitive PageRank vectors to identify topics in search results, leading to a new type of Faceted Search: the membership of a page to a particular topic dimension can be seen quantitatively as the ratio between the logarithm of the page's topic-specific PageRank and the overall number of topics the page is regarded a member (membership is defined as possessing a PageRank score above a given static threshold).

With this approach, I was able to predict a query's topic in 91.6% of all cases just using link structure, without ever inspecting the pages' full-text.

With respect to text structure, I have shown that Web text is composed from two different types of text and that these types can be very accurately separated by observing lengths (number of words) and densities (text density – number of words per limited area, as well as link density – number of linked words vs. overall number of words). I introduced the notion of text density and the text-density-based BlockFusion algorithm for Web page segmentation, which I have derived from Computer Vision and ported from the level of bitmap images to the level of text.

My approach performs significantly better than the state-of-the-art graph-theoretic algorithm, as the experimental evaluation on large real-world data sets demonstrates. Moreover, I have performed a large-scale analysis on text density on the Web and found that the corresponding distribution can be described very accurately (99.8%) by a simple Beta-Gaussian mixture model; this structure corroborates recent findings from the field of Quantitative Linguistics.

Finally, through an extensive evaluation using several different shallow text features, I show that using a simple decision tree classifier with only two feature classes (*text density + link density* or rather *number of words + link density*), we can achieve a detection quality that is superior to state-of-the-art techniques. At last, I derive a simple, plausible and well-fitting stochastic model for describing the boilerplate-aware text creation process (98.8%), based upon a simple Random Writer model.

5.2 Stratified Random User Models

The two problems, posed as questions in Section 1.2, can be approached from the perspective of "random user" models, both for link structure as well as for text structure. The solutions that have been presented in this thesis are heavily influenced by these models. The very high goodness of the presented models and the efficient optimizations affirm this.

5.2.1 Random Surfer Model

Imagine a random surfer model that is simplified to a state machine with only two states, whereas the states do not refer to individual pages but to the two different types of user behavior, browsing (B) and jumping (J), as depicted in Figure 5.1. Initially, the user *jumps (J)* into the graph onto a random web page and decides to either jump directly to another page with some probability $P_J(J)$ or decides to follow any link on the page with $P_J(B) = 1 - P_J(J)$. She then picks a link on the page with probability $P_B(B)$ and *browses* to that page. The user may now stay perpetually in state B (browsing), but at some point she gets bored and again jumps to another page; this occurs with probability $P_B(J)$.

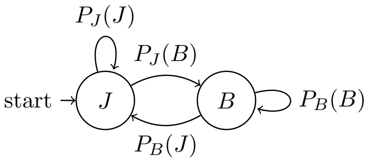


Figure 5.1: Random Surfer Model (simplified)

In the model, jumping is independent of any previous state, such that $P_J(J) = P_B(J) = 1 - \alpha = P(J)$. Which page actually is jumped to is solely determined by $\vec{\tau} = \vec{1}$ (unbiased PageRank) or by $\vec{\pi_B}$ (biased/personalized PageRank) respectively. Thus, also, the decision to actually browse $P_J(B) = P_B(B) = \alpha = P(B)$ is constant, which means it is only influenced by the global importance of a particular page (i.e., the inner, undepicted states of B) as defined by the graph's overall link structure L.

Since PageRank could then only be seen as an "over-complication" of in-degree and out-degree [46], we may conclude that PageRank's surfer model only is useful when deviating from the simple case, i.e. when using biased jumps for personalized PageRank. In Section 3.3 I have shown that we can indeed compute significantly different page vectors using topic-specific PageRank even for deeply-nested sub-topics (instead of only the top categories as in [54]), which then obviously differ from the pure in-degrees, and that these differences can be used to identifying topics in a set of pages (Section 3.4).

To reduce the time to compute these vectors, I used the approach presented in Section 3.2. I have shown that the PageRank computation can efficiently be parallelized when separating the computation of two different types of links, internal ones (intra-host) and external ones (inter-host). Indeed, we may argue that the internal links usually serve a completely different purpose than external ones. Assuming that the user can easily differentiate between these two types of links by their appearance, the decision to follow a specific type of link may in fact be made upstream to the actual browsing process, and thus should be modeled as depicted in Figure 5.2.

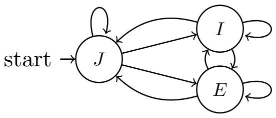
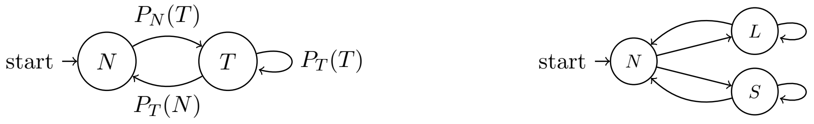


Figure 5.2: Stratified Random Surfer Model

After jumping to a page, the user decides either to browse locally (navigate in the structure of the current website) with $P_J(I) = P_I(I) = \alpha \cdot \beta = P(I)$ or to leave to another site with $P_J(E) = P_E(E) = \alpha{\cdot}\gamma = P(E)$; the probabilities to switch from browsing locally to external links (and vice versa) or to randomly jump thus are $P_E(I) = \alpha \cdot (1 - \gamma)$, $P_I(E) = \alpha \cdot (1 - \beta)$, $P_E(J) = (1 - \alpha) \cdot (1 - \gamma)$ and $P_I(J) = (1-\alpha)\cdot(1-\beta)$ respectively. Even though in PageRank we have $\beta = \gamma = 0.5$, i.e., there is no actual difference in *scoring* these two types of behavior, we can still utilize the disparate distribution of internal links versus external ones for reducing the computation time by orders of magnitudes, as shown in Equation 3.10.

5.2.2 Random Writer Model

In Section 4.4.4 we have observed that similar models can also be successfully applied to *text structure*:



(a) Simple

(b) Stratified

Figure 5.3: Random Writer Models

We all know that the creation of text is not conveyed in a random fashion like a Bernoulli trial. But given the plethora of documents from innumerable authors, it is a good approximation (see Equations 4.20 and 4.21). What I have shown that indeed is more relevant to the search quality than finding a particular unimodal distribution, is the fact that again a simple, stratified model can be utilized for classification purposes. That is, for the purpose of improving search quality it is less relevant whether we can model a 1-displaced Geometric or a Beta distribution, but rather whether and how we can separate the different strata from each other.

I have shown that the membership of a particular textual unit (in my case: a text block) to one or the other stratum can be used for Web page segmentation (Section 4.2) and that this fact is connected to fundamental laws of Quantitative Linguistics, namely Frumkina's Law, the Menzerath-Altmann Law and Zipf's law (Section 4.3).

Through a carefully conveyed analysis of different shallow text features (Section 4.4) on representative datasets, I finally come to the conclusion that the reason why simple classifiers that are just based on word counts perform so well for boilerplate detection and improving top-k search is that again, just like for the Stratified Random *Surfer* Model discussed above, the user makes decisions upstream to the actual writing process, and this separation needs to be taken into account for any "representative" model of text generation. We might thus call the stratified model a good candidate for a new statistical text law:

> When composing written text, the writer chooses, for each text block (segment, paragraph), from two different types of text, *short text* and *long text.* These types of text significantly differ in the average number of words, intended perception and indirectly also in writing style.
>
> $$\begin{aligned} Pr(Y = x) &= P(\text{short}) \cdot P(x|\text{short}) + \\ + \left[P(\text{long}) = 1 - P(\text{short})\right] &\cdot P(x|\text{long}) \\ E\left(P(x|\text{short})\right) &\ll E\left(P(x|\text{long})\right) \end{aligned} \tag{5.1}$$
>
> The choice depends to a large extent on the types of the surrounding text blocks (short/long), and deviations from the expected choice indicate a segmentation boundary. As a first approximation, the probability of writing a particular number of words (in short or long text) can be modeled as a 1-displaced Geometric distribution.

So far, there is nothing to be said against it.

5.3 Future Work

The presented results raise research issues in many different directions. Obviously, for boilerplate detection we need to remove the "last 5%" of classification error. Even though the generality of my approach might suggest that it is universally applicable, we need to test that on other content domains and languages.[1]

As I have shown that for both, link and text structure specific stratified models can be applied, we need to further analyze the relationship between the two types of links (intra-/extra-site, presumably navigational/informational) and the two types of text (short/long, presumably navigational/informational as well), for example: To what extent does intra-site linking correlate with short/navigational text, and can we use

[1] Besides English and German, users of my algorithms reported good success with Dutch, Turkish and Spanish websites, see comments at http://code.google.com/p/boilerpipe/wiki/FAQ and http://lingpipe-blog.com/2010/01/06/blegging-for-help-web-scraping-for-content/. Preliminary tests on Chinese texts are promising as well.

the knowledge about short and long text to further improve topic-specific PageRank by separating links contained in each of these types of text? Knowing the model describing the underlying human behavior when compositing Web text, can we use the approach to identify automatically generated spam pages? To what extent can we classify a Web page without deeply inspecting the text (only using link structure and the text's densitometric fingerprint)?

Moreover, we need to deeper investigate and extend the textual model from a linguistic perspective, e.g. is the differentiation between short and long text only necessary for the Web? Where can we observe short/long text strata outside in the "offline" world? How do semantics differ between blocks of short and long text? How does vocabulary richness and growth differ between short and long text? Will higher-level Markov models (i.e., not only inspecting the previous and next block) improve the classifier? And, lastly, how much better will our classifiers be on a particular language when taking word and segment frequencies into account?

To open up further possibilities, a re-implementation of the presented boilerplate detection algorithms, called "Boilerpipe", is available as Open Source from the Google Code project website[2]; the corresponding annotated Web page collection (L3S-GN1) is available from the author's website.[3]

Since July 2010, Boilerpipe is also part of Apache Tika[4], the Open Source content and metadata extraction framework, which certainly helps attracting users and researchers and will lead to new problems and solutions.

[2] http://code.google.com/p/boilerpipe/

[3] http://www.L3S.de/~kohlschuetter/boilerplate/

[4] http://tika.apache.org/

Curriculum Vitae

Christian Kohlschütter, born October 26th 1979, in Hof (Bavaria), Germany, son of Dr. iur. Hans Kohlschütter and Rita Kohlschütter. Married, two children.

2004-2010	Researcher at L3S Research Center Leibniz Universität Hannover
2003-2004	Researcher at RRZN Computing Center Leibniz Universität Hannover
1999-2003	Studies of Computer Science/Software Engineering Diplom-Informatiker (FH), FH Hof
Since 2000	Member of the German National Merit Foundation (Studienstiftung des deutschen Volkes) Scholarship 2000-2003
1999	General Qualification for University Entrance (Abitur) Jean-Paul-Gymnasium Hof
Since 1997	Freelance Software Developer and Consultant http://www.kohlschutter.com

Bibliography

[1] Asma Al-Saleh and Ali El-Zaart. Unsupervised Learning Technique for Skin Images Segmentation Using a Mixture of Beta Distributions. In *IFMBE Proceedings, 3rd Kuala Lumpur International Conference on Biomedical Engineering 2006 (Biomed 2006)*, pages 304–307, 2007.

[2] Gabriel Altmann. Verteilungen der Satzlängen (Distribution of Sentence Lengths). In K.-P. Schulz, editor, *Glottometrika 9*. Brockmeyer, 1988.

[3] Gabriel Altmann. Das Problem der Datenhomogenität. In *Glottometrika 13*, pages 287–298. Brockmeyer, Bochum, 1992.

[4] Gabriel Altmann. *Quantitative Linguistics - An International Handbook*, chapter Diversification processes. de Gruyter, 2005.

[5] Gabriel Altmann and Violetta Burdinski. Towards a Law of Word Repetitions in Text-Blocks. In U. Strauss W. Lehfeldt, editor, *Glottometrika 4*, volume 14 of *Quantitative Linguistics*, pages 147–167, Bochum, 1982. Brockmeyer.

[6] Aris Anagnostopoulos, Andrei Z. Broder, and David Carmel. Sampling search-engine results. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 245–256, New York, NY, USA, 2005. ACM. ISBN 1-59593-046-9. doi: http://doi.acm.org/10.1145/1060745.1060784.

[7] Apostolos Antonacopoulos, Basilios Gatos, and David Bridson. Page segmentation competition. *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on*, 2:1279–1283, 23-26 Sept. 2007. ISSN 1520-5363. doi: 10.1109/ICDAR.2007.4377121.

[8] Arvind Arasu, Jasmine Novak, Andrew Tomkins, and John Tomlin. PageRank Computation and the Structure of the Web: Experiments and Algorithms, 2001.

[9] Ricardo A. Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999. ISBN 020139829X.

[10] Shumeet Baluja. Browsing on Small Screens: Recasting Web-Page Segmentation into an Efficient Machine Learning Framework. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 33–42, New York, NY, USA, 2006. ACM. ISBN 1-59593-323-9. doi: http://doi.acm.org/10.1145/1135777.1135788.

[11] Ziv Bar-Yossef and Sridhar Rajagopalan. Template detection via data mining and its applications. In *WWW '02: Proceedings of the 11th international conference on World Wide Web*, pages 580–591, New York, NY, USA, 2002. ACM. ISBN 1-58113-449-5. doi: http://doi.acm.org/10.1145/511446.511522.

[12] Marco Baroni, Francis Chantree, Adam Kilgarriff, and Serge Sharoff. Cleaneval: a competition for cleaning web pages. In Nicoletta Calzolari, Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odjik, Stelios Piperidis, and Daniel Tapias, editors, *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, 2008. ISBN 2-9517408-4-0.

[13] Ori Ben-Yitzhak, Nadav Golbandi, Nadav Har'El, Ronny Lempel, Andreas Neumann, Shila Ofek-Koifman, Dafna Sheinwald, Eugene Shekita, Benjamin Sznajder, and Sivan Yogev. Beyond basic faceted search. In *WSDM '08: Proceedings of the international conference on Web search and web data mining*, pages 33–44, New York, NY, USA, 2008. ACM. ISBN 978-1-59593-927-9. doi: http://doi.acm.org/10.1145/1341531.1341539.

[14] Tim Berners-Lee and Dan Conolly. RFC 1866: Hypertext Markup Language - 2.0, November 1995.

[15] Karl-Heinz Best. *Quantitative Linguistics - An International Handbook*, chapter Satzlänge (Sentence length), pages 298–304. de Gruyter, 2005.

[16] Karl-Heinz Best. Sprachliche Einheiten in Textblöcken. In *Glottometrics 9*, pages 1–12. RAM Verlag, Lüdenscheid, 2005.

[17] Krishna Bharat, Bay-Wei Chang, Monika Rauch Henzinger, and Matthias Ruhl. Who links to whom: Mining linkage between web sites. In *Proc. of the IEEE Intl. Conf. on Data Mining*, pages 51–58, 2001. ISBN 0-7695-1119-8.

[18] Sergey Brin, Rajeev Motwani, Lawrence Page, and Terry Winograd. What can you do with a web in your pocket? *Data Engineering Bulletin*, 21(2):37–47, 1998. URL `citeseer.ist.psu.edu/brin98what.html`.

[19] Andrei Broder, Ravi Kumar, Farzin Maghoul, Prabhakar Raghavan, Sridhar Rajagopalan, Raymie Stata, Andrew Tomkins, and Janet Wiener. Graph structure in the web. In *Proc. of the 9th international World Wide Web conference*, pages 309–320. North-Holland Publishing Co., 2000. doi: http://dx.doi.org/10.1016/S1389-1286(00)00083-9. URL `http://www9.org/w9cdrom/160/160.html`.

[20] Andrei Z. Broder, Ronny Lempel, Farzin Maghoul, and Jan Pedersen. Efficient PageRank Approximation via Graph Aggregation. In *Proc. of the 13th International World Wide Web Conference*, pages 484–485, 2004. ISBN 1-58113-912-8.

[21] Michael K. Buckland. What is a “document”? *Journal of the American Society for Information Science*, 48:804–809, September 1997.

[22] Deng Cai, Shipeng Yu, Ji-Rong Wen, and Wei-Ying Ma. Extracting Content Structure for Web Pages Based on Visual Representation. In X. Zhou, Y. Zhang, and M. E. Orlowska, editors, *APWeb*, volume 2642 of *LNCS*, pages 406–417. Springer, 2003. ISBN 3-540-02354-2.

[23] Deng Cai, Shipeng Yu, Ji-Rong Wen, and Wei-Ying Ma. Block-based web search. In *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR*

conference on Research and development in information retrieval, pages 456–463, New York, NY, USA, 2004. ACM. ISBN 1-58113-881-4. doi: http://doi.acm.org/10.1145/1008992.1009070.

[24] Deepayan Chakrabarti, Ravi Kumar, and Kunal Punera. Page-level Template Detection via Isotonic Smoothing. In *WWW '07: Proc. of the 16th int. conf. on World Wide Web*, pages 61–70, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-654-7. doi: http://doi.acm.org/10.1145/1242572.1242582.

[25] Deepayan Chakrabarti, Ravi Kumar, and Kunal Punera. A graph-theoretic approach to webpage segmentation. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 377–386, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-085-2. doi: http://doi.acm.org/10.1145/1367497.1367549.

[26] Ming Chen, Xiaoqing Ding, and Jian Liang. Analysis, understanding and representation of chinese newspaper with complex layout. *Image Processing, 2000. Proceedings. 2000 International Conference on*, 2:590–593 vol.2, 2000. doi: 10.1109/ICIP.2000.899500.

[27] Yen-Yu Chen, Qingqing Gan, and Torsten Suel. I/O-efficient Techniques for Computing PageRank. In *CIKM '02: Proceedings of the eleventh international conference on Information and knowledge management*, pages 549–557, New York, NY, USA, 2002. ACM. ISBN 1-58113-492-4. doi: http://doi.acm.org/10.1145/584792.584882.

[28] Yu Chen, Wei-Ying Ma, and Hong-Jiang Zhang. Detecting web page structure for adaptive viewing on small form factor devices. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 225–233, New York, NY, USA, 2003. ACM. ISBN 1-58113-680-3. doi: http://doi.acm.org/10.1145/775152.775184.

[29] Paul Alexandru Chirita, Wolfgang Nejdl, Raluca Paiu, and Christian Kohlschütter. Using ODP metadata to personalize search. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 178–185, New York, NY, USA, 2005. ACM. ISBN 1-59593-034-5. doi: http://doi.acm.org/10.1145/1076034.1076067.

[30] Junghoo Cho and Hector Garcia-Molina. The evolution of the web and implications for an incremental crawler. In *Proceedings of the 26th International Conference on Very Large Databases*, 2000. URL `citeseer.ist.psu.edu/cho00evolution.html`.

[31] Bruce Croft, Donald Metzler, and Trevor Strohman. *Search Engines: Information Retrieval in Practice*. Addison-Wesley Publishing Company, USA, 2009. ISBN 0136072240, 9780136072249.

[32] Jeffrey Dean and Monika R. Henzinger. Finding related pages in the World Wide Web. *Computer Networks (Amsterdam, Netherlands)*, 31(11–16):1467–1479, 1999. URL `citeseer.ist.psu.edu/dean99finding.html`.

[33] Sandip Debnath, Prasenjit Mitra, Nirmal Pal, and C. Lee Giles. Automatic identification of informative sections of web pages. *IEEE Trans. on Knowledge and Data Engineering*, 17(9):1233–1246, 2005. ISSN 1041-4347. doi: http://doi.ieeecomputersociety.org/10.1109/TKDE.2005.138.

[34] Lukasz Debowski. Zipf's law against the text size: a half-rational model. In *Glottometrics 4*, pages 49–60. RAM Verlag, Lüdenscheid, 2002.

[35] William Denton. How to make a faceted classification and put it on the web. `http://www.miskatonic.org/library/facet-web-howto.pdf`, November 2003.

[36] William Denton. Putting facets on the web: An annotated bibliography. `http://www.miskatonic.org/library/facet-biblio.html`, October 2003.

[37] Nadav Eiron, Kevin S. McCurley, and John A. Tomlin. Ranking the web frontier. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 309–318, New York, NY, USA, 2004. ACM. ISBN 1-58113-844-X. doi: http://doi.acm.org/10.1145/988672.988714.

[38] Mehmet S. Aktas et al. Personalizing pagerank based on domain profiles. In *WEBKDD'04, Seattle, USA*, pages 83–90, August 2004. URL `citeseer.ist.psu.edu/708503.html`.

[39] Pavel Calado et al. Link-based similarity measures for the classification of web documents. *J. Am. Soc. Inf. Sci. Technol.*, 57(2):208–221, 2006. ISSN 1532-2882. doi: http://dx.doi.org/10.1002/asi.v57:2.

[40] Soumen Chakrabarti et al. Enhanced hypertext categorization using hyperlinks. In *SIGMOD '98*, pages 307–318, New York, NY, US, 1998. ACM Press. ISBN 0-89791-995-5. doi: http://doi.acm.org/10.1145/276304.276332.

[41] Stefan Evert. A lightweight and efficient tool for cleaning web pages. In Nicoletta Calzolari, Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odjik, Stelios Piperidis, and Daniel Tapias, editors, *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, may 2008. European Language Resources Association (ELRA). ISBN 2-9517408-4-0. http://www.lrec-conf.org/proceedings/lrec2008/.

[42] Fariza Fauzi, Jer-Lang Hong, and Mohammed Belkhatir. Webpage segmentation for extracting images and their surrounding contextual information. In *MM '09: Proceedings of the seventeen ACM international conference on Multimedia*, pages 649–652, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-608-3. doi: http://doi.acm.org/10.1145/1631272.1631379.

[43] David Fernandes, Edleno S. de Moura, Berthier Ribeiro-Neto, Altigran S. da Silva, and Marcos André Gonçalves. Computing block importance for searching on web sites. In *CIKM '07*, pages 165–174, 2007. ISBN 978-1-59593-803-9. doi: http://doi.acm.org/10.1145/1321440.1321466.

[44] Adriano Ferraresi, Eros Zanchetta, Marco Baroni, and Silvia Bernardini. Introducing and evaluating ukWaC, a very large Web-derived corpus of English. In *Proceedings of the WAC4 Workshop at LREC 2008.*

[45] Aidan Finn, Nicholas Kushmerick, and Barry Smyth. Fact or fiction: Content classification for digital libraries. Joint DELOS-NSF Workshop on Personalisation and Recommender Systems in Digital Libraries (Dublin), 2001.

[46] Santo Fortunato, Marián Bogu ná, Alessandro Flammini, and Filippo Menczer. Approximating pagerank from in-degree. pages 59–71, 2008. doi: http://dx.doi.org/10.1007/978-3-540-78808-9_6.

[47] David Gibson, Kunal Punera, and Andrew Tomkins. The volume and evolution of web page templates. In *WWW '05: Special interest tracks and posters of the 14th international conference on World Wide Web*, pages 830–839, New York, NY, USA, 2005. ACM. ISBN 1-59593-051-5. doi: http://doi.acm.org/10.1145/1062745.1062763.

[48] John Gibson, Ben Wellner, and Susan Lubar. Adaptive web-page content identification. In *WIDM '07: Proceedings of the 9th annual ACM international workshop on Web information and data management*, pages 105–112, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-829-9. doi: http://doi.acm.org/10.1145/1316902.1316920.

[49] David Gleich, Leonid Zhukov, and Pavel Berkhin. Fast parallel PageRank: A linear system approach. Technical report, Yahoo! Research Labs, 2004. URL `http://research.yahoo.com/publications/38.pdf`.

[50] Scott Golder and Bernardo A. Huberman. The structure of collaborative tagging systems. Technical report, Information Dynamics Lab, HP Labs, 2005. URL `http://www.isrl.uiuc.edu/~amag/langev/paper/golder05taggingSystems.html`.

[51] Peter Grzybek, editor. *Contributions to the Science of Text and Language.* Springer, 2006.

[52] Peter Grzybek. On the systematic and system-based study of grapheme frequencies - a re-analysis of german letter frequencies. In G. Altmann, K.-H. Best, and P. Grzybek et al., editors, *Glottometrics 15*, pages 82–91. RAM Verlag, Lüdenscheid, 2007.

[53] Taher H. Haveliwala. Efficient computation of PageRank. Technical Report 1999-31, Stanford Library Technologies Project, 1999. URL `citeseer.ist.psu.edu/haveliwala99efficient.html`.

[54] Taher H. Haveliwala. Topic-sensitive PageRank. In *Proc. of the eleventh International Conference on World Wide Web*, pages 517–526. ACM Press, 2002. ISBN 1-58113-449-5. doi: http://doi.acm.org/10.1145/511446.511513.

[55] Taher H. Haveliwala et al. 2001 Crawl of the WebBase project, 2001. URL `http://dbpubs.stanford.edu:8091/~testbed/doc2/WebBase/`.

[56] Marti A. Hearst. Multi-paragraph segmentation of expository text. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, pages 9–16, Morristown, NJ, USA, 1994. Association for Computational Linguistics. doi: http://dx.doi.org/10.3115/981732.981734.

[57] Marti A. Hearst. Clustering versus faceted categories for information exploration. *Commun. ACM*, 49(4):59–61, 2006.

[58] Marti A. Hearst, Ame Elliott, Jennifer English, Rashmi R. Sinha, Kirsten Swearingen, and Ka-Ping Yee. Finding the flow in web site search. *Commun. of the ACM*, 45(9):42–49, 2002.

[59] James Hendler, Nigel Shadbolt, Wendy Hall, Tim Berners-Lee, and Daniel Weitzner. Web science: an interdisciplinary approach to understanding the web. *Commun. ACM*, 51(7):60–69, 2008. ISSN 0001-0782. doi: http://doi.acm.org/10.1145/1364782.1364798.

[60] Katja Hofmann and Wouter Weerkamp. Web Corpus Cleaning using Content and Structure. In *Building and Exploring Web Corpora*, pages 145–154. UCL Presses Universitaires de Louvain, September 2007.

[61] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218, December 1985. URL `http://ideas.repec.org/a/spr/jclass/v2y1985i1p193-218.html`.

[62] Glen Jeh and Jennifer Widom. Scaling personalized web search. In *WWW '03*, pages 271–279, New York, NY, USA, 2003. ISBN 1-58113-680-3. doi: http://doi.acm.org/10.1145/775152.775191.

[63] Maryam Kamvar, Melanie Kellar, Rajan Patel, and Ya Xu. Computers and iphones and mobile phones, oh my!: a logs-based comparison of search users on different devices. In *WWW '09: Proceedings of the 18th international conference on World wide web*, pages 801–810, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-487-4. doi: http://doi.acm.org/10.1145/1526709.1526817.

[64] Sepandar Kamvar, Taher Haveliwala, Christopher Manning, and Gene Golub. Exploiting the block structure of the web for computing PageRank. Technical report, Stanford University, 2003. URL `citeseer.ist.psu.edu/article/kamvar03exploiting.html`.

[65] Sepandar D. Kamvar, Taher H. Haveliwala, and Gene H. Golub. Adaptive methods for the computation of PageRank. Technical report, Stanford University, 2003. URL `citeseer.ist.psu.edu/kamvar03adaptive.html`.

[66] Sepandar D. Kamvar, Taher H. Haveliwala, Christopher D. Manning, and Gene H. Golub. Extrapolation methods for accelerating PageRank computations. In *Proc. of the 12th Intl. Conf. on the World Wide Web*, pages 261–270, 2003. ISBN 1-58113-680-3.

[67] Hung-Yu Kao, Jan-Ming Ho, and Ming-Syan Chen. Wisdom: Web intrapage informative structure mining based on document object model. *Knowledge and Data Engineering, IEEE Transactions on*, 17(5):614–627, May 2005. ISSN 1041-4347. doi: 10.1109/TKDE.2005.84.

[68] Maurice G. Kendall. *Rank Correlation Methods*. Hafner, New York, USA, 1955.

[69] Sung Jin Kim and Sang Ho Lee. An improved computation of the PageRank algorithm. In *Proc. of the European Conference on Information Retrieval (ECIR)*, pages 73–85, 2002. URL `citeseer.ist.psu.edu/kim02improved.html`.

[70] David P. Koester, Sanjay Ranka, and Geoffrey C. Fox. A parallel gauss-seidel algorithm for sparse power system matrices. In *Supercomputing '94: Proceedings of the 1994 ACM/IEEE conference on Supercomputing*, pages 184–193, New York, NY, USA, 1994. ACM. ISBN 0-8186-6605-6. doi: http://doi.acm.org/10.1145/602770.602806.

[71] Reinhard Köhler. Elemente der synergetischen Linguistik. In *Glottometrika 12*, pages 179–187. Brockmeyer, Bochum, 1990.

[72] Reinhard Köhler. Synergetic linguistics. In *Quantitative Linguistics – An International Handbook*, pages 760–774. de Gruyter, 2005.

[73] Christian Kohlschütter. A Densitometric Classification of Web Template Content. In Emmerich Kelih, Viktor Levickij, and Gabriel Altmann, editors, *Methods of Text Analysis: Omnibus volume*, pages 133–155. Chernivtsi: CNU, 2009.

[74] Christian Kohlschütter. A Densitometric Analysis of Web Template Content. In *WWW '09: Proceedings of the 18th International World Wide Web Conference*, pages 1165–1166, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-487-4. doi: http://doi.acm.org/10.1145/1526709.1526909.

[75] Christian Kohlschütter and Wolfgang Nejdl. A Densitometric Approach to Web Page Segmentation. In *CIKM '08: Proceedings of the 17th ACM conference on Information and Knowledge Management*, pages 1173–1182, New York, NY, USA, 2008. ACM. ISBN 978-1-59593-991-3. doi: http://doi.acm.org/10.1145/1458082.1458237.

[76] Christian Kohlschütter, Paul-Alexandru Chirita, and Wolfgang Nejdl. Using Link Analysis to Identify Aspects in Faceted Web Search. In *SIGIR'2006 Workshop on Faceted Search*, Seattle, WA, USA, August 2006.

[77] Christian Kohlschütter, Paul-Alexandru Chirita, and Wolfgang Nejdl. Efficient parallel computation of pagerank. In *ECIR 2006: Advances in Information Retrieval 2006: 28th European Conference on IR Research*, volume LNCS 3936, London, UK, April 2006.

[78] Christian Kohlschütter, Paul-Alexandru Chirita, and Wolfgang Nejdl. Utility analysis for topically biased PageRank. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 1211–1212, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-654-7. doi: http://doi.acm.org/10.1145/1242572.1242770.

[79] Christian Kohlschütter, Peter Fankhauser, and Wolfgang Nejdl. Boilerplate detection using shallow text features. In *WSDM '10: Proceedings of the third ACM International Conference on Web search and Data Mining*, pages 441–450, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-889-6. doi: http://doi.acm.org/10.1145/1718487.1718542.

[80] Amy N. Langville and Carl D. Meyer. Deeper inside PageRank. *Internet Mathematics*, 1(3):335–380, 2004.

[81] Daniel Lavalette. A general purpose ranking variable with applications to various ranking laws. In Peter Grzybek and Reinhard Köhler, editors, *Exact Methods in the Study of Language and Text*, pages 371–382. de Gruyter, 2007.

[82] Chris P. Lee, Gene H. Golub, and Stefanos A. Zenios. A fast two-stage algorithm for computing PageRank. Technical report, Stanford University, 2003.

[83] Sonya Liberman and Ronny Lempel. Approximately optimal facet selection. In *The 4th Workshop on the Future of Web Search*, April 2009. URL `http://research.yahoo.com/files/facets_ysite.pdf`.

[84] Dmitry Lizorkin, Pavel Velikhov, Maxim Grinev, and Denis Turdakov. Accuracy estimate and optimization techniques for simrank computation. *The VLDB Journal*, 19(1):45–66, 2010. ISSN 1066-8888. doi: http://dx.doi.org/10.1007/s00778-009-0168-8.

[85] Qing Lu and Lise Getoor. Link-based text classification. Text-Mining & Link-Analysis Workshop TextLink 2003, 2003.

[86] Bundit Manaskasemsak and Arnon Rungsawang. Parallel PageRank computation on a gigabit pc cluster. In *Proc. of the 18th International Conference on Advanced Information Networking and Application (AINA'04)*, 2004.

[87] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008. ISBN 0521865719, 9780521865715.

[88] Adam Mathes. Folksonomies - cooperative classification and communication through shared metadata. Computer Mediated Communication - LIS590CMC, Graduate School of Library and Information Science, University of Illinois Urbana-Champagin, December 2004.

[89] Frank McSherry. A Uniform Approach to Accelerated PageRank Computation. In *Proceedings of the 14th international World Wide Web Conference*, pages 575–582, New York, NY, USA, 2005. ACM Press. ISBN 1-59593-046-9. doi: http://doi.acm.org/10.1145/1060745.1060829.

[90] Sundaresan Naranan and Viddhachalam K. Balasubrahmanyan. Power laws in statistical linguistics and related systems. In *Quantitative Linguistics – An International Handbook*, pages 716–738. de Gruyter, 2005.

[91] Iadh Ounis, Craig Macdonald, Maarten de Rijke, Gilad Mishne, and Ian Soboroff. Overview of the trec 2006 blog track. In Ellen M. Voorhees and Lori P. Buckland, editors, *TREC*, volume Special Publication 500-272. National Institute of Standards and Technology (NIST), 2006.

[92] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998. URL `citeseer.ist.psu.edu/page98pagerank.html`.

[93] Jeff Pasternack and Dan Roth. Extracting article text from the web with maximum subsequence segmentation. In *WWW '09: Proceedings of the 18th international conference on World wide web*, pages 971–980, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-487-4. doi: http://doi.acm.org/10.1145/1526709.1526840.

[94] Ioan-Iovitz Popescu. On a Zipf's Law Extension to Impact Factors. In *Glottometrics 6*. RAM Verlag, Lüdenscheid, 2003.

[95] Ioan-Iovitz Popescu and Gabriel Altmann. Some aspects of word frequencies. In *Glottometrics 13*, pages 23–46. RAM Verlag, Lüdenscheid, 2006.

[96] Feng Qiu and Junghoo Cho. Automatic identification of user interest for personalized search. In *Proc. of the 15th international World Wide Web conference*, 2006.

[97] Davood Rafiei, Krishna Bharat, and Anand Shukla. Diversifying web search results. In *WWW '10: Proceedings of the 19th international conference on World wide web*, pages 781–790, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-799-8. doi: http://doi.acm.org/10.1145/1772690.1772770.

[98] Stephen Robertson. Understanding inverse document frequency: On theoretical arguments for idf. *Journal of Documentation*, 60:2004, 2004.

[99] Stephen Robertson, Steve Walker, Susan Jones, Micheline M. Hancock-Beaulieu, and Mike Gatford. Okapi at trec-3. pages 109–126, 1996.

[100] Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., 1986. ISBN 0070544840.

[101] Gerard Salton, Anita Wong, and Chung-Shu Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975. ISSN 0001-0782. doi: http://doi.acm.org/10.1145/361219.361220.

[102] Gerard Salton, Edward A. Fox, and Harry Wu. Extended boolean information retrieval. *Commun. ACM*, 26(11):1022–1036, 1983. ISSN 0001-0782. doi: http://doi.acm.org/10.1145/182.358466.

[103] Karthikeyan Sankaralingam, Simha Sethumadhavan, and James C. Browne. Distributed Pagerank for P2P Systems. In *Proc. of the 12th IEEE Intl. Symp. on High Performance Distributed Computing (HPDC)*, page 58, 2003. ISBN 0-7695-1965-2.

[104] Tamas Sarlos, Andras A. Benczur, Karoly Csalogany, Daniel Fogaras, and Balazs Racz. To randomize or not to randomize: Space optimal summaries for hyperlink analysis. In *Proc. of the 15th international World Wide Web conference*, 2006.

[105] Claude E. Shannon. A Mathematical Theory of Communication. *The Bell System Technical Journal*, 27:379–423, 623–656, October 1948. URL `http://cm.bell-labs.com/cm/ms/what/shannonday/paper.html`.

[106] Shu-Ming Shi, Jin Yu, Guang-Wen Yang, and Ding-Xing Wang. Distributed Page Ranking in Structured P2P Networks. In *Proc. of the 2003 International Conference on Parallel Processing (ICPP'03)*, pages 179–186, 2003.

[107] Amit Singhal and Marcin Kaszkiel. A case study in web search using TREC algorithms. In *WWW '01: Proceedings of the 10th international conference on World Wide Web*, pages 708–716, New York, NY, USA, 2001. ACM. ISBN 1-58113-348-0. doi: http://doi.acm.org/10.1145/371920.372186.

[108] Jared M. Spool, Tara Scanlon, Carolyn Snyder, Will Schroeder, and Terri DeAngelo. *Web site usability: a designer's guide*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999. ISBN 1-55860-569-X.

[109] Miroslav Spousta, Michael Marek, and Pavel Pecina. Victor: the web-page cleaning tool. In *WaC4*, 2008.

[110] George Stockman and Linda G. Shapiro. *Computer Vision*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2001. ISBN 0130307963.

[111] Alexander Strehl and Joydeep Ghosh. Cluster ensembles — a knowledge reuse framework for combining multiple partitions. *J. Mach. Learn. Res.*, 3:583–617, 2003. ISSN 1533-7928.

[112] Juhan Tuldava. Stylistics, author identification. In *Quantitative Linguistics – An International Handbook*, pages 368–387. de Gruyter, 2005.

[113] Fiona J. Tweedie. Statistical models in stylistics and forensic linguistics. In *Quantitative Linguistics – An International Handbook*, pages 387–397. de Gruyter, 2005.

[114] Karane Vieira, Altigran S. da Silva, Nick Pinto, Edleno S. de Moura, Joao M. B. Cavalcanti, and Juliana Freire. A fast and robust method for web page template detection and removal. In *CIKM '06: Proceedings of the 15th ACM International Conference on Information and Knowledge Management*, pages 258–267, 2006. ISBN 1-59593-433-2. doi: http://doi.acm.org/10.1145/1183614.1183654.

[115] Relja Vulanovic and Reinhard Köhler. *Quantitative Linguistics - An international Handbook*, chapter Syntactic units and structures, pages 274–291. de Gruyter, 2005.

[116] Yuan Wang and David J. DeWitt. Computing PageRank in a distributed internet search system. In *Proceedings of the 30th VLDB Conference*, 2004.

[117] Gejza Wimmer and Gabriel Altmann. *Thesaurus of univariate discrete probability distributions*. Stamm Verlag, 1999.

[118] Gejza Wimmer and Gabriel Altmann. Unified derivation of some linguistic laws. In *Quantitative Linguistics – An International Handbook*, pages 791–807. de Gruyter, 2005.

[119] Alex Wright. Ready for a web os? *Commun. ACM*, 52(12):16–17, 2009. ISSN 0001-0782. doi: http://doi.acm.org/10.1145/1610252.1610260.

[120] Jie Wu and Karl Aberer. Using SiteRank for P2P Web Retrieval, March 2004. URL `citeseer.ist.psu.edu/wu04using.html`.

[121] Yiming Yang, Sean Slattery, and Rayid Ghani. A study of approaches to hypertext categorization. *Journal of Intelligent Information Systems*, 18(2-3): 219–241, 2002. URL `citeseer.ist.psu.edu/478602.html`.

[122] Ka-Ping Yee, Kirsten Swearingen, Kevin Li, and Marti A. Hearst. Faceted metadata for image search and browsing. In *Proc. of the SIGCHI Conf. on Human Factors in Computing Systems*, pages 401–408, 2003.

[123] Lan Yi, Bing Liu, and Xiaoli Li. Eliminating noisy information in web pages for data mining. In *KDD '03: Proc. of the 9th ACM SIGKDD int. conf. on Knowledge discovery and data mining*, pages 296–305, 2003. ISBN 1-58113-737-0. doi: http://doi.acm.org/10.1145/956750.956785.

[124] Yangbo Zhu, Shaozhi Ye, and Xing Li. Distributed pagerank computation based on iterative aggregation-disaggregation methods. In *Proc. of the 14th ACM international conference on Information and knowledge management*, 2005.

[125] George K. Zipf. *Human Behavior and the Principle of Least Effort.* Addison-Wesley, Reading, 1949.