Technologies for Reusing Text from the Web

From the
Faculty of Media
of the
Bauhaus-Universität Weimar
Germany

The Accepted Dissertation of
Martin Potthast

To Obtain the Academic Degree of
Dr. rer. nat.

Advisor:	Prof. Dr. Benno M. Stein
Reviewer:	Paul D. Clough, PhD
Oral Exam:	December 16, 2011

Für meine Väter

Contents

Preface

Abstract

Texts from the web can be reused individually or in large quantities. The former is called text reuse and the latter language reuse. We first present a comprehensive overview of the different ways in which text and language is reused today, and how exactly information retrieval technologies can be applied in this respect. The remainder of the thesis then deals with specific retrieval tasks. In general, our contributions consist of models and algorithms, their evaluation, and for that purpose, large-scale corpus construction.

The thesis divides into two parts. The first part introduces technologies for text reuse detection, and our contributions are as follows: (1) A unified view of projecting-based and embedding-based fingerprinting for near-duplicate detection and the first time evaluation of fingerprint algorithms on Wikipedia revision histories as a new, large-scale corpus of near-duplicates. (2) A new retrieval model for the quantification of cross-language text similarity, which gets by without parallel corpora. We have evaluated the model in comparison to other models on many different pairs of languages. (3) An evaluation framework for text reuse and particularly plagiarism detectors, which consists of tailored detection performance measures and a large-scale corpus of automatically generated and manually written plagiarism cases. The latter have been obtained via crowd-

sourcing. This framework has been successfully applied to evaluate many different state-of-the-art plagiarism detection approaches within three international evaluation competitions.

The second part introduces technologies that solve three retrieval tasks based on language reuse, and our contributions are as follows: (4) A new model for the comparison of textual and non-textual web items across media, which exploits web comments as a source of information about the topic of an item. In this connection, we identify web comments as a largely neglected information source and introduce the rationale of comment retrieval. (5) Two new algorithms for query segmentation, which exploit web *n*-grams and Wikipedia as a means of discerning the user intent of a keyword query. Moreover, we crowdsource a new corpus for the evaluation of query segmentation which surpasses existing corpora by two orders of magnitude. (6) A new writing assistance tool called NETSPEAK, which is a search engine for commonly used language. NETSPEAK indexes the web in the form of web *n*-grams as a source of writing examples and implements a wildcard query processor on top of it.

Danksagung

Mein Dank gilt Prof. Dr. Benno Stein, unter dessen Anleitung diese Arbeit entstanden ist. Sein Blick für das Wesentliche und sein unstillbarer Wissensdurst waren und sind mir ein großes Vorbild. Ich danke auch den Mitgliedern seiner Arbeitsgruppe Maik Anderka, Steven Burrows, Tim Gollub, Matthias Hagen, Dennis Hoppe, Nedim Lipka, Sven Meyer zu Eißen und Peter Prettenhofer, die mir bei zahllosen Gelegenheiten tatkräftig und inspirierend zur Seite standen. Weiterhin danke ich Patrick Riehmann und Bernd Fröhlich sowie Alberto Barrón-Cedeño und Paolo Rosso für die fruchtbare interdisziplinäre und internationale Zusammenarbeit. Ich danke Paul Clough für die Begutachtung dieser Arbeit.

Ich möchte mich auch bei Dietmar Bratke, Jürgen Eismann, Nadin Glaser, Maria-Theresa Hansens, Melanie Hennig, Dana Horch, Antje Klahn, Hildegard Kühndorf, Tina Meinhardt, Christin Oehmichen und Ursula Schmidt bedanken, die fortwährend bei der Überwindung (verwaltungs)technischer Hürden halfen und Kontakt zur Öffentlichkeit herstellten.

Besonders möchte ich mich bei den mehr als 40 Studenten bedanken, die sich in Projekten und Abschlussarbeiten sowie als Hilfskräfte für meine Forschung begeistern konnten. Ohne sie wäre die Bandbreite der Themen dieser Arbeit nicht machbar gewesen. An dieser Stelle möchte ich vor allem Steffen Becker, Christof Bräutigam, Andreas Eiselt, Robert Gerling, Teresa Holfeld, Alexander Kümmel, Fabian Loose und nicht zuletzt Martin Trenkmann hervorheben, die sich weit über ihr planmäßiges Studium hinaus für mich engagiert und an zahlreichen meiner Veröffentlichungen beteiligt haben.

Abschließend danke ich meinen Familien und Freunden für ihre Unterstützung, insbesondere Stephan Bongartz, Daniel, Wiebke, Marc und Merle Potthast, Steffi, Leonie und Louisa Daniel, Gabi und Günter Aab, Georg Potthast und Hildegard Knoke, Ellinor Pfützner, sowie Martin Weitert, Daniel Warner und Christian Ederer.

Chapter 1

Introduction

To reuse something means to use it again after the first time. While this concept is ubiquitous in all our lives it is less well-known in connection with text—at least not by that name. Better known are things like quotations, translations, paraphrases, metaphrases, summaries, boilerplate text, and plagiarism, all of which can be grouped under the term "text reuse" (see Figure 1.1). Reusing text is an integral part of writing in many genres so that the above kinds of reuse are widespread. The extent to which text is reused today, however, is still largely unknown, which is partly due to a lack of tools to study the phenomenon at scale.

Large quantities of text can be found on the web, readily available for reuse. Besides reusing them individually, certain tasks can be accomplished by analyzing them as a whole, which is called language reuse. Herein lies the potential to uncover new forms of reuse since texts have countless relations with other web items. Hence, from a computer science perspective, we ask the following questions:

- How and to what extent can text reuse be detected?
- What tasks can be supported by language reuse from the web?

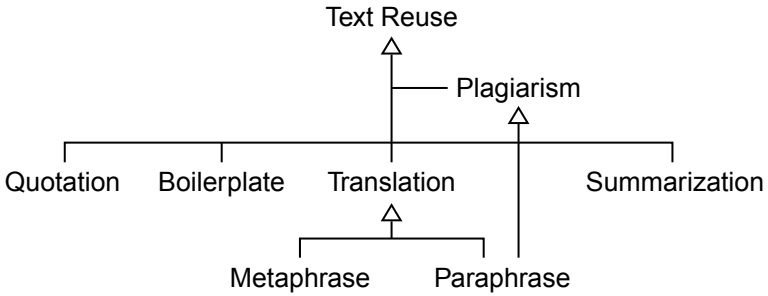


Figure 1.1: Taxonomy of the well-known forms of text reuse.

In this thesis, we tackle a selection of problems related to these questions using technologies from information retrieval. Beforehand, we give an overview of related research and technologies (Section 1.1), detail the contributions made (Section 1.2), and introduce the necessary information retrieval terminology (Section 1.3).

1.1 Related Research and Technologies

Text reuse is the reuse of individual texts, whereas language reuse is the reuse of large quantities of texts at the same time. For example, a translator reuses a text by translating it without using other texts, whereas a linguist reuses language by reviewing many occurrences of a word in different texts to better understand all its meanings. More generally, the reuse of a text happens linearly and independent of other texts, while language is reused by exploiting properties that many texts have in common. It is interesting to note that, for a human, reusing a single text in a sensible way is much easier than reusing large text collections, while for a computer, it is the other way around. The following two subsections overview research fields and technologies related to text reuse and language reuse, and to our research questions in particular.

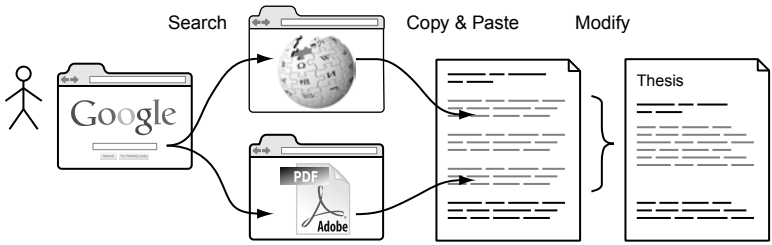


Figure 1.2: The basic steps of reusing a text from the web.

1.1.1 Text Reuse

Text reuse technologies can be divided into the categories support and detection.[1] Looking at the history of copying and printing, however, it becomes apparent that support technologies are far ahead: starting with the invention of the printing press, the costs of duplicating a text decreased dramatically so that nowadays every word processor supports instant text duplication via copy and paste commands. In combination with the increasing number of texts on the web, quick access to them via search engines, and the ongoing digitization efforts, texts can seemingly be reused at peak efficiency. Today, text reuse—and plagiarism in particular—often happens as follows (see Figure 1.2): first a web search for a suitable source is conducted, then text from that source is copied, and finally, the copied text is possibly modified. By contrast, detection technologies are still in their infancy. In what follows, the basic principles of detecting text reuse, and related research are reviewed.

[1] Note that detection technologies inadvertently support text reuse, for example, when they are used to modify a reused text until it cannot be detected anymore. This implies an arms race between reusing authors and reuse detectors. But as long as computers don't outsmart humans, the reusing authors will always win, though at significantly higher costs. Hence, if the intention of detecting text reuse is its prevention, a realistic goal is to raise the costs of getting away undetected up to a point at which most reusing authors consider writing a text themselves less laborious.

Detecting Text Reuse If one is given a document of uncertain originality, and the task of identifying all passages of reused text therein, there are three ways to accomplish this manually: (1) by searching for documents that may have served as original to the author of the suspicious document, (2) by checking whether the suspicious document has been written by its alleged author or someone else, and (3) by checking whether all passages of the suspicious document have been written by the same author. The first alternative attempts to retrace the steps of a reusing author depicted in Figure 1.2. The latter two are derived from the fact that humans are capable of recognizing authors by their writing style or changes thereof. Basically, however, all three alternatives boil down to comparing (passages of) the suspicious document one by one to others, looking for a "striking resemblance" in terms of syntax and semantics. If a document resembles the suspicious document in terms of certain syntactic characteristics that indicate writing style, they may have been written by the same author. If a document resembles the suspicious document in terms of semantics, it may be an original. Ideally, the suspicious document would be compared thus to all other documents available, but in practice, because of the efforts involved, one has to limit comparisons to a reasonable number of "candidate documents." Hence, these candidates must be chosen carefully in order to maximize the likelihood of finding the true originals, if there are any.

It is here where technology can help to scale up investigations, namely by automating the search for candidate documents, and by automating the comparisons. In [212] we have proposed a generic retrieval process to detect text reuse and plagiarism (see Figure 1.3). The first two steps of this process, candidate retrieval and detailed comparison, correspond to the two aforementioned possibilities of automation. The third step, knowledge-based post-processing, involves distinguishing different kinds of text reuse, filtering false detections, identifying citations, and visualizing modifications made on the reused text in order for a tool to be of further assistance.

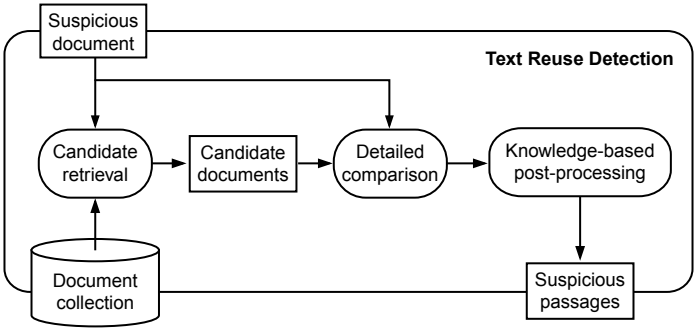


Figure 1.3: The basic steps of detecting reused text.

Related Research As a research field, text reuse has received but little attention. Among the first to conduct research on this topic have been researchers from the University of Sheffield in the course of a research project called Meter, a short for "Measuring Text Reuse."[2] Their primary focus was text reuse within the domain of journalism, and their goals included the development of a corpus of cases of journalistic text reuse as well as the development of methods to detect them automatically. Following up this project was one from Lancaster University in which the technologies developed in the aforementioned project were used to study text reuse in journalism from the 17th century.[3] Later on, a third project started at the University of Massachusetts Amherst to study methods for text reuse detection on the web.[4] Apart from the research done in these projects, hardly anything else can be found. Despite the fact that text reuse is a more general concept than plagiarism, the latter has received comparably a lot more attention in terms of publications. This includes research in many different research fields, including

[2] http://www.dcs.shef.ac.uk/nlp/meter, see also [43, 48]
[3] http://www.lancs.ac.uk/fass/projects/newsbooks/reuse.htm
[4] http://ciir.cs.umass.edu/research/textreuse.html, see also [19]

law, medicine, biology, and computer science. Most research on detecting text reuse comes from two sub-fields of computer science, namely natural language processing and information retrieval. Here, different research topics are dedicated to the detection of all kinds of text reuse shown in Figure 1.1, such as plagiarism, paraphrases, quotations, boilerplate text, and translations. The topics are called plagiarism detection, paraphrase identification, near-duplicate detection, and cross-lingual parallel corpus construction. While all of them are primarily concerned with analyzing text semantics, the topics author identification and intrinsic plagiarism detection deal with syntax characteristics of texts that reveal authorial style.

1.1.2 Language Reuse

There are hardly any publicly known language reuse technologies. Perhaps the oldest one is the so-called concordancer which is a specialized search engine used by linguists to study language usage in large corpora of texts. A concordance [sic] is an index which allows for immediate access to all occurrences of a given keyword including the various contexts in which they are used. Such indexes have been constructed manually since the middle ages, whereas today, they are constructed on-the-fly by concordancers. Perhaps the most widely known technologies based on language reuse is machine translation (e.g., Google Translate), where existing translations are reused to translate previously unseen texts. Besides machine translation, language is increasingly being reused in support of solving a variety of computational tasks. This typically happens behind the scenes of a system, the details of which are often hidden from its users.

As a research field, language reuse has not yet been recognized or studied systematically. Instead, language reuse is conducted independently—and rather implicitly—in many different research fields, such as linguistics, digital humanities, information visualization, natural language processing, information retrieval, and

machine learning. In linguistics, language is reused as a matter of course to learn more about language. This includes research within computer linguistics on tools, such as concordancers, as well as the systematic acquisition of text data within corpus linguistics. In this connection, the web-as-corpus initiative should be mentioned which has recognized the web itself as a corpus for study. In the digital humanities, a topic called "culturomics" has recently emerged, which mashes up tools similar to concordancers with corpora of books from as many time periods as possible in order to study how cultural developments are reflected within language use over time, and to predict future trends based on today's language [135]. In information visualization, an emerging sub-field is visual analytics, which is about combining data mining and visualization to better make sense of large amounts of data, including text. Here, an often studied topic are keyword-in-context visualizations, which is in fact a synonym of concordance visualizations.

In natural language processing, one of the grant challenges is to teach computers to write on a given topic. Since computers are not yet capable of doing that, one of the straightforward solutions is to reuse language written by humans for this task. This solution is currently applied in order to automatically generate translations, paraphrases, and summaries. In fact, the term "language reuse" has been coined in connection with automatic text summarization [178]. Since generating text reliably is still out of reach for computers, an emerging field of research and development is to crowdsource translations, paraphrases, and summaries. In information retrieval and machine learning, language is reused to support a variety of tasks. We have not attempted to map out all of these tasks, but there are two large sources of language which are being reused frequently in diverse ways: Wikipedia and web n-grams. Wikipedia has proven to be an important resource for many tasks; Medelyan et al. [133] give a comprehensive overview. Similarly, since large corpora of web n-grams (i.e., short phrases up to n words and their frequency of

occurrence on the web) became available, many tasks can be solved more easily or with better performance or both. Particularly the Web 1T 5-gram Corpus [26] released by Google has had a sizable impact on research. But besides these two, there are numerous other text corpora big and small, enriched with meta information that make them valuable resources for language reuse.

Given this potpourri of use cases, it is not entirely clear which tasks and classes of problems can be solved in part or in full by means of language reuse. An obvious example is automatic text generation, but apparently, besides the syntax and semantics of texts also meta information and context relations can be exploited. The not so obvious problems which may benefit from language reuse are indeed difficult to be identified, let alone enumerated.

1.2 Contributions of this Thesis

The thesis in hand divides into two parts. In the first part we shed light on the question of how and to what extent reused text can be detected. The second part is concerned with the question of what tasks can be supported by language reuse.

1.2.1 Text Reuse

The general approach to automatically detect text reuse is similar to that of manual detection for all the different kinds of reuse depicted in Figure 1.1, but there is no one-fits-all approach. Hence, our research on detecting text reuse focuses on three kinds of reuse, namely quotations, boilerplate text, and translations. Another key aspect of our research is the development of the first standardized evaluation framework for text reuse and plagiarism detectors.

Quotations and boilerplate text are two of a kind since both have in common that a reused text is hardly modified and therefore nearly identical to its original source. In the literature, these two kinds of

reuse are hence often grouped under the term "near-duplicates." In Chapter 2 we present a unifying overview and a large-scale evaluation of fingerprinting methods for the detection of near-duplicates. For the first time, connections between fingerprinting methods that employ a projection-based dimensionality reduction and those based on embedding are discussed and compared. The results of our evaluations suggest that the projection-based method supershingling performs best among the evaluated algorithms. Another insight from this evaluation is that the low-dimensional embeddings of texts produced as an intermediate step of fuzzy-fingerprinting can also be used as a retrieval model, rather than the fingerprints computed from them. These embeddings can be computed in linear time in the text length and they perform comparable to the standard vector space model but with a smaller memory footprint.

Translations are a significant challenge for text reuse detection since one cannot rely on syntactical similarities between a reused text and its original. Modeling the semantics of a text and mapping it from one language to another usually involves setting up translation dictionaries, or collecting large numbers of translations in so-called parallel corpora in order to build a machine translator. Such resources are difficult to obtain in practice and the technologies involved difficult to handle. In Chapter 3 we introduce a new model for the comparison of texts across languages that gets by without translation dictionaries, parallel corpora, and machine translation. Instead, our model relies on comparable corpora. They are a much cheaper resource, since only pairs of texts about the same topic are required for the languages in question. For instance, Wikipedia is a comparable corpus as it comprises large quantities of texts whose topic has been described exhaustively, but independently in many languages. Our model exploits the multilingual topic relations of Wikipedia articles to quantify text similarity across languages. In a large-scale evaluation on 6 pairs of languages, our model performs comparable to or better than two traditional approaches.

Evaluating a text reuse detector encompasses building a corpus in which a large number of cases of reused text is found as well as developing performance measures that quantify to what extent the detector is capable of retrieving them. Ideally, the corpus and the measures are publicly available so that researchers may use them to ensure comparability of results across papers. After an extensive review of literature related to automatic text reuse and plagiarism detection, however, it turns out that no such standard corpus exists and that the performance measures employed disregard important aspects of detecting text reuse. In Chapter 4 we present the first standardized evaluation framework for text reuse and plagiarism detectors. It comprises a large-scale corpus with more than 60 000 plagiarism cases of all kinds, about 4 000 of which have been paraphrased manually via crowdsourcing. In addition we have devised performance measures that are tailored to assessing the performance of plagiarism detectors and that take into account specific characteristics not measured by traditional measures. We furthermore report on the results of three evaluation competitions we organized using our framework in which a total of 32 plagiarism detectors have been evaluated. The insights gained from these workshops have advanced the state of the art in text reuse and plagiarism detection.

Altogether, our contributions to detecting text reuse shed light onto important problems: efficient retrieval, multilingual retrieval, and retrieval evaluation. The latter two have not been addressed in sufficient detail before.

1.2.2 Language Reuse

Language can be reused in many ways to solve computational problems. We have identified three new ways of doing so with texts from the web, allowing for models and algorithms that compare web items across media, determine intended quotations in a keyword query, and assist writers with searching for words.

Multimedia items make up a significant portion of the web these days, and naturally web users search for them, too. Web search engines hence face the problem of matching keyword queries against multimedia items. Traditional approaches to this problem rely on corpora of multimedia items which have been manually annotated with keywords and they train machine learning algorithms to learn the connection between low-level item features and keywords. Such corpora are available only on a small scale. In Chapter 5 we identify web comments on multimedia items as a source of information about them, a source which has been neglected until now. We introduce a new retrieval model that is capable of quantifying the topical similarity of two web items of arbitrary media. The model uses the combined text from all comments on the items in order to represent them. Our comprehensive literature survey reveals that web comments in general have not been studied well in information retrieval. We organize the retrieval tasks related to web comments and conduct four experimental case studies.

Keyword queries are the predominant form of web search queries. Although almost all search engines offer advanced options, allowing their users to narrow down a search, few users are even aware of them. One such option is to quote phrases in a query to search for webpages that contain them verbatim, which speeds up retrieval. In Chapter 6 we introduce a new algorithm to automatically insert quotes into keyword queries around phrases which would have likely been quoted by a web searcher. The basic assumption of our approach is that only phrases should be quoted which are sufficiently frequent on the web, and we reuse a large portion of the web in the form of n-grams as a representative sample of web phrases. In a large-scale evaluation we show that our algorithm outperforms 8 other algorithms that rely on significantly more complex features. We furthermore construct a new corpus of 50 000 manually quoted queries via crowdsourcing which is two orders of magnitude larger than the current standard corpus.

Writing is not so much about what to write, but how. Finding the right words to say something is essential to maximize understanding in a text's target audience. Today, in many research fields English is the language of science. However, most scientists' first language isn't English, which brings about difficulties for them. For instance, second language speakers often lack the vocabulary and usage skills of native speakers. Searching for words, however, has not been well supported until now. In Chapter 7 we introduce Netspeak, a new kind of word search engine. Netspeak reuses the web as a corpus of writing examples. It indexes the web in the form of n-grams and implements wildcard search on top of that. Search queries are short phrases where wildcards are inserted at positions of doubt. Given a query, Netspeak retrieves matching n-grams, ranked according to their occurrence frequency on the web. This way, commonly used phrases can be distinguished from less common ones.

Altogether, we show three new ways to accomplish specific tasks by means of language reuse: cross-media item comparison, keyword search, and word search. There may be countless possibilities of reusing language to support computational tasks, yet identifying and exploiting them is the main challenge.

1.2.3 Text Reuse in this Thesis

The thesis in hand is a thesis by publication and hence an example of scientific text reuse. A thesis by publication collects and organizes the author's prior scientific publications, whereas vital contributions have already been subject to the scrutiny of external peer-reviewers. Table 1.1 gives an overview of the publications that are collected in this thesis and where they have been published previously.

Table 1.1: Overview of text reuse within this thesis.

Reused Text	**Original Text**				
	Venue	Type	Length	Year	Reference
Chapter 2	DIR	workshop	full	2007	[210]
	ICTIR	conference	full	2007	[211]
	GFKL	conference	full	2008	[165]
Chapter 3	ECIR	conference	short	2008	[166]
	ECIR	conference	poster	2010	[6]
	LRE	journal	full	2011	[174]
Chapter 4	PAN@SEPLN	workshop	full	2009	[169]
	COLING	conference	full	2010	[171]
	CLEF	conference	full	2010	[170]
	CLEF	conference	full	2011	[175]
Chapter 5	SIGIR	conference	poster	2009	[162]
	WWW	conference	poster	2010	[172]
	TIST	journal	full	2012	[35]
Chapter 6	SIGIR	conference	poster	2010	[81]
	WWW	conference	full	2011	[82]
Chapter 7	ECIR	conference	short	2010	[214]
	PacificVis	conference	full	2011	[183]

1.3 A Brief Introduction to Information Retrieval

In a nutshell, research and development in information retrieval is concerned with building systems that help people to recover the, to them, useful portions of data stored on computers.

Today, web search engines are prime examples of information retrieval systems, and their importance for the web can not be underestimated. Since the web cannot be used effectively without search engines, hardly any further motivation for information retrieval is required. The starting point for research and development is a person who perceives a certain need for information and who eventually decides to do something about it. All of us, frequently, perceive information needs, at work and at leisure. Few, however, reify the need itself but rather its cause: for instance, when noticing a gap of knowledge while doing something [97]. However, gaps of knowledge hardly account for all types of information needs, and while they are still subject to research, suffice it to say that information needs are interrelated with and arise from the fundamental human needs and our innate desire to satisfy them. There are three possible courses of action to satisfy an information need:

1. To ask someone else.
2. To search a collection of documents.
3. To reason based on prior knowledge.

The choice of action depends on the information need at hand. For example, when Bob needs Alice's new cell phone number, he may ask a common friend or search Alice's profile on a social network, while reasoning what the new number might be is pointless. But then, when Eve tries to recall the circle of fifths, she may look it up in a book or deduce it from the mnemonic "Father Charles Goes Down And Ends Battle" that she memorized earlier, while asking someone else might be embarrassing for her, being a musician.

All of these actions can be, and indeed are, supported by technology. Traditionally, the focus of information retrieval has been on search, but there is really no reason for such limitations since its main goal is to help satisfying all kinds of information needs.

Related Work With the advent of computers, information retrieval grew out of library science in response to the accelerating growth of digital data. It is a sub-discipline of information science. Research in information retrieval is by definition interdisciplinary (i.e., it takes advantage of whatever method or technology that helps to accomplish the task of satisfying information needs). The usual suspects in this connection are databases; (computer) linguistics and natural language processing; statistics, data mining, and machine learning; as well as information visualization and human-computer interaction. Technologies from these fields are applied for data organization and storage; for syntactic and semantic analyses of natural language text; for theoretical analyses of retrieval approaches, pattern discovery, and their discrimination; as well as for visual preparation of retrieved information and improving system interfaces. Much more can be said about the contributions of either field to information retrieval, whereas the latter has had a lot of impact on them, too.

A number of information retrieval primers are available: those of Rijsbergen [184], Salton and McGill [195], and Baeza-Yates and Ribeiro-Neto [7] are among the most influential. The book of Manning et al. [128] provides an up-to-date introduction. Regarding web information retrieval, and search engines in particular, the books of Witten et al. [230] and Croft et al. [53] must also be mentioned.

1.3.1 Information Retrieval Terminology

This section outlines a formal framework of information retrieval, defining most of the terminology that is being used along the way, namely queries, data, information, relevance, retrieval models, retrieval tasks, retrieval processes, and retrieval systems.

Again, the starting point is a person's information need, denoted by q, where all people's information needs are denoted by Q. Note that q is often used interchangeably to refer to different aspects of a person's need, such as the cognitive need, the need's cause, the need's topic, the way need's expression, and the expression's formalized version called query. The formulation of a query is the interface between human and machine, since the query has to be formulated in the query language of the retrieval system used. For instance, the query language of a web search engine basically defines valid queries to be lists of keywords from the vocabulary of the web.

The data supposed to contain information related to a given q is denoted by D. This can be all kinds of digital data; usually text, imagery, audio, and video. Information, in this connection, is defined as the useful portions of D, where a data portion's usefulness depends on its capability to satisfy the information need q. A data portion that is useful with regard to q is said to be relevant to q. Defining information this way brings about the problem that, inasmuch as the people's information needs Q cannot be enumerated, so can't the information in D. Therefore, information retrieval systems make use of the fact that the data in D is usually organized in documents, so that each document d in D contains information about a particular topic. This way, information retrieval boils down to identifying a document $d \in D$ that is relevant to q.

The relevance of a document d to a query q is measured by means of a retrieval model $\mathcal{R} = \langle \alpha, \beta, \rho \rangle$. In general, $\mathcal{R}$ consists of the representation functions $\alpha : D \to \mathbf{D}$ and $\beta : Q \to \mathbf{Q}$ and a relevance function $\rho : \mathbf{D} \times \mathbf{Q} \to \mathbf{R}$. Both α and β map documents $d \in D$ and queries $q \in Q$ onto purposeful, machine-readable abstractions $\mathbf{d}$ and $\mathbf{q}$ that represent their human-readable counterparts d and q in the computations. The relevance function maps a pair of representations $(\mathbf{d}, \mathbf{q})$ onto a real value that is supposed to quantify the relevance of d to q. By convention, the bigger the relevance value of d to q, the more relevant it is—that is, according to the retrieval model used.

Given a query q, a set of documents D, and a retrieval model $\mathcal{R}$, retrieving documents from D is the same as ranking them in descending order of their relevance to q. The ranked list may already be presented to the user who may study them until the information need is satisfied or the search is deemed unsuccessful. To obtain a binary relevance decision on every document, a relevance threshold τ has to be specified that divides D into disjunct sets of, supposedly, relevant documents $D^+ = \{d \mid d \in D \text{ and } \rho(\mathbf{d}, \mathbf{q}) \geq \tau\}$ and non-relevant documents $D^- = D \setminus D^+$.

The above outlines the formal framework within which most information retrieval solutions can be described. However, not all information needs are the same, and there is of course no one-fits-all retrieval model. In practice, retrieval models are tailored to solve particular retrieval tasks, which in turn are formal descriptions of real-world problems that imply an information need. For example, consider the well-known task of web search:

> *Web Search.* Given a set of web documents D and a keyword query q, the task is to find all documents in D that are relevant to the information need underlying q.

A retrieval model for this task will be very different from, say, a model for search by example:

> *Search by Example.* Given a set of documents D and a document d_q for a query, the task is to find all documents in D that are similar to d_q.

Retrieval models are operationalized in the form of retrieval processes. For one, a retrieval process is the procedural description of steps to be taken and sub-problems to be solved in order to compute α, β, and ρ of a given $\mathcal{R}$. For another, it also often includes the software architecture and data structures used to implement $\mathcal{R}$ efficiently, robust, and scalable. Finally, a retrieval system implements one or more retrieval models and offers an interface to use them.

1.3.2 Information Retrieval Evaluation

As with all technical solutions to real-world problems, sure enough, the first question about a new information retrieval system is: "Does it work?" A thorough answer to this question requires the evaluation of the system with regard to different performance yardsticks. In research, most evaluations are done in laboratory experiments, which recreate the real world situation underlying a retrieval task on a relatively small scale and in a controlled environment. They are purposeful abstractions of the real world, disregarding everything that does not pertain to the retrieval task while striving for realism on everything that does. When designing an experiment, there are tradeoffs to be made with regard to realism and many demands to be met, such as ensuring its replicability. However, an overview of all parameters and requirements that govern experiment design is beyond the scope of this section; for further reading we refer to the aforementioned primers as well as Vorhees and Harman [227] and Robertson [187], who survey the history of information retrieval evaluation up to the present day. In what follows, we exemplify methods for the evaluation of information retrieval systems, regarding the quality of retrieval models, the speed of retrieval processes, and the usability of a system's user interface.

To assess the quality of a retrieval model $\mathcal{R}$, two things are needed: (1) a corpus of documents D along with queries Q so that for every $q \in Q$ a subset of documents $D^* \subset D$ exists which are known to be relevant to q, and (2) performance measures that quantify the success of $\mathcal{R}$ in retrieving D^* for a given q by comparing it with the obtained retrieval results D^+. Constructing a realistic corpus and choosing reasonable performance measures is of the utmost importance: the validity of the evaluation results rests with them, let alone the decision to invest further in the evaluated retrieval model.

Corpora that consist of documents and queries whose relevance relation is known are usually handmade, but sometimes they can also be constructed automatically, or in the best case, sampled from real queries and their relevant documents. In case a corpus is constructed manually, human judges are presented with pairs of queries and documents and asked to second-guess the information needs underlying the queries in order to judge whether the document presented alongside a query is relevant to it. Obviously, this approach to corpus construction limits the size of a corpus, since the number of pairs of queries and documents grows in $\mathcal{O}(|D| \cdot |Q|)$. The situation is aggravated by the fact that even expert human judges make errors so that more than one judgment is required for every query-document pair; the more the better. Hence, oftentimes, a post-retrieval evaluation on the basis of pooling is conducted by applying retrieval models to a corpus of queries and documents whose relevance relation is unknown and by judging samples of pairs of queries and documents from the obtained retrieval results until an estimation of the overall quality of each model becomes possible. In case a corpus can be constructed automatically, special attention needs to be paid noz to introduce biases of any kind into the corpus. In case a retrieval system is already in use or queries and documents relevant to them can be mined from another application domain, a corpus can be constructed by means of statistical sampling of queries and documents. Anyway, a corpus should contain a representative sample of the population of documents and queries found in the real situation underlying a retrieval task. Statistical representativeness, however, may not always be achieved since not all variables and their distributions are known a-priori, and since the real situation underlying task often cannot be overviewed entirely.

A straightforward way to performance measurement is to count the errors a retrieval model $\mathcal{R}$ makes when retrieving documents from a corpus D for a given q under a relevance threshold τ. Errors, in this connection, are irrelevant documents considered relevant,

Table 1.2: Confusion matrix of retrieval model $\mathcal{R}$ for a given q, when applied on document collection D under relevance threshold τ.

Prediction of $\mathcal{R}$ under τ	**Actual**		**Σ**
	Relevant	Irrelevant	
Relevant	$\lvert D^+ \cap D^* \rvert$	$\lvert D^+ \setminus D^* \rvert$	$\lvert D^+ \rvert$
Irrelevant	$\lvert D^- \cap D^* \rvert$	$\lvert D^- \setminus D^* \rvert$	$\lvert D^- \rvert$
Σ	$\lvert D^* \rvert$	$\lvert D \setminus D^* \rvert$	$\lvert D \rvert$

and vice versa. Likewise, the model's correct decisions are counted. These counts are typically arranged in a so-called confusion matrix, which contrasts the instances where $\mathcal{R}$ confuses relevance and irrelevance with those where it doesn't (see Table 1.2).

Confusion matrices are a useful tool to overview the performance of a single retrieval model, but they are cumbersome when comparing many models, since they do not suggest a ranking among them. Therefore, measures have been proposed that combine the absolute values of a confusion matrix into single, relative performance values, each based on different presumptions and each emphasizing different performance aspects. The two most often applied measures in information retrieval are precision and recall:

$$prec(D^+, D^*) = \frac{|D^+ \cap D^*|}{|D^+|}; \qquad rec(D^+, D^*) = \frac{|D^+ \cap D^*|}{|D^*|}.$$

These measures focus on the subset of relevant documents D^* of D for a given q rather than the typically much larger subset of irrelevant documents; precision measures the purity of the set of retrieved documents D^+, whereas recall measures its completeness regarding D^*. The domain of both measures is $[0, 1]$, where 0 denotes minimum and 1 maximum performance. When designing a retrieval model, performance gains on one of the measures usually come at the price of losses on the other. Note, however, that perfect recall can be achieved simply by adjusting τ so that all documents are

considered relevant (i.e., $D^+ = D$). Hence, only the two of them paint a clear picture of a retrieval model's performance. However, they allow only for a partial order among retrieval models: one retrieval model outperforms another if, and only if, it does so on both precision and recall. To obtain a total order, the two measures can be combined into a single performance value using the F-Measure (i.e., the weighted harmonic mean of precision and recall):

$$F_b(D^+, D^*) = \frac{(1 + b^2) \cdot prec(D^+, D^*) \cdot rec(D^+, D^*)}{b^2 \cdot prec(D^+, D^*) + rec(D^+, D^*)},$$

where $b \in [0, \infty)$ and $b < 1$ emphasizes precision, $b > 1$ emphasizes recall, and $b = 1$ gives both equal weight. Again, the domain of the F-Measure is $[0, 1]$, 0 denoting minimum, and 1 maximum performance. The harmonic mean is used instead of the arithmetic mean since the latter would map a naïve retrieval model that returns $D^+ = D$ onto a performance value close to 0.5 if $|D^*| \ll |D|$, whereas the former maps it onto a more sensible performance value close to 0. Note that all of the above measures still depend on the choice of relevance threshold τ for $\mathcal{R}$; however, it is common practice to choose the threshold τ that maximizes the performance measure under consideration (i.e., to assume best case performance). Also, the performances of different retrieval models can be compared only if they have been obtained based on the same evaluation corpus D. Naturally, experiments should be repeated for many different queries, averaging the results.

The three aforementioned measures represent one of the most common approaches to measure and compare the performance of retrieval models. Yet, there are plenty of other approaches to performance measurement and visualization, including precision-recall graphs, the receiver operating characteristic, the normalized discounted cumulative gain, and many more. We refrain from introducing all of these measures here, since the choice of measure depends on the retrieval task at hand, and since each measure is best explained alongside a practical application.

Finally, the speed of a retrieval process and the usability of a retrieval system's interface can be evaluated. Regarding speed, an analysis of a retrieval process's runtime complexity tells something about its scalability. Successful retrieval systems face thousands up to billions of queries a day, dependent on their success, so that one of the main goals is to achieve at least linear runtime. However, an absolute runtime measurement of the average time to answer a query is also important, since this tells something about the quality of the implementation of the retrieval process in question. Regarding usability, the interface of a retrieval system should be designed so as to support querying and it should visualize the retrieval results in a useful manner. While following basic interface design principles, such as simplicity and minimalism, may be a good start, the only way to tell whether the target audience will accept and use a retrieval system is to conduct user studies and user monitoring. In the former case, test users are asked to answer predefined search queries using a system and then to answer a questionnaire about their experience, while in the latter case the system is released and its users' behavior is monitored in order to draw conclusions about their satisfaction.

1.3.3 Discussion

There are other schools of information retrieval that arrange and formalize its building blocks differently. Many, particularly those presented in the aforementioned primers, focus on keyword search. Above, we take a pragmatic approach in that we consider retrieval tasks and retrieval models as basic framework to explain solutions to satisfy real-world information needs, whereas solutions may differ wildly and even be incompatible to one another in terms of their theoretical or empirical foundations. This pragmatism is rooted in the fact that there is no Grant Theory of information retrieval, yet, and rumor has it there never will be [186].

Web search engines have severely lowered the bar for conducting research in its broadest sense, and people use them to find answers to even the simplest of questions. Web search engines have made us realize information needs at an heretofore unknown scale, which is one of information retrieval's major achievements. An this is in fact what research in information retrieval is all about: identifying and then satisfying information needs we didn't even know we had.

Part I

Text Reuse

Chapter 2

Detecting Near-duplicate Text Reuse

In this chapter we study methods to detect cases of text reuse where the reused text is close to identical to its original. Reusing a text without modifying it is difficult to be accomplished: for example, a reused text is typically reformatted to blend in with the remainder of the document, comments or omissions are added to quotations (e.g., "[sic]" after misspelled words), variable portions of boilerplate text are modified, or small modifications are made in a plagiarized text to hide this fact. Hence, technologies to detect such cases of text reuse cannot rely on the reused text being exactly the same as its original but have to be robust to slight or even heavy modifications.

A sub-field of information retrieval called near-duplicate detection deals with this kind of matching. Near-duplicates occur in many document collections, from which the most prominent one is the web. Studies show that about 30% of all web documents are duplicates of others [33, 63]. Examples include mirror sites, revisions and versioned documents, and boilerplate texts such as disclaimers [243]. The negative impact of near-duplicates on web search engines is twofold: indexes waste storage space and search results can be cluttered with almost identical entries.

Within our research, we have studied different approaches to near-duplicate detection, and this chapter compiles an in-depth overview of our respective publications [123, 161, 165, 209, 210, 211]. Section 2.1 introduces a formal framework for the detection of near-duplicate texts based on a technique called fingerprinting. Section 2.2 details the principles of constructing fingerprints and surveys the relevant algorithms from the literature. Section 2.3 reports on a large-scale evaluation of a selection of these algorithms.

Our contributions are as follows. First, a unified view of the diverse existing approaches to near-duplicate detection. Most prominently, two major paradigms of near-duplicate detection are identified and compared for the first time, namely, projecting-based fingerprinting and embedding-based fingerprinting. Moreover, we show that the latter is a space partitioning method. Finally, within our evaluation, we compare well-known algorithms of both paradigms, and propose a new, representative evaluation corpus for this purpose. The evaluation itself provides new insights regarding the performance characteristics of the evaluated algorithms.

2.1 Near-duplicate Detection Based on Fingerprinting

Text-based information retrieval in general deals with the search in a large document collection D. In this connection, we distinguish between a "real" document $d \in D$ and its computer representation $\mathbf{d}$ under a given retrieval model. Likewise, $\mathbf{D}$ denotes the set of computer representations of the real documents in D. Typically, a document representation $\mathbf{d}$ is an m-dimensional feature vector, which means that the objects in $\mathbf{D}$ can be considered as points in the m-dimensional vector space. The similarity between two documents d and d_q is inversely proportional to the distance of their feature vectors $\mathbf{d}$ and $\mathbf{d}_q$. It is measured by a function $\varphi(\mathbf{d}, \mathbf{d}_q)$ which maps onto $[0, 1]$, with 0 and 1 indicating no and a maximum similarity respectively; φ may rely on the l_1-norm, the l_2-norm, or on the angle

between the feature vectors. Obviously, the most similar document $d^* \in D$ respecting a query document d_q maximizes φ. The task of finding d^* is called nearest neighbor search. The task of finding all documents whose similarity is within an ε-environment of $\mathbf{d}_q$ is called similarity search, and when restricting the search so that the ε-environment is small, say, the similarity of the documents sought-after to d_q is close to 1, the task is also known as near-duplicate detection. Here, two documents d and d_q are considered as near-duplicates if the following condition holds:

$$\varphi(\mathbf{d}, \mathbf{d}_q) \geq 1 - \varepsilon \quad \text{with } 0 < \varepsilon \ll 1.$$

A solution to the outlined problem shall provide a high retrieval performance in terms of precision and recall, and at the same time a high runtime performance, which are competitive objectives. Regarding retrieval performance, the optimal approach to identify two documents d and d_q as near-duplicates under a given retrieval model is to compute their representations $\mathbf{d}$ and $\mathbf{d}_q$, and to measure their similarity directly. With this approach, the computation of the set $D_q \subset D$ which contains all near-duplicates of d_q in D requires $\mathcal{O}(|D|)$ time, say, linear time in the collection size. Documents are typically represented with a high dimensionality m, where "high" means $m > 10$, which forecloses a sub-linear search based on spatial indexes, such as kd-trees, quad-trees, or R-trees. In this case, these data structures are outperformed by an exhaustive comparison [228]. If the document representations are sparse, which is typical in information retrieval, this fact can be exploited to speed up an exhaustive comparison: by utilizing an inverted index, documents having nothing in common with d_q can be pruned from the search. However, the runtime complexity with this approach remains in $\mathcal{O}(|D|)$.

By relaxing the retrieval requirements in terms of precision and recall, the runtime performance can be further improved. Basic idea is to approximate the similarity between d and d_q within a k-dimensional space, with $k \ll m$, by means of fingerprinting. A

fingerprint F_d of a document d is a k-dimensional vector whose elements are natural numbers computed from d using a certain hash function. If two fingerprints F_d and F_{d_q} have at least κ hash values in common, with $\kappa \leq k$, it is assumed that d and d_q are near-duplicates. Let $F_D = \bigcup_{d \in D} F_d$ denote the union of the fingerprints of all documents in D, then a fingerprint index $\mu : F_D \rightarrow \mathcal{D}$ can be constructed which maps each hash value $x \in F_D$ onto the power set of D, so that $\mu(x)$ is the set of documents that share x in their fingerprints. If $\kappa = 1$ is assumed, the retrieval of near-duplicate documents D_q from D for a given query document d_q can be accomplished in $\mathcal{O}(1)$ time by computing d_q's fingerprint F_{d_q} and then $D_q = \bigcup_{x \in F_{d_q}} \mu(x)$.

2.2 Fingerprint Construction

This section introduces both a unifying framework and the underlying principles for a wide range of fingerprint algorithms. The construction of a fingerprint F_d of document d can be organized within four steps, consisting of representation, dimensionality reduction, quantization, and encoding (see Figure 2.1 for an illustration):

1. Representation of d as a high-dimensional feature vector $\mathbf{d}$ under a given retrieval model.

2. Dimensionality reduction by projecting or by embedding. Algorithms of the former type select dimensions in $\mathbf{d}$ whose values occur unmodified in the low-dimensional vector $\mathbf{d}'$. Algorithms of the latter type reformulate $\mathbf{d}$ as a whole to obtain $\mathbf{d}'$, maintaining as much information as possible.

3. Quantization is the mapping of the elements in $\mathbf{d}'$ onto small integer numbers, obtaining $\mathbf{d}''$.

4. Encoding is the computation of one or several codes from $\mathbf{d}''$, which together form the fingerprint F_d.

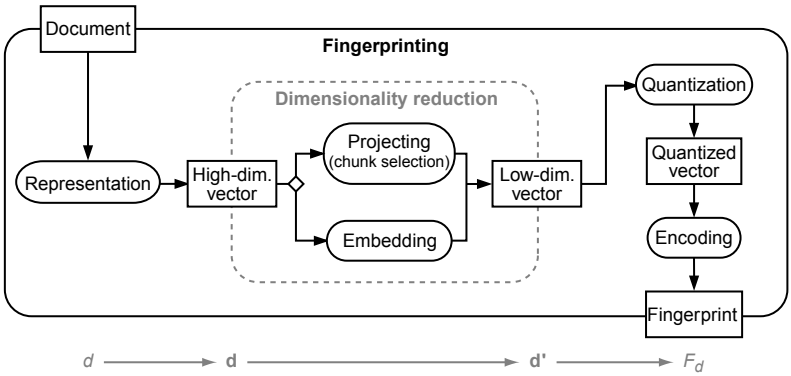


Figure 2.1: The construction of a fingerprint F_d of a document d. After the dimensionality reduction step, the low-dimensional vector $\mathbf{d}'$ is quantized and then encoded into F_d.

Fingerprint algorithms differ mainly in their dimensionality reduction method. Figure 2.2 organizes the methods alongside well-known fingerprint algorithms.[1] Before going into detail, we compare the orders of magnitude that can be expected from a dimensionality reduction. The dimension m of vectors used to represent documents depends on the retrieval model employed. Most retrieval models chunk a document which yields a vocabulary T. T is the union set of all chunks (terms, 1-grams, or n-grams in general) that occur in at least one document representation $\mathbf{d}$ from a collection $\mathbf{D}$. Hence, $\mathbf{d}$ may be understood as an m-dimensional vector whose components correspond to the chunks in T. In practice, however, only a fraction of the components of a given representation $\mathbf{d}$ have a non-zero weight. A more efficient way to encode such sparse vectors is a list of $(\mathit{descriptor}, \mathit{weight})$-tuples, disregarding all zero-weight

[1] We have excluded latent and hidden variable models (such as LSI, pLSI, LDA, etc.), which also embed documents into a low-dimensional space, since their exponential runtimes disqualify them for large-scale use.

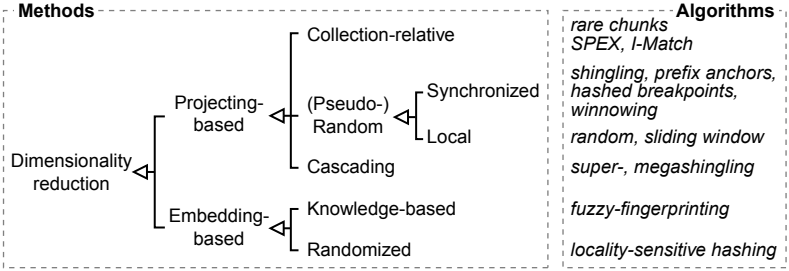


Figure 2.2: Taxonomy of dimensionality reduction methods (left), aligned with fingerprint algorithms using them (right).

dimensions. It is important not to confuse the encoding size of a document $|\mathbf{d}|$ with the dimensionality of its vector space $m = |T|$. Figure 2.3 contrasts the dimensionality of different document spaces with the respective sizes of the document representations $\mathbf{d}$.

2.2.1 Projecting-based Fingerprinting

A chunk or an n-gram of a document d is a sequence of n consecutive words from d. Let C_d be the set of all chunks of d. The size of C_d is at most $|d| - n$, say, it is within $\mathcal{O}(|d|)$. Let $\mathbf{d}$ be a vector representation of d where each $c \in C_d$ is used as descriptor of a dimension with non-zero weight. If dimensionality reduction is done by projecting, a fingerprint F_d for document d can be formally defined as follows:

$$F_d = \{h(c) \mid c \in C_d \text{ and } \sigma(c) = true\},$$

where σ denotes a chunk selection heuristic that becomes true if a chunk fulfills a certain property, and h denotes a hash function, such as MD5 or Rabin's hash function, which maps chunks to natural numbers, serving as a means for quantization. Usually the identity mapping is applied as encoding rule, but for example, the authors of [32] describe a more intricate encoding rule called supershingling.

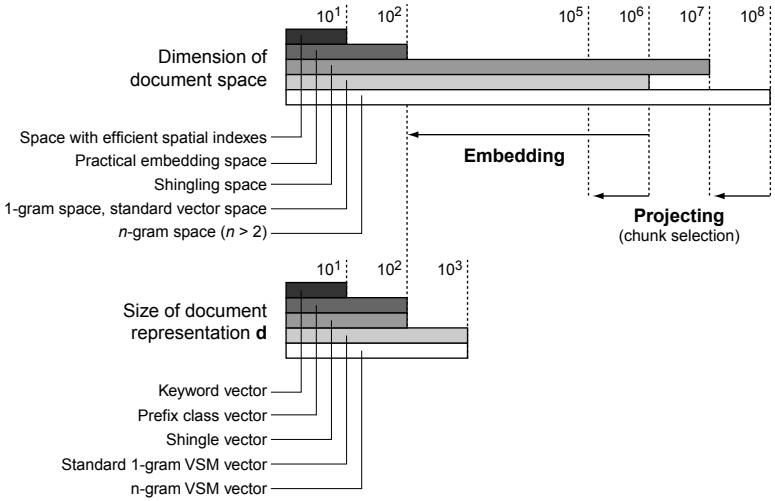


Figure 2.3: The diagram contrasts the dimensions of different document spaces with the sizes of the document representations. Also, the dimensionality reduction typically achieved with fingerprint algorithms is shown.

The objective of σ is to select chunks to be part of a fingerprint which are best-suited for a reliable near-duplicate identification. Table 2.1 presents a unified overview of existing projecting-based near-duplicate detection algorithms, and their respective chunk selection heuristics, whereas a heuristic is of one of the types denoted in Figure 2.2. In this connection, we emphasize the distinction between chunk selection heuristics that reach their decision relative to a given collection of documents compared to those which don't. This distinction pays tribute to the fact that, typically, the former is much more efficient with respect to runtime, while the latter enables one to integrate global considerations as well as knowledge from the retrieval task. Note, however, that a document-specific dimension reduction presumes a closed retrieval situation [207].

Table 2.1: Summary of chunk selection heuristics. The rows contain the name of the fingerprint algorithm along with the constraints that are to be fulfilled by a chunk in order to be selected by σ.

Algorithm	**Reference**	**Chunk selection heuristic** $\sigma(c)$
rare chunks	[83]	c occurs once in D
SPEX	[24]	c occurs at least twice in D
I-Match	[41, 50, 111]	$c = d$; excluding non-discriminant terms of d
shingling	[32]	$c \in \{c_1, \ldots, c_k\}$, $\{c_1, \ldots, c_k\} \subset_{rand} C_d$
prefix anchor	[126]	c starts with a particular prefix, or
	[83]	c starts with a prefix which is infrequent in d
hashed breakpoints	[126]	$h(c)$'s last byte is 0, or
	[30]	c's last word's hash value is 0
winnowing	[197]	c minimizes $h(c)$ in a window sliding over d
random	misc.	c is part of a local random choice from C_d
one of a sliding window	misc.	c starts at word $i \bmod m$ in d; $1 \leq m \leq \lvert d \rvert$
super- / megashingling	[32, 63]	c is a combination of hashed chunks which have been selected with shingling

2.2.2 Embedding-based Fingerprinting

An embedding-based fingerprint F_d for a document d is typically constructed with a technique called locality (or similarity) sensitive hashing [96]. Unlike standard hash functions, which aim at minimizing hash collisions, a similarity sensitive hash function $h_\varphi : \mathbf{D} \rightarrow U$, where $U \subset \mathbf{N}$, is designed to produce hash collisions with a high probability if a pair of objects $\mathbf{d}$ and $\mathbf{d}_q$ from $\mathbf{D}$ are similar. Ideally, such a hash function would fulfill the following property [206]:

$$\underbrace{h_\varphi(\mathbf{d}) = h_\varphi(\mathbf{d}_q)}_{\alpha} \quad \Leftrightarrow \quad \underbrace{\varphi(\mathbf{d}, \mathbf{d}_q) \geq 1 - \varepsilon}_{\beta},$$

where $\mathbf{d}, \mathbf{d}_q \in \mathbf{D}$ and $0 < \varepsilon \ll 1$. That is, a hash collision between two elements from $\mathbf{D}$ indicates a high similarity between them and vice versa. The most salient property of h_φ is the simplification of a fine-grained similarity quantification, operationalized as similarity function φ, to the binary concept "similar or not similar." However,

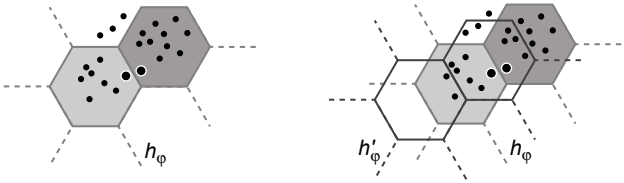


Figure 2.4: A set of documents projected in the two-dimensional plane. A hash function h_φ partitions the space into regions that are characterized by different hash values. Even if two documents are very similar to each other, they may be mapped onto different hash keys (left). This threshold-behavior can be alleviated by employing several functions h_φ and h'_φ (right).

this property cannot be guaranteed: if $\alpha \not\Rightarrow \beta$, collisions occur for pairs of documents which are not near-duplicates, and if $\alpha \not\Leftarrow \beta$, not all pairs of near-duplicate documents are mapped onto the same hash value. The former relates to the concept of precision and the latter to that of recall. In fact, $\alpha \Leftarrow \beta$ can be disproved: given a hash function h_φ, it statically determines a hash value for every document in existence, and as a consequence, defines an absolute partitioning of the space of document representations.[2] Therefore, highly similar documents may exist which are mapped onto different hash values.

Figure 2.4 illustrates this connection: despite their high similarity (= low distance), a hash function h_φ will map some of the document representations onto different hash values. By employing a second hash function h'_φ, which defines a different partitioning, the logical disjunction of both functions can be used to map more of the near-duplicates onto the same hash value. In practice, one can observe a monotonic relationship between the number k of hash functions used simultaneously and the achieved recall, but there is no free lunch: the improved recall is bought with a decrease in precision.

[2] By contrast, a complete similarity graph underlying a set $\mathbf{D}$ of document representations defines for each element its specific partitioning.

For a given document d, a fingerprint F_d is constructed as follows:

$$F_d = \{h_\varphi^{(i)}(\mathbf{d}) \mid i \in \{1, \ldots, k\}\}.$$

Two kinds of similarity sensitive hash functions have been proposed: fuzzy-fingerprinting and locality-sensitive hashing. The former exploits domain knowledge, whereas the latter grounds on domain-independent randomization techniques (see again Figure 2.2). In what follows, we describe these hash functions in detail.

Fuzzy-Fingerprinting Fuzzy-fingerprinting is a similarity sensitive hash function designed for, but not limited to text information retrieval [206]. It defines a small number of $k \in [10, 100]$ prefix equivalence classes. A prefix class, for short, contains all terms starting with a given prefix. For the Latin alphabet, 26^i prefix classes can be defined, where i denotes prefix length. For a given vector representation $\mathbf{d}$, $h_\varphi(\mathbf{d})$ computes as follows (see Figure 2.5): (1) Embedding of $\mathbf{d}$ into a k-dimensional space whose dimensions quantify the distribution of the index terms in $\mathbf{d}$ with respect to the k prefix classes. The embedded vector $\mathbf{d}'$ is normalized by computing its difference from the vector of expected values. The expected values for each prefix class dimension are pre-computed based on a large document collection. (2) Quantization of each vector component of $\mathbf{d}'$ by means of fuzzification. A fuzzification scheme $\rho : \mathbf{R} \to \{1, \ldots, r\}$ is used which maps the real-valued deviations from the expectation in $\mathbf{d}'$ onto one of r deviation intervals, thus obtaining the quantized vector $\mathbf{d}''$. (3) Encoding of $\mathbf{d}''$ by summation of its vector components:

$$h_\varphi^{(\rho)}(\mathbf{d}) = \sum_{i=0}^{k-1} \delta_i^{(\rho)} \cdot r^i, \quad \text{with } \delta_i^{(\rho)} \in \{0, \ldots, r-1\},$$

where $\delta_i^{(\rho)}$ is the i-th vector component of $\mathbf{d}''$, which represents the fuzzified deviation from the expectation of the i-th component of $\mathbf{d}'$, when applying fuzzification scheme ρ.

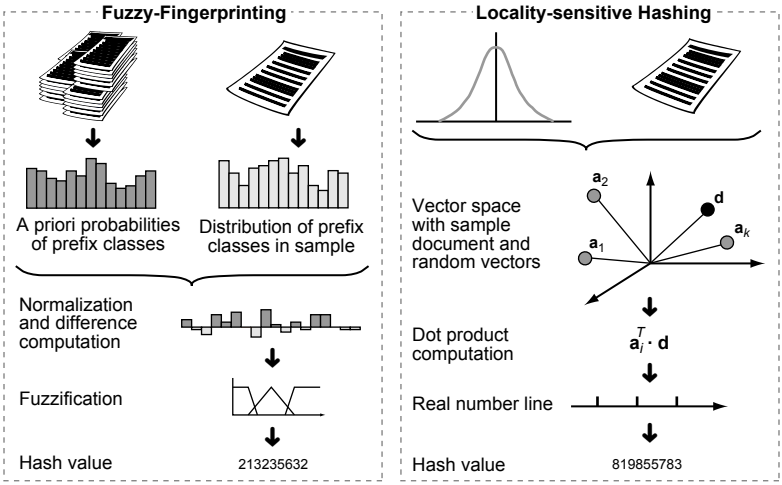


Figure 2.5: The basic steps of hashing with fuzzy-fingerprinting and locality-sensitive hashing.

Locality-Sensitive Hashing Locality-sensitive hashing (LSH) is a generic framework for the randomized construction of hash functions [96]. Based on a family H_φ of simple hash functions h, $h : \mathbf{D} \rightarrow U$, a locality-sensitive hash function h_φ is a combination of k functions $h \in H_\varphi$ obtained by an independent and identically distributed random choice. When using summation as combination rule the construction of $h_\varphi(\mathbf{d})$ is defined as follows:

$$h_\varphi(\mathbf{d}) = \sum_{i=1}^{k} h_i(\mathbf{d}), \quad \text{with } \{h_1, \ldots, h_k\} \subset_{rand} H_\varphi.$$

Several hash families H_φ that are applicable for text-based information retrieval have been proposed [40, 57, 15]; our focus lies on the approach of Datar et al. [57]. The idea of this hash family is to map a document representation $\mathbf{d}$ to a real number by computing the dot

product $\mathbf{a}^T \cdot \mathbf{d}$, where $\mathbf{a}$ is a random vector whose vector components are chosen independently from a given probability distribution. The real number line is divided into equidistant intervals of width r each of which having assigned a unique natural number, and the result of the dot product is identified with the number of its enclosing interval. Under this approach the construction of h_φ for a given list $\rho = (\mathbf{a}_1, \ldots, \mathbf{a}_k)$ of random vectors reads as follows:

$$h_\varphi^{(\rho)}(\mathbf{d}) = \sum_{i=1}^{k} \left\lfloor \frac{\mathbf{a}_i^T \cdot \mathbf{d} + c}{r} \right\rfloor,$$

where $c \in [0, r]$ is chosen randomly to allow for all possible segmentations of the real number line (see Figure 2.5 for an illustration).

A lower bound for the retrieval quality of locality-sensitive hashing can be stated: if the average distance of a document to its nearest neighbor is known in advance, h_φ can be adjusted so that the retrieval probability for the nearest neighbor is above a certain threshold [71]. This fact follows from the locality-sensitivity of a hash family H_φ, which claims that, for any $h \in H_\varphi$, the probability of a hash collision of two documents increases with their similarity.[3]

2.3 Evaluating Fingerprint Algorithms

When evaluating near-duplicate detection algorithms, one faces the problem of choosing a representative corpus for the retrieval situation which provides a realistic basis to measure both precision and recall. Today's standard corpora, such as the TREC or Reuters Corpus Volume 1 [190], have deficiencies in this respect: the distribution of similarities decreases exponentially from a very high number of low similarities to a very low number of high similarities. Figure 2.6

[3] In order to guarantee a hash family being locality-sensitive, the distribution must be α-stable. An example for such a distribution is the Gaussian distribution. For further theoretical background we refer to [95, 144].

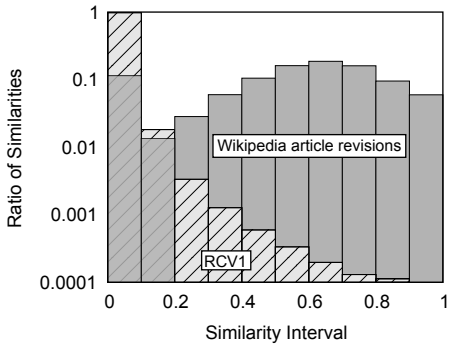


Figure 2.6: Similarity distribution of the documents in the Reuters Corpus Volume 1 (RCV1) and the Wikipedia article revisions.

illustrates this characteristic on the Reuters Corpus Volume 1. This fact allows only for precision evaluations since recall performance depends on very few pairs of documents. Also the corpora employed in the near-duplicate evaluations found in [84, 85, 235] lack in this respect; moreover, they are custom-built and not publicly available. Conrad and Schriber [49] attempt to overcome this issue by the artificial construction of a suitable corpus.

For the evaluation of near-duplicate detection algorithms, we propose to use Wikipedia including all revisions of all Wikipedia articles.[4] Wikipedia periodically publishes snapshots of itself which are available free of charge. The snapshot used in our experiments was the one of August 16, 2006, comprising about 6 million articles with more than 80 million revisions. A pilot study revealed that an article's revisions are often very similar to one another: the expected similarity between two article revisions is about 0.5. Since Wikipedia articles undergo constant editing, the corpus meets the requirements of a meaningful evaluation of near-duplicate detection algorithms.

[4] http://en.wikipedia.org/wiki/Wikipedia:Download

Experiment 1: Fingerprinting Performance We analyzed a selection of the fingerprint algorithms with a total of 7 million pairs of documents, using the following strategy: each article's first revision serves as query document d_q and is compared to all other revisions as well as to the first revision of its immediate successor article. The former ensures a large number of near-duplicates and hence ensures the reliability of the recall values; the latter is to gather sufficient data to evaluate the precision. Figure 2.6 shows the obtained similarity distribution; the similarities are almost evenly distributed. Figure 2.7 presents the results of our experiments in the form of precision-over-similarity curves (top) and recall-over-similarity curves (bottom). The curves are computed as follows: for a number of similarity thresholds from the interval $[0, 1]$ the set of document pairs whose similarity is above a certain threshold is determined. Each such set is compared to the set of near-duplicates identified by a particular fingerprinting method. From the intersection of these sets then the threshold-specific precision and recall values are computed in the standard way. As can be seen in the plots, the chunking-based methods perform better than similarity sensitive hashing, while hashed breakpoint chunking performs best. It must be added, however, that the fingerprints of hashed breakpoint chunking and shingling comprised 50 times more hash values to represent a document than those of fuzzy-fingerprinting and supershingling, which puts into perspective the performance difference between these methods.

Experiment 2: Dimensionality Reduction Performance Furthermore, we have evaluated the quality of the low-dimensional document representation $\mathbf{d}'$ obtained from the dimensionality reduction step when computing the fingerprint of a document d. In particular, we compared the ranking performance of shingling with that of locality-sensitive hashing and fuzzy-fingerprinting, while varying the dimensionality k of the space into which documents were projected or embedded, respectively. For each pair of the mentioned fingerprint algorithms and $k \in \{10, 50, 100\}$, the following exper-

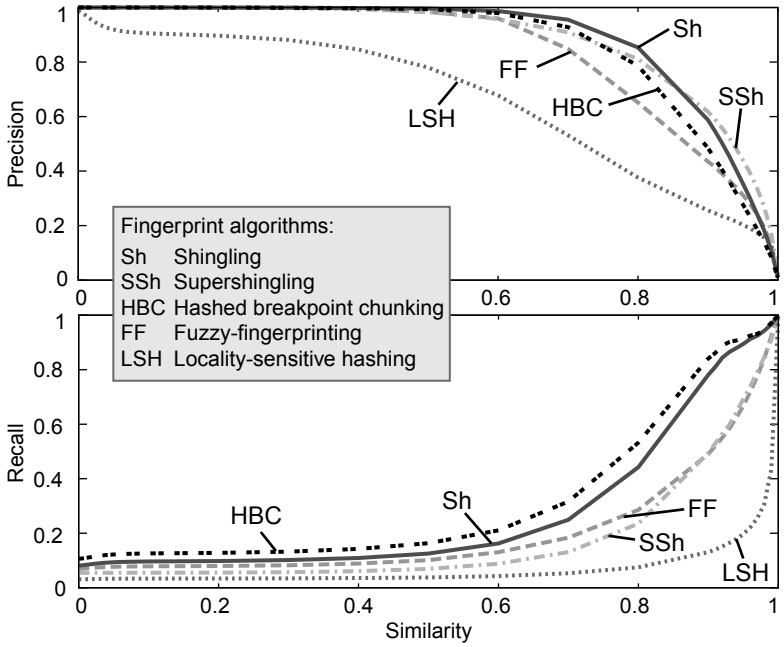


Figure 2.7: Precision and recall over similarity for 5 fingerprint algorithms.

iment was repeated 1000 times, averaging the results: (1) a query document d_q was chosen at random from one category of the Reuters corpus. (2) 1000 other documents, chosen at random from the same category as d_q, were ranked according to their similarity to d_q using the fingerprint method in question. (3) The same documents were ranked again according to their similarity to d_q, this time using a high-dimensional representation based on the standard vector space model. (4) The rank correlation of the two rankings was computed using Kendall's τ. More precisely, six correlations were computed regarding the top most similar documents to d_q having at least a similarity of 0, 0.25, 0.5, 0.65, 0.8, and 0.9.

Rationale for computing the correlation of a ranking obtained with a low-dimensional document representation and that from a high-dimensional representation is to investigate whether the low-dimensional representation may replace the high-dimensional one in practice. Say, if the low-dimensional representation is capable of approximating the ranking obtained with vector space model closely, it can be used in lieu of this model.

Figure 2.8 shows the results of this experiment. As can be seen, an increase in the similarity threshold goes along with an increase in rank correlation. Shingling performs worst; the rank correlation at medium similarity thresholds is considerable smaller than those of the other models. Both locality-sensitive hashing and fuzzy-fingerprinting show a high rank correlation, where the former outperforms the latter. A reduction in the dimensionality impairs the rank correlation for all approaches. Here, the compact models based on embedding are affected by at most 25%, whereas shingling decreases by more than 75%. Considering the third row, both embedding LSH and fuzzy-fingerprinting perform similar.

2.3.1 Conclusions and Future Work

Algorithms for near-duplicate detection are applied for retrieval tasks such as web mining, text reuse and plagiarism detection, and corporate storage maintenance. In this chapter, we developed an integrative view to existing technologies for near-duplicate detection. Theoretical considerations and practical evaluations show that shingling, supershingling, and fuzzy-fingerprinting perform best in terms of retrieval recall, retrieval precision, and chunk index size. Moreover, a new, publicly available corpus is proposed, which overcomes weaknesses of the standard corpora when analyzing use cases from the field of near duplicate detection.

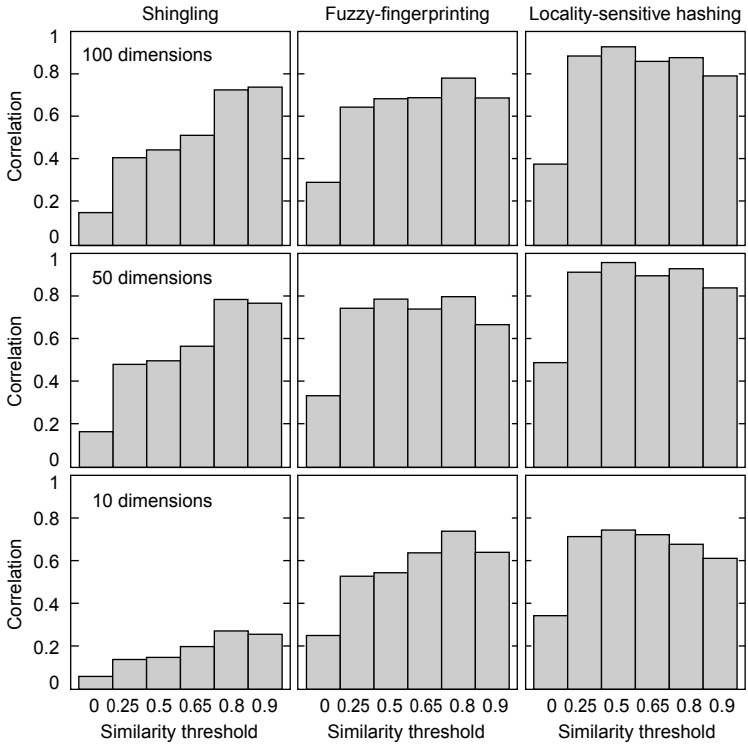


Figure 2.8: Correlation of rankings obtained with low-dimensional representations to those obtained with the high-dimensional standard vector space model. Each plot quantifies the achieved rank correlation value, dependent on six similarity thresholds, as an average over 1000 rankings.

Another insight presented in this chapter related to the simple construction of low-dimensional document representations, as opposed to the more intricate methods used in latent and hidden variable models, such as the singular value decomposition employed by LSI. In this connection, the embeddings produced within fuzzy-fingerprinting and locality-sensitive hashing provide for a lightweight alternative in terms of runtime. Finally, from a practical point of view, fingerprinting can only be employed for near-duplicate detection and retrieval if one has access to the whole document collection, so their fingerprints can be computed and indexed offline. Since, for example, text reuse detection is supposed to happen on web scale, this can hardly be accomplished by others than the major search engine companies.

Chapter 3

Detecting Cross-Language Text Reuse

In this chapter we study methods to detect cases of text reuse across languages. These methods are applied when collecting training material for machine translation systems or evidence of plagiarism. Regarding the latter, we assume that plagiarism does not stop at language barriers. For instance, non-English speaking scholars often write homeworks in their native languages, whereas scientific discourse to refer to is often published in English. Cross-language plagiarism detection has attracted considerable attention [10, 38, 159, 166, 177]. The detection of translated text reuse relates to cross-language information retrieval, a sub-field of information retrieval dealing with satisfying information needs across languages. According to internet statistics services, English is still the top language found on the web, but in total, all other languages together dwarf the size of the English portion of the web.

Within our research, we have studied different techniques that may be applied for the purpose of cross-language text reuse detection, and this chapter compiles an in-depth overview of our respective publications [6, 166, 174]. Section 3.1 revisits the retrieval process for text reuse detection introduced in Chapter 1, emphasizing the differences that have to be dealt with in a multilingual setting.

Section 3.2 surveys retrieval models for the detailed comparison of documents across languages and continues to introduce three recent models called CL-CNG [132], CL-ESA [166], and CL-ASA [10]. Section 3.3 presents a large-scale evaluation of the three mentioned retrieval models. All experiments were repeated on test collections sampled from the parallel JRC-Acquis corpus and the comparable Wikipedia corpus. Each test collection contains aligned documents written in English, Spanish, German, French, Dutch, and Polish.

Our contributions are the following: besides the comprehensive discussion of alternative methods for cross-language text reuse detection, our most important contribution is the cross-language explicit semantic analysis model (CL-ESA), which is a new model for the assessment of cross-language text similarity. Moreover, we report on a large-scale evaluation which compares CL-ESA with two other models. An important observation is the applicability of Wikipedia as a comparable corpus. The chapter concludes with a first time discussion of the feasibility of cross-language fingerprinting.

3.1 Differences to Monolingual Text Reuse Detection

In this chapter we revisit the three-step text reuse retrieval process shown in Figure 1.3, placing emphasis on applying it in a multilingual setting. Let d_q denote a suspicious document written in language L, and let D' denote a document collection written in another language L'. The detection of a text passage in d_q that is reused from some document in D' can still be organized within the aforementioned three steps candidate retrieval, detailed comparison, and knowledge-based post-processing (see Figure 1.3):

1. *Candidate Retrieval.* From D' a set of candidate documents D'_q is retrieved where each document is likely to contain passages that are very similar to certain passages in d_q. This step requires methods to map the topic of d_q from L to L'.

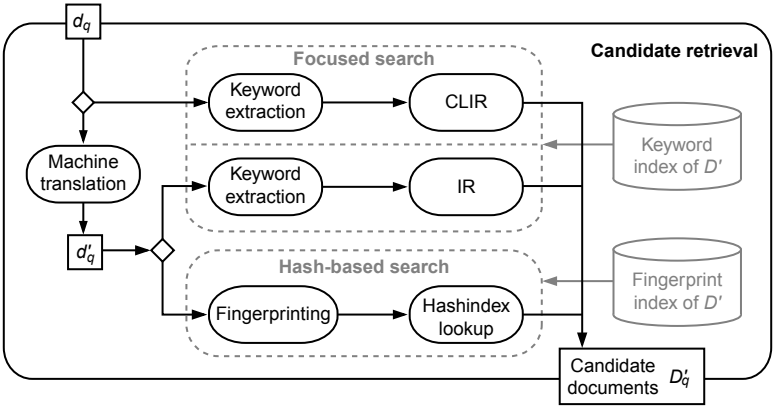


Figure 3.1: Retrieval process for candidate retrieval within cross-language text reuse detection.

2. *Detailed Comparison.* Each document in D'_q is compared passage-wise with d_q, using a cross-language retrieval model. If a high similarity is measured for a pair of passages, a possible case of cross-language reuse is assumed.

3. *Knowledge-based Post-Processing.* Filtering of false positives from the candidate documents, visualization of the remainder.

We identify three alternatives candidate retrieval of candidate documents across languages. They all demonstrate solutions for this task, utilizing well-known methods from cross-language information retrieval (CLIR), monolingual information retrieval (IR), and hash-based search. Figure 3.1 shows the alternatives. The approaches divide into methods based on a focused search and methods based on hash-based search. The former reuse existing keyword indexes and well-known keyword retrieval methods to retrieve D'_q, the latter rely on a fingerprint index of D' where text passages are mapped onto sets of hash codes.

Approach 1 Research in cross-language information retrieval addresses keyword query tasks in first place, where for a user-specified query q in language L documents are to be retrieved from a collection D' in language L'. By contrast, our task is a so-called "query by example task", where the query is the document d_q, and documents similar to d_q are to be retrieved from D'. Given a keyword extraction algorithm both tasks are solved in the same way using standard CLIR methods: translation of the keywords from L to L' and querying of a keyword index which stores D'.

Approach 2 In this approach d_q is translated from L to L' with machine translation technology, obtaining d'_q. Afterwards keyword extraction is applied to d'_q, similar to Approach 1, and the keyword index of D' is queried with the extracted words in order to retrieve D'_q. This approach compares to the first one in terms of retrieval quality.

Approach 3 The fingerprint F_d of a document d is a small set of integers computed from d in a way so that similar documents are mapped onto the same hash values with a high probability. Given d_q's translation d'_q, the candidate documents may be retrieved by querying a fingerprint index of D' with $F_{d'_q}$. Alternatively, one may attempt fingerprinting across languages without first translating d_q to d'_q. This option has not been investigated until now.

Remarks Given the choice among the outlined alternatives the question is: "Which way to go?" We argue as follows: there is no reason to disregard existing web indexes, such as the keyword indexes maintained by the major search engines. This favors Approach 1 and 2, and it is up to the developer if he trusts the CLIR approach more than the combination of machine translation and IR, or vice versa. Both approaches require careful development and adjustment in order to work in practice. Approach 3 looks good on paper, but it

presumes that either the machine translation of d_q yields a document d'_q which is a near-duplicate of the original text reused in d_q, or that cross-language fingerprinting can be realized. The former is unlikely as a text has numerous possible translations and machine translation technology is not mature enough to be controlled in this way, while the latter turns out to be infeasible as well.

3.2 Measuring Cross-Language Text Similarity

This section surveys retrieval models which can be applied in the detailed analysis step of cross-language text reuse detection; they measure the cross-language similarity between passages of the query document d_q and passages of the candidate documents in D'_q. Three retrieval models are described in detail, the cross-language character 3-gram model, the cross-language explicit semantic analysis model, and the cross-language alignment-based similarity analysis model.

3.2.1 Terminology and Related Work

In information retrieval, two documents d_q and d' are compared using a retrieval model $\mathcal{R}$, which provides the means to compute document representations $\mathbf{d}_q$ and $\mathbf{d}'$ as well as a similarity function φ: $\varphi(\mathbf{d}_q, \mathbf{d}')$ maps onto a real value which indicates the topic similarity of d_q and d'. A common retrieval model is the vector space model (VSM) which represents documents as term vectors and measures their similarity using the cosine similarity. However, deductions that come at no expense in monolingual retrieval are difficult to be achieved between two languages L and L': terms, named entities, time or currency expressions, etc. have to be identified and mapped from L to L', which entails the problem of translation ambiguity.

We distinguish four kinds of cross-language retrieval models (see Figure 3.2): (1) models based on language syntax, (2) models based on dictionaries, gazetteers, rules, and thesauri, (3) models based on

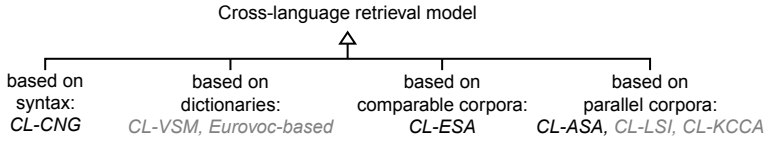


Figure 3.2: Taxonomy of cross-language retrieval models.

comparable corpora, and (4) models based on parallel corpora. Models of the first kind rely on syntactical similarities between languages and on the appearance of foreign words. Models of the second kind can be called cross-language vector space models: they bridge the language barrier by translating words or concepts, such as locations, dates, and number expressions. Models of the third and fourth kind have to be trained on an aligned corpus consisting of documents from the languages to be compared. The two approaches differ with respect to the required degree of alignment: comparable alignment refers to documents in different languages describing roughly the same topic, while parallel alignment refers to documents that are translations of one another whose words or sentences have been mapped manually or heuristically to their respective translations.[1] Obviously the latter poses a much higher requirement than the former. The following models have been proposed:

- CL-CNG represents documents by character n-grams [132].
- CL-VSM and Eurovoc-based model construct a vector space model [117, 176, 215].
- CL-ESA exploits correlations of the vocabulary of comparable corpora [166, 233].
- CL-ASA is based on statistical machine translation [10].
- CL-LSI performs latent semantic indexing [62, 119].
- CL-KCCA uses a kernel canonical correlation analysis [226].

[1] There has been much confusion concerning corpora termed "parallel" and "comparable"; the authors of [131] provide a consistent definition.

The alternatives imply a trade-off between retrieval quality and retrieval speed. Also, the availability of necessary resources for all considered languages is a concern. CL-CNG can be straightforwardly operationalized and requires only little language-specific adjustments (e.g., alphabet normalization by removal of diacritics). The CL-VSM variants offer a retrieval speed comparable to that of the VSM in monolingual information retrieval, but the availability of handmade translation dictionaries depends on the frequency of translations between the respective languages. Moreover, this model requires significant efforts with respect to disambiguation and domain-specific term translations [8, 215]. CL-LSI and CL-KCCA are reported to achieve a high retrieval quality, but their runtime behavior disqualifies them for many practical applications: at the heart of both models is a singular value decomposition of a term-document matrix which has cubic runtime. This is why we chose to compare CL-CNG, CL-ESA, and CL-ASA. All of them are reported to provide a reasonable retrieval quality, they require no manual fine-tuning, pretty few cross-language resources, and they can be scaled to work in a real-world setting. A comparison of these models is also interesting since they operationalize different paradigms for cross-language similarity assessment.

3.2.2 Cross-Language Character n-Gram Model (CL-CNG)

Character n-grams for cross-language information retrieval achieve a remarkable performance in keyword retrieval for languages with syntactical similarities [132]. We expect that this approach extends to measuring the cross-language document similarity between such languages as well. Given a pre-defined alphabet Σ and an $n \in [1, 5]$, a document d is represented as a vector $\mathbf{d}$ whose dimension is in $O(|\Sigma|^n)$. Obviously $\mathbf{d}$ is sparse, since only a fraction of the possible n-grams occur in a given d. In analogy to the VSM, the elements in $\mathbf{d}$ can be weighted according to a standard weighting scheme,

and two documents d and d' can be compared with a standard measure $\varphi(\mathbf{d}, \mathbf{d}')$. Here we choose $\Sigma = \{a, \ldots, z, 0, \ldots, 9\}$, $n = 3$, $tf \cdot idf$-weighting, and the cosine similarity as φ. In what follows, we refer to this model variant as CL-C3G.

3.2.3 Cross-Language Explicit Semantic Analysis (CL-ESA)

In [166], we introduce the CL-ESA model as an extension of the explicit semantic analysis model (ESA) [68, 233]. ESA represents a document d relative to a so-called index collection D_I. More precisely, d is represented by its similarities to the documents in D_I. These similarities, in turn, are computed with a monolingual retrieval model such as the VSM [208]:

$$\mathbf{d}_{|D_I} = A_{D_I}^T \cdot \mathbf{d}_{\text{VSM}},$$

where $A_{D_I}^T$ denotes the matrix transpose of the term-document matrix of D_I, and $\mathbf{d}_{\text{VSM}}$ denotes the term vector representation of d. Again, various term weighting schemes may be applied.

If a second index collection D_I' in another language is given such that the documents in D_I' have a topical one-to-one correspondence to the documents in D_I, the ESA representations in both languages become comparable. For example, the cross-language similarity between d and d' can be expressed as $\varphi(\mathbf{d}_{|D_I}, \mathbf{d}'_{|D_I'})$. Figure 3.3 illustrates this principle for two languages but CL-ESA naturally extends to multiple languages. Moreover, it gets by without translation technology, be it dictionary-based or other. The model requires merely a comparable corpus of documents written in different languages about similar topics. These documents may still be written independently of each other. An example for such a corpus is Wikipedia, where numerous concepts are covered in many languages.

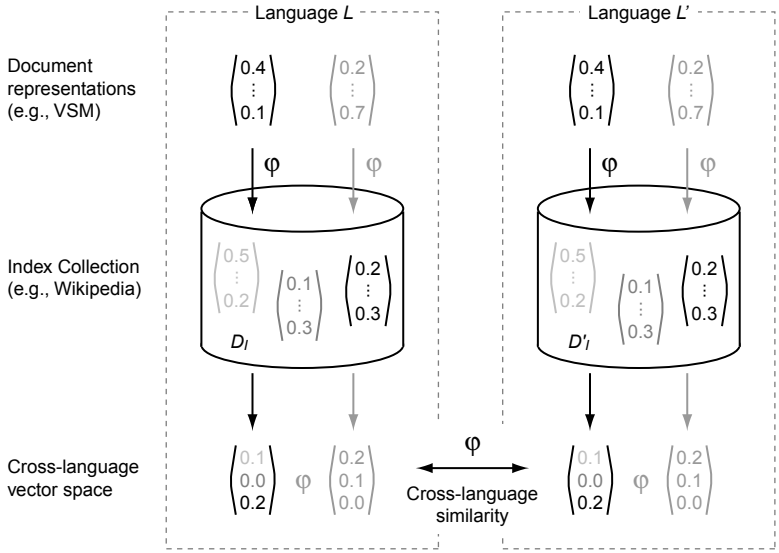


Figure 3.3: Illustration of cross-language explicit semantic analysis.

3.2.4 Cross-Language Alignment-based Similarity Analysis (CL-ASA)

The CL-ASA model is based on statistical machine translation technology; it combines a two-step probabilistic translation and similarity analysis [10]. Given d_q, written in L, and a document d' from a collection D' written in L', the model estimates the probability that d' is a translation of d_q is estimated according to Bayes' rule:

$$p(d' \mid d_q) = \frac{p(d')\ p(d_q \mid d')}{p(d_q)}.$$

$p(d_q)$ does not depend on d' and hence is neglected. From a machine translation viewpoint $p(d_q \mid d')$ is known as *translation model probability*; it is computed using a statistical bilingual dictionary. $p(d')$ is known as *language model probability*; the underlying language model

represents the target language L' in order to obtain grammatically acceptable text in a translation [34]. Since the goal is to measure the similarities of d_q with documents written in L', and not actually translating d_q into L', two adapted sub-models are proposed: (1) the adapted translation model is a non-probabilistic measure $w(d_q \mid d')$, and (2) the language model is replaced by a *length model* $\varrho(d')$, which depends on document lengths instead of language structures. Based on these adaptions the following similarity measure is defined:

$$\varphi(d_q, d') = s(d' \mid d_q) = \varrho(d')\, w(d_q \mid d').$$

Unlike other similarity measures, this one is not normalized; note that the partial order induced among documents resembles the order of other similarity measures. The following paragraphs describe the adapted translation model $w(d_q \mid d')$ and the length model $\varrho(d')$.

Translation Model Adaption The translation model requires a statistical bilingual dictionary. Given the vocabularies of the languages $\mathcal{X} \in L$ and $\mathcal{Y} \in L'$, the bilingual dictionary provides estimates of the translation probabilities $p(x, y)$ for every $x \in \mathcal{X}$ and $y \in \mathcal{Y}$. This distribution expresses the probability for a word x to be a valid translation of a word y. The bilingual dictionary is estimated using the well-known IBM M1 alignment model [34, 147], which has been successfully applied in monolingual and cross-lingual information retrieval tasks [22, 158]. To generate a bilingual dictionary, M1 requires a sentence-aligned parallel corpus.[2] The translation probability of two texts d and d' is originally defined as:

$$p(d \mid d') = \prod_{x \in d} \sum_{y \in d'} p(x, y),$$

where $p(x, y)$ is the probability that the word x is a translation of the word y. The model was demonstrated to generate good

[2] The estimation is carried out on the basis of the EM algorithm [14, 59]. See [34, 159] for an explanation of the bilingual dictionary estimation process.

Table 3.1: Length factors for translations between the language pairs L-L'. A $\mu > 1$ implies $|d| < |d'|$ for d and its translation d'.

Parameter	en-de	en-es	en-fr	en-nl	en-pl
μ	1.089	1.138	1.093	1.143	1.216
σ	0.268	0.631	0.157	1.885	6.399

sentence translations, but since entire documents of variable lengths are considered, the formula is adapted as follows:

$$w(d \mid d') = \sum_{x \in d} \sum_{y \in d'} p(x, y).$$

The weight $w(d \mid d')$ increases if valid translations (x, y) appear in the implied vocabularies. For a word x with $p(x, y) = 0$ and for all $y \in d'$, $w(d \mid d')$ is decreased by 0.1.

Length Model Though it is unlikely to find a pair of translated documents d and d' such that $|d| = |d'|$, their lengths will be closely related by a certain length factor for each language pair. In accordance with [177], the length model probability is defined as follows:

$$\varrho(d') = \exp\left(-0.5\left(\frac{(|d'|/|d|) - \mu}{\sigma}\right)^2\right),$$

where μ and σ are the average and the standard deviation of the character lengths between translations of documents from L to L'. Observe that in cases where a translation d' of a document d_q has not the expected length, the similarity $\varphi(d_q, d')$ is reduced. Table 3.1 lists the values for μ and σ that are used in our evaluation; these values have been estimated using the JRC-Acquis corpus.

3.3 Evaluating Cross-Language Text Similarity Models

In our evaluation, we evaluate and compare CL-C3G, CL-ESA, and CL-ASA in a ranking task. Three experiments are conducted on two test collections with each model and over all language pairs whose first language is English and whose second language is one of Spanish, German, French, Dutch, and Polish. In total, more than 100 million similarities are computed with each model.

3.3.1 Corpora for Training and Evaluation

To train the retrieval models and to test their performance we extracted large collections from the parallel corpus JRC-Acquis and the comparable corpus Wikipedia. The JRC-Acquis Multilingual Parallel Corpus comprises legal documents from the European Union which have been translated and aligned with respect to 22 languages [216]. The Wikipedia encyclopedia is considered to be a comparable corpus since it comprises documents from more than 200 languages which are linked across languages in case they describe the same topic [166]. From these corpora only those documents are considered for which aligned versions exist in all of the aforementioned languages: JRC-Acquis contains 23 564 such documents, and Wikipedia contains 45 984 documents, excluding articles that are lists of things or that describe a date.[3]

The extracted documents from both corpora are divided into a training collection that is used to train the respective retrieval model, and a test collection that is used in the experiments (4 collections in total). The JRC-Acquis test collection and the Wikipedia test collection contain 10 000 aligned documents each, and the corresponding training collections contain the remainder. In total, the test collec-

[3] If only pairs of languages are considered, many more aligned documents can be extracted from Wikipedia, e.g., currently more than 200 000 between English and German.

tions comprise 120 000 documents: 10 000 documents per corpus $\times$ 2 corpora $\times$ 6 languages. As described above, CL-ESA requires the comparable Wikipedia training collection as index collection, whereas CL-ASA requires the parallel JRC-Acquis training collection to train bilingual dictionaries for all of the considered language pairs. CL-C3G requires no training.

3.3.2 Comparing CL-C3G, CL-ESA, and CL-ASA

Let d_q be a query document from a test collection D, let D' be the documents aligned with those in D, and let d'_q denote the document that is aligned with d_q. The following experiments have been repeated for 1 000 randomly selected query documents with all three retrieval models on both test collections, averaging the results.

Experiment 1: Cross-language Ranking Given d_q, all documents in D' are ranked according to their cross-language similarity to d_q; the retrieval rank of d'_q is recorded. Ideally, d'_q should be on the first or, at least, on one of the top ranks. This experiment resembles the situation of cross-language text reuse detection in which (a passage of) a document is given and its translation has to be retrieved from a collection of documents. The results of the experiment are shown in Figure 3.4 as recall-over-rank plots.

Observe that CL-ASA achieves near-perfect performance on the JRC-Acquis test collection, while its performance on the Wikipedia test collection is poor for all language pairs. CL-ESA achieves medium to good performance on both collections, dependent on the language pair, and so does CL-C3G, which outperforms CL-ESA in most cases. With respect to the different language pairings, all models vary in their performance, but, except for CL-ASA and CL-C3G on the English-Polish portion of JRC-Acquis (bottom right plot), the performance characteristics are the same on all language pairs.

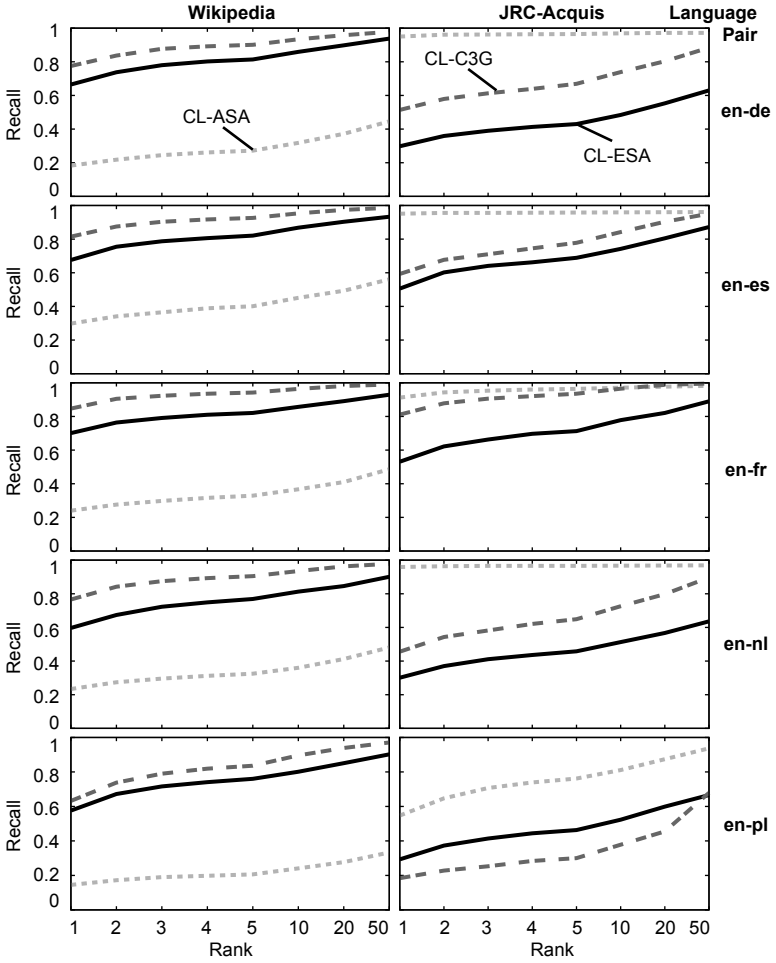


Figure 3.4: Results of Experiment 1.

It follows that CL-ASA has, in general, a large variance in its performance, while CL-ESA and CL-C3G show a stable performance across the corpora. Remember that JRC-Acquis is a parallel corpus while Wikipedia is a comparable corpus, so that CL-ASA seems to be working much better on translations than on comparable documents. Interestingly, CL-ESA and CL-C3G work better on comparable documents than on translations. An explanation for these findings is that the JRC-Acquis corpus is biased to some extent; it contains only legislative texts from the European Union and is hence pretty homogeneous. In this respect, both CL-ESA and CL-C3G appear much less susceptible than CL-ASA, while the latter may perform better when trained on a more diverse parallel corpus. The Polish portion of JRC-Acquis seems to be a problem for both CL-ASA and CL-C3G, but less so for CL-ESA, which shows that the latter can cope even with less related languages.

Experiment 2: Bilingual Rank Correlation Given a pair of aligned documents $d_q \in D$ and $d'_q \in D'$, the documents from D' are ranked twice: (1) with respect to their cross-language similarity to d_q using one of the cross-language retrieval models, and, (2) with respect to their monolingual similarity to d'_q using the vector space model. The top 100 ranks of the two rankings are compared using Spearman's ρ, a rank correlation coefficient which measures the disagreement and agreement of rankings as a value between -1 and 1. This experiment relates to "diagonalization:" a monolingual reference ranking is compared to a cross-lingual test ranking. This experiment can be considered as a standard ranking task where documents have to be ranked according to their similarity to a document written in another language. The results of the experiment are reported as averaged rank correlations in Table 3.2.

Table 3.2: Results of Experiment 2.

Language	Wikipedia			JRC-Acquis		
Pair	CL-ASA	CL-ESA	CL-C3G	CL-ASA	CL-ESA	CL-C3G
en-de	0.14	0.58	0.37	0.47	0.31	0.28
en-es	0.18	0.17	0.10	0.66	0.51	0.42
en-fr	0.16	0.29	0.20	0.38	0.54	0.55
en-nl	0.14	0.17	0.11	0.58	0.33	0.31
en-pl	0.11	0.40	0.22	0.15	0.35	0.15

As in Experiment 1, CL-ASA performs good on JRC-Acquis and unsatisfactory on Wikipedia. In contrast to Experiment 1, CL-ESA performs similar to both CL-CNG and CL-ESA on JRC-Acquis with respect to different language pairs, and it outperforms CL-ASA on Wikipedia. Again, unlike in the first experiment, CL-C3G is outperformed by CL-ESA. With respect to the different language pairings all models show weaknesses (e.g., CL-ASA on English-Polish and, CL-ESA as well as CL-C3G on English-Spanish and English-Dutch). It follows that CL-ESA is more applicable as a general purpose retrieval model than are CL-ASA or CL-C3G, while special care needs to be taken with respect to the involved languages. We argue that the reason for the varying performance is rooted in the varying quality of the employed language-specific indexing pipelines and not in the retrieval models themselves.

Experiment 3: Cross-Language Similarity Distribution This experiment contrasts the similarity distributions of comparable documents and parallel documents. It shall give an idea about what can be expected from each retrieval model; the experiment cannot directly be used to compare the models or to tell something about their quality. Rather, it tells something about the range of cross-language similarity values one will measure when using the model, in particular, which values indicate a high similarity and which values indicate a low similarity. The results of the experiment are shown in Figure 3.5 as plots of ratio of similarities-over-similarity intervals.

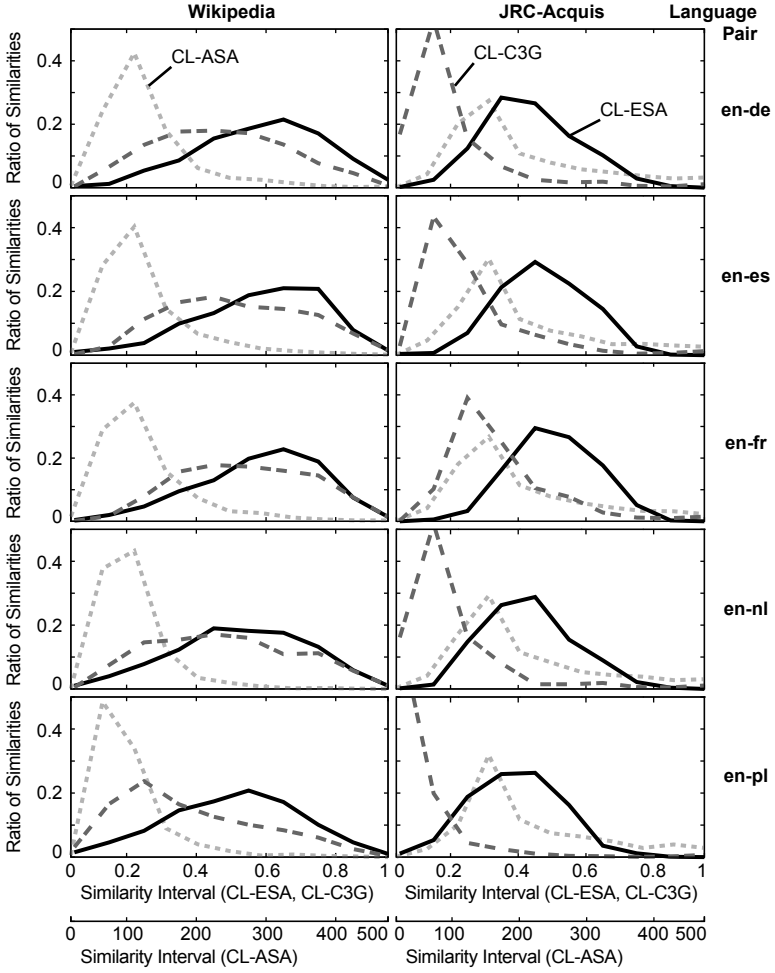


Figure 3.5: Results of Experiment 3.

Observe that the similarity distributions of CL-ASA has been plotted on a different scale than those of CL-ESA and CL-C3G: the top x-axis of the plots shows the range of similarities measured with CL-ASA, the bottom x-axis shows the range of similarities measured with the other models. This is necessary since the similarities computed with CL-ASA are not normalized. The absolute values measured with the three retrieval models are not important, but the order they induce among the compared documents is. In fact, this holds for every retrieval model, be it cross-lingual or no. This is also why the similarity values computed with two models cannot be compared to one another: for example, the similarity distribution of CL-ESA looks "better" than that of CL-C3G because it is more to the right, but in fact, CL-C3G outperforms CL-ESA in Experiment 1.

3.3.3 Adjusting CL-ESA

The following three experiments shed light on how to adjust CL-ESA. Moreover, we provide insights about how many languages CL-ESA can represent at the same time, and how fast it can be built.

Experiment 4: Dimensionality CL-ESA's retrieval quality and runtime depend on its dimensionality, which in turn corresponds the size of its index collection $|D_I|$. In this experiment, we repeat Experiments 1, 2, and 3 while varying the dimensionality from 10 to 100 000 index documents. Figure 3.6 shows the results of this experiment. As can be seen, the dimensionality $|D_I|$ of CL-ESA is its most important parameter: it can be used to seamlessly adjust the retrieval performance of CL-ESA from absolute failure to near perfection. The recall at rank plots for Experiment 1 show this behavior. The results for Experiment 2 show that the rank correlation of CL-ESA and the vector space model increases with the number of dimensions, however, perfect correlation is not reached. This is not too surprising as CL-ESA and ESA, by design, incorporates more

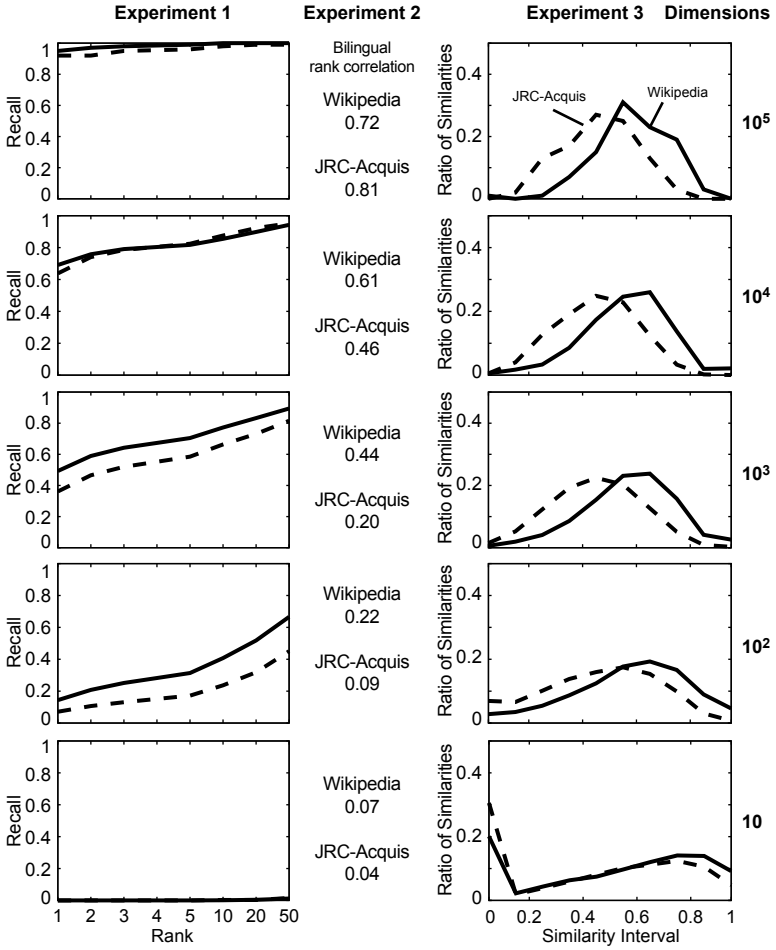


Figure 3.6: Results of Experiment 4.

information into its representation than the vector space model does, so that differences within the top ranks are to be expected. Hence, the less than perfect correlation values at high dimensionalities do not necessarily indicate a bad performance of CL-ESA. The similarity distributions of CL-ESA at different dimensionalities reveal that the smaller the dimensionality is, the more similar become the measured similarities, whereas at 10 dimensions, many similarities are 0. The higher the dimensionality, the steeper the distributions become, centering around an expected similarity of 0.5 to 0.6.

Experiment 5: Multilingualism CL-ESA is a multilingual retrieval model in that it is capable of representing documents from more than two languages within the same vector space, allowing for immediate comparisons. This property fully depends on the underlying index collections from the languages in question, and how well their topic alignment is. Starting with the two most prominent languages in Wikipedia, English and German, we evaluate how many concepts are described in both languages, and how many remain in the intersection set if more languages are considered, ordered by their total number of articles in Wikipedia. The left plot in Figure 3.7 shows the results of this experiment. Unsurprisingly, the number of concepts for which articles are available in all three languages English, German, and French is smaller than for English and German alone. When adding more languages, the overlap decreases exponentially. In particular, a strong reduction can be observed when Japanese is added to the set of languages, and another when Chinese is added. This indicates that "cultural distance" between languages is a strong factor. Once the portion of Wikipedia written in the constructed language Volapük is added to the set, no intersecting concepts could be found anymore. Mind, however, that this plot may not show the complete picture: if the languages in Wikipedia are not considered ordered by their size but by geographical, cultural, or linguistic relations, there may be more intersecting concepts in the respective

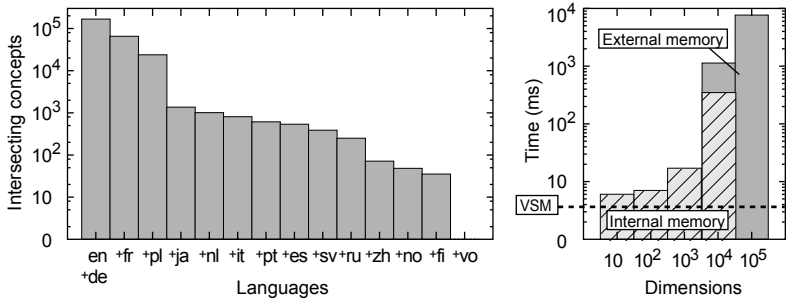


Figure 3.7: Left: number of intersecting concepts between Wikipedia's languages. The languages are organized in descending order of the number of available articles. Right: average time to index a document under ESA, depending on the number of dimensions.

groups. And if only two languages are considered, the number of shared concepts between the English portion of Wikipedia and a non-English portion and will be much higher in most cases.

Experiment 6: Indexing Time Since much depends on the size of the index collection D_I underlying CL-ESA, which has an immediate effect of the time to index a document, we conducted runtime experiments as well. The right plot in Figure 3.7 shows the averaged runtimes, dependent on the size of the index collection. The time to index a document is between 10 to 100 milliseconds at small dimensionalities, which is comparable to the time to compute a vector space representation (cf. Figure 3.7, right plot). With increasing dimensionality, and especially if one has to resort to external memory, the time to index a documents increase exponentially. Employed hardware: Intel Core 2 Duo processor at 2 GHz and 1 GB RAM.

Discussion If, for example, a high retrieval quality is desired, documents should be represented as 10^5-dimensional concept vectors: ranking with respect to a particular query document will provide similar documents on the top ranks with high accuracy. High retrieval quality comes at the price that with the current Wikipedia corpus only 2 languages can be represented at the same time, and that the time to index a document will be high. If high retrieval speed or a high multilingualism is desired, documents should be represented as 1000-dimensional concept vectors. At a lower dimension, the retrieval quality deteriorates significantly. A reasonable trade-off between retrieval quality and runtime is achieved for a concept space dimensionality between 1 000 and 10 000.

3.3.4 The Difficulties of Cross-language Fingerprinting

In Section 2.1 we have introduced fingerprinting as a technology for the task of near-duplicate detection. The question remains whether fingerprinting can also be applied across languages. In [6] we found that it can't, but first things first: is there even such a thing as a pair of documents from different languages which are near-duplicates?

The notion of near-duplicate documents is a matter of frequent debate within the community, but judging from the proposed solutions (see Table 2.1), the majority of researchers expect a high syntactical similarity (that is, shared word n-grams) between near-duplicate documents while allowing for small differences. The closest thing to syntactical duplication across languages is a metaphrase (i.e., a literal word-by-word translation). However, since most pairs of languages do not have a common vocabulary, no syntactical overlaps beyond character n-grams can be expected for such translations. Also, most translations are not metaphrases but paraphrases, so that we resort to the definition of a paraphrase as a working definition for a pair of cross-language near-duplicate documents: two documents (from different languages) which are semantically equivalent.

A prerequisite for fingerprinting is a retrieval model which represents documents in way so that the representations can be used to check pairs of documents for being near-duplicate or no. The cross-language retrieval models introduced in this chapter may be used for this purpose, but they cannot be combined with all of the available fingerprint algorithms. In fact, because of the lack of syntactical overlap between languages, only one technique can be used: the locality-sensitive hashing (LSH) framework. LSH is a context-free framework, since it is entirely rooted in algebra. Its purpose is to map vectors, which are similar according to a certain similarity or distance measure, onto the same hash value. Together with a retrieval model which produces vector representations for documents, LSH is used to identify near-duplicates.

Given a query document d_q written in language L and a (very large) collection of documents D' written in L', the task of cross-language near-duplicate detection is to retrieve a subset $D'_q \subset D'$, containing all near-duplicates of d_q, say:

$$d' \in D'_q \;\; \Rightarrow \;\; \varphi(\mathbf{d}_q, \mathbf{d}') \geq 1 - \epsilon,$$

where φ is a similarity measure and $\mathbf{d}_q$ and $\mathbf{d}'$ are vector representations of d_q and d' under a cross-language retrieval model. D'_q is called the ϵ-neighborhood of d_q. To accomplish this task, fingerprints F_{d_q} and F_d are constructed which comprise a number k of hash values computed with k different similarity sensitive hash functions $h_\varphi : \mathbf{D}' \rightarrow \mathbf{N}$, where $\mathbf{D}'$ the set of cross-language vector representations of D'. If F_{d_q} and F_d share at least some of their hash values, d_q and d are considered near-duplicates.

A reasonable setting for ε may be 0.4, which can be derived from the results of Experiment 3 in Section 3.3.2 (see Figure 3.5). At $\varepsilon = 0.4$, a maximum recall of about 0.5 can be expected. However, when comparing this with the results of our fingerprinting experiments in Chapter 2, it becomes obvious that cross-language fingerprinting cannot be realized this way: locality-sensitive hashing

works only if the vector similarities of near-duplicates are above 0.9 (see Figure 2.6).[4] In conclusion, unless a retrieval model is conceived which maps near-duplicates onto vectors whose algebraic similarity is above 0.9, fingerprinting cannot be applied across languages.

3.3.5 Conclusions and Future Work

Cross-language text reuse is an important direction of research on detecting text but is still in its infancy. We pointed out a basic retrieval strategy for this task, including two important subtasks which require special attention: multilingual candidate retrieval for text reuse from the web, and the detailed comparison of two documents across languages. With respect to the former, well-known and less well-known state-of-the-art research is reviewed. With respect to the latter, we survey existing retrieval models and describe three of them in detail, namely the cross-language character n-gram model (CL-CNG), the cross-language explicit semantic analysis (CL-ESA) and the cross-language alignment-based similarity analysis (CL-ASA). For these models we report on a large-scale comparative evaluation. Moreover, we conduct an in-depth evaluation of CL-ESA.

The evaluation covers 6 experiments on two aligned corpora: the comparable Wikipedia corpus and the parallel JRC-Acquis corpus. In the experiments the models are employed in different tasks related to cross-language ranking in order to determine whether or not they can be used to retrieve highly similar documents across languages. Our findings include that the CL-C3G model and the CL-ESA model are in general better suited for this task, while CL-ASA achieves good results on professional and automatic translations. CL-CNG outperforms CL-ESA and CL-ASA. However, unlike the

[4] Theoretically, the LSH framework can be adjusted to support a reliable detection of near-duplicates even if their vector representations have low absolute similarity values, however, this comes at the price of precision and a significantly higher dimensionality of the document fingerprints, which altogether defeats the goal of fingerprinting to improve the retrieval runtime of near-duplicate detection.

former, CL-ESA and CL-ASA can also be used on syntactically different languages. The experiments specific to the parameters of CL-ESA reveal, that the model can be flexibly adjusted to meet requirements such as high quality or high speed as well as a tradeoff between the two. CL-ESA can also be used to represent documents from more than two languages at the same time.

Chapter 4

Evaluating Plagiarism Detectors

Research and development on automatic plagiarism detection is one of the most prominent topics in the broad field of text reuse. Various papers have been published on the topic, and many commercial software systems are being developed. However, when asked to name the best algorithm or the best system for plagiarism detection, hardly any evidence can be found to make an educated guess among the alternatives. One reason for this is that the research field of text reuse and plagiarism detection lacks a controlled evaluation framework. The lack of an evaluation framework is a serious problem for every empirical research field. We have addressed this shortcoming for plagiarism detection in the context of our series of evaluation workshops called PAN. This chapter presents the evaluation framework we have developed in the past years and the evaluation results obtained, thus compiling an overview of our respective publications [168, 169, 170, 171, 175]. But before going into details, we survey the state of the art in evaluating plagiarism detection, which has not been studied systematically until now.

Table 4.1: Summary of the plagiarism detection evaluations in 205 papers, from which 104 deal with text and 101 deal with code.

Evaluation Aspect	Text	Code
Experiment Task		
local collection	80%	95%
web retrieval	15%	0%
other	5%	5%
Performance Measure		
precision, recall	43%	18%
manual, similarity	35%	69%
runtime only	15%	1%
other	7%	12%
Comparison		
none	46%	51%
parameter settings	19%	9%
other algorithms	35%	40%

Evaluation Aspect	Text	Code
Corpus Acquisition		
existing corpus	20%	18%
homemade corpus	80%	82%
Corpus Size [# documents]		
$[1, 10)$	11%	9%
$[10, 10^2)$	19%	31%
$[10^2, 10^3)$	38%	37%
$[10^3, 10^4)$	8%	18%
$[10^4, 10^5)$	16%	5%
$[10^5, 10^6)$	8%	0%

Survey of Plagiarism Detection Evaluations We have queried academic databases and search engines to get an overview of all kinds of contributions to automatic plagiarism detection. Altogether 275 papers were retrieved, from which 139 deal with plagiarism detection in text, 123 with plagiarism detection in code, and 13 with other media types. From the papers related to text and code, we analyzed the 205 which present evaluations. Our analysis covers the following aspects: experiment tasks, performance measures, underlying corpora, and, whether comparisons to other plagiarism detection approaches were conducted. Table 4.1 summarizes our findings.

With respect to the experiment tasks, the majority of the approaches perform overlap detection by exhaustive comparison against some locally stored document collection—albeit a web retrieval scenario is more realistic. We explain this shortcoming by two facts, namely that the web cannot be utilized easily as a corpus, and that in the case of code plagiarism the focus is on collusion detection

in student courseworks. With respect to performance measures, the picture is less clear: a manual result evaluation based on similarity measures is used about the same number of times for text (35%), and even more often for code (69%), as an automatic computation of precision and recall. 21% and 13% of the evaluations on text and code use custom measures or examine only the detection runtime. This indicates that precision and recall may not be well-defined in the context of plagiarism detection. Moreover, comparisons to existing research are conducted in less than half of the papers, a fact that underlines the lack of an evaluation framework.

The right-hand side of Table 4.1 overviews two corpus-related aspects: the use of existing corpora versus the use of handmade corpora, and the size distribution of the used corpora. In particular, we found that researchers follow two strategies to compile a corpus. Small corpora (<1 000 documents) are built from student courseworks or from arbitrary documents into which plagiarism-alike overlap is manually inserted. Large corpora (>1 000 documents) are collected from sources where overlap occurs more frequently, such as rewritten versions of news wire articles, or from consecutive versions of open source software. Altogether, we see a need for an open, commonly used plagiarism detection corpus.

Related Work There are some surveys about automatic plagiarism detection in text [42, 44, 130] and in code [86, 191, 193, 194]. These papers, as well as nearly all papers from our survey, omit a discussion of evaluation methodologies; the following four papers are an exception. In [229] the authors introduce graph-based performance measures for code plagiarism detection intended for unsupervised evaluations. We argue that evaluations in this field should be done in a supervised manner, while the proposed measures have not been adopted since their first publication. In [169] we introduce preliminary parts of our framework. The focus of that paper, however, is

more on the comparison of the detection approaches that were submitted to the first PAN plagiarism detection competition. In [45, 46] the authors report on an unnamed corpus that comprises 57 cases of simulated plagiarism. We refer to this corpus as the Clough09 corpus; a comparison to our approach is given later on. Finally, a related corpus is the METER corpus, which has been the only alternative for the text domain up to now [47]. It comprises 445 cases of text reuse among 1 716 news articles. Although the corpus can be used to evaluate plagiarism detection, its structure does not support this task. This might be the reason why it has not been used more often. Furthermore, it is questionable how similar cases of news reuse and plagiarism are, since plagiarists strive to remain undetected.

Contributions Besides the above survey, our contributions are the following: Section 4.1 presents formal foundations for the evaluation of plagiarism detectors and hence text reuse detectors in general, introducing three performance measures. Section 4.2 introduces methods to create artificial and simulated plagiarism cases on a large scale, and the PAN plagiarism corpus in which these methods have been operationalized. We compare our corpus with the Clough09 corpus and the METER corpus, which reveals important insights for the different kinds of text reuse in these corpora. Finally, Section 4.3 reports on the results of the first three editions of our series of PAN evaluation workshops, in which the proposed framework has been successfully employed to evaluate 32 plagiarism detectors.

4.1 Detection Performance Measures

This section introduces measures to quantify the precision and recall performance of a plagiarism detector; we present a micro-averaged and a macro-averaged variant. Moreover, the so-called detection

granularity is introduced, which quantifies whether the contiguity of plagiarized text passages is properly recognized. This concept is important: a low granularity simplifies both the human inspection of algorithmically detected passages as well as an algorithmic style analysis within a potential post-process. The three measures can be applied in isolation but also be combined into a single, overall score called *plagdet*. A reference implementation of the performance measures is published as part of our evaluation framework.

Precision, Recall, and Granularity Let d_{plg} denote a document that contains plagiarism. A *plagiarism case* in d_{plg} is a 4-tuple $s = \langle s_{\text{plg}}, d_{\text{plg}}, s_{\text{src}}, d_{\text{src}} \rangle$, where s_{plg} is a plagiarized passage in d_{plg}, and s_{src} is its original counterpart in some source document d_{src}. Likewise, a *plagiarism detection* for d_{plg} is denoted as $r = \langle r_{\text{plg}}, d_{\text{plg}}, r_{\text{src}}, d'_{\text{src}} \rangle$; r associates an allegedly plagiarized passage r_{plg} in d_{plg} with a passage r_{src} in d'_{src}. We say that r *detects* s iff $r_{\text{plg}} \cap s_{\text{plg}} \neq \emptyset$, $r_{\text{src}} \cap s_{\text{src}} \neq \emptyset$, and $d'_{\text{src}} = d_{\text{src}}$. It is assumed that different plagiarized passages in d_{plg} do not intersect, whereas no such restriction applies for detections in d_{plg}. Finally, S and R denote sets of plagiarism cases and detections.

The above 4-tuples resemble an intuitive view of plagiarism detection, but subsequent notations can be simplified using an equivalent, more concise view: a document d is represented as a set of references to its characters $\mathbf{d} = \{(1, d), \ldots, (|d|, d)\}$, where (i, d) refers to the i-th character in d. A plagiarism case s is then represented as $\mathbf{s} = \mathbf{s}_{\text{plg}} \cup \mathbf{s}_{\text{src}}$, where $\mathbf{s}_{\text{plg}} \subseteq \mathbf{d}_{\text{plg}}$ and $\mathbf{s}_{\text{src}} \subseteq \mathbf{d}_{\text{src}}$. The character references in $\mathbf{s}_{\text{plg}}$ and $\mathbf{s}_{\text{src}}$ form the passages s_{plg} and s_{src}. Likewise, a detection r is represented as $\mathbf{r} = \mathbf{r}_{\text{plg}} \cup \mathbf{r}_{\text{src}}$. It follows that r detects s iff $\mathbf{r}_{\text{plg}} \cap \mathbf{s}_{\text{plg}} \neq \emptyset$ and $\mathbf{r}_{\text{src}} \cap \mathbf{s}_{\text{src}} \neq \emptyset$. This way,

precision and recall of R under S can be measured micro-averaged:

$$prec_{\text{micro}}(S,R) = \frac{\left|\bigcup_{(s,r)\in(S\times R)}(\mathbf{s}\sqcap\mathbf{r})\right|}{\left|\bigcup_{r\in R}\mathbf{r}\right|}, \tag{4.1}$$

$$rec_{\text{micro}}(S,R) = \frac{\left|\bigcup_{(s,r)\in(S\times R)}(\mathbf{s}\sqcap\mathbf{r})\right|}{\left|\bigcup_{s\in S}\mathbf{s}\right|}, \tag{4.2}$$

$$\text{where}\quad \mathbf{s}\sqcap\mathbf{r} = \begin{cases} \mathbf{s}\cap\mathbf{r} & \text{if } r \text{ detects } s, \\ \emptyset & \text{otherwise.} \end{cases}$$

Similarly, precision and recall can be measured macro-averaged:

$$prec_{\text{macro}}(S,R) = \frac{1}{|R|}\sum_{r\in R}\frac{\left|\bigcup_{s\in S}(\mathbf{s}\sqcap\mathbf{r})\right|}{|\mathbf{r}|}, \tag{4.3}$$

$$rec_{\text{macro}}(S,R) = \frac{1}{|S|}\sum_{s\in S}\frac{\left|\bigcup_{r\in R}(\mathbf{s}\sqcap\mathbf{r})\right|}{|\mathbf{s}|}, \tag{4.4}$$

The former measures are simpler to be computed, whereas the latter have the advantage to be unaffected by a plagiarism case's length.

Besides precision and recall there is another concept that characterizes the power of a detection algorithm, namely, whether a plagiarism case $s \in S$ is detected as a whole or in several pieces. The latter can be observed in today's commercial plagiarism detectors, and the user is left to combine these pieces into a consistent approximation of s. Ideally, an algorithm should report detections R in a one-to-one manner to the true cases S. To capture this characteristic we define the detection granularity of R under S:

$$gran(S,R) = \frac{1}{|S_R|}\sum_{s\in S_R}|R_s|, \tag{4.5}$$

where $S_R \subseteq S$ are plagiarism cases detected by detection in R, and $R_s \subseteq R$ are detections that detect a given s:

$$S_R = \{s \mid s \in S \wedge \exists r \in R : r \text{ detects } s\},$$
$$R_s = \{r \mid r \in R \wedge r \text{ detects } s\}.$$

The domain of $gran(S, R)$ is $[1, |R|]$, with 1 indicating the desired one-to-one correspondence and $|R|$ indicating the worst case, where a single $s \in S$ is detected over and over again.

Precision, recall, and granularity allow for a partial ordering among plagiarism detection algorithms. To obtain an absolute order they must be combined to an overall score:

$$plagdet(S, R) = \frac{F_\alpha}{\log_2(1 + gran(S, R))}, \tag{4.6}$$

where F_α denotes the F-Measure (i.e., the weighted harmonic mean of precision and recall). We suggest using $\alpha = 1$ where precision and recall are equally weighted, since there is currently no indication that either of the two is more important. We take the logarithm of the granularity to decrease its impact on the overall score.

Discussion Plagiarism detection is both a retrieval task and an extraction task. In light of this fact, not only retrieval performance but also extraction accuracy becomes important, the latter of which being neglected in the literature. Our measures incorporate both. Another design objective of our measures is minimization of restrictions imposed on plagiarism detectors. While plagiarism cases within a document are unlikely to have more than one source, imprecision or lack of evidence may cause humans or algorithms to report overlapping detections (e.g., when being unsure about the true source of a plagiarized passage). The measures (4.1)-(4.4) provide for a sensible treatment of this fact since the set-based passage representations eliminate duplicate detections of characters. Finally, the macro-averaged variants allot equal weight to each plagiarism case, regardless of its length. Conversely, the micro-averaged variants favor the detection of long plagiarism passages, which are generally easier to be detected. Which of these variants is to be preferred, however, is still an open question.

4.2 An Evaluation Corpus for Plagiarism Detectors

This section organizes and analyzes the practices that are employed—most of the time implicitly—for the construction of plagiarism corpora. We introduce three levels of *plagiarism authenticity*, namely real, simulated, and artificial plagiarism. It turns out that simulated plagiarism and artificial plagiarism are the only viable alternatives for corpus construction. We propose a new approach to scale up the generation of simulated plagiarism based on crowdsourcing, and heuristics to generate artificial plagiarism. Moreover, based on these methods, we compile the PAN plagiarism corpus which is the first corpus of its kind that contains both a large number and a high diversity of simulated and artificial plagiarism cases.

4.2.1 Real, Simulated, and Artificial Plagiarism

Syntactically, a plagiarism case is the result of copying a passage s_{src} from a source document into another document d_{plg}. Since verbatim copies can be detected easily, plagiarists often modify s_{src} to obfuscate their illegitimate act. This behavior must be modeled when constructing a corpus for plagiarism detection, which can be done at three levels of authenticity. Ideally, one would secretly observe many plagiarists and use their *real plagiarism* cases; realistically, one could resort to plagiarism cases which have been detected in the past. The following aspects object against this approach:

- The distribution of detected real plagiarism is skewed towards ease of detectability.
- The acquisition of real plagiarism is expensive since it is often concealed from the public.
- Publishing real plagiarism possibly requires consents from the plagiarist and the original author.

- A public corpus of real plagiarism cases is questionable from an ethical and legal viewpoint.
- Anonymizing real plagiarism is rendered difficult due to web search engines and author identification technologies.

It is hence more practical to let people create plagiarism cases by "purposeful" modifications, or to tap resources that contain similar kinds of text reuse. We subsume these strategies under the term *simulated plagiarism*. The first strategy has often been applied in the past, though on a small scale and without a public release of the corpora; the second strategy comes in the form of the METER corpus [47]. Note that, from a psychological viewpoint, people who simulate plagiarism act under a different mental attitude than plagiarists. From a linguistic viewpoint, however, it is unclear whether real plagiarism differs from simulated plagiarism.

A third possibility is to generate plagiarism algorithmically [37, 169, 192], which we call *artificial plagiarism*. Generating artificial plagiarism cases is a non-trivial task if one requires semantic equivalence between a source passage s_{src} and the passage s_{plg} obtained by an automatic obfuscation of s_{src}. Such semantics-preserving algorithms are still in their infancy; however, the similarity computation between texts is usually done on the basis of document models like the bag of words model and not on the basis of the original text, which makes obfuscation amenable to simpler approaches.

4.2.2 Creating Simulated Plagiarism

Our approach to scale up the creation of simulated plagiarism is based on Amazon's Mechanical Turk (AMT), a commercial crowdsourcing service [9]. This service has gathered considerable interest among researchers and practitioners alike (e.g., to recreate TREC assessments [3], but also to write and translate texts [5]). We offered the following task on AMT: *Rewrite the original text found below* [on

Table 4.2: Summary of 4 000 Mechanical Turk tasks done by 907 workers.

Worker Demographics			
Age		*Education*	
18, 19	10%	HS	11%
20–29	37%	College	30%
30–39	16%	BSc.	17%
40–49	7%	MSc.	11%
50–59	4%	Dr.	2%
60–69	1%		
n/a	25%	n/a	29%
Native Speaker		*Gender*	
yes	62%	male	37%
no	14%	female	39%
n/a	23%	n/a	24%
Prof. Writer		*Plagiarized*	
yes	10%	yes	16%
no	66%	no	60%
n/a	24%	n/a	25%

Task Statistics	
Tasks per Worker	
average	15
std. deviation	20
minimum	1
maximum	103
Work Time (minutes)	
average	14
std. deviation	21
minimum	1
maximum	180
Compensation	
pay per task	0.5 US$
rejected results	25%

the task webpage] *so that the rewritten version has the same meaning as the original, but with a different wording and phrasing. Imagine a scholar copying a friend's homework just before class, or imagine a plagiarist willing to use the original text without proper citation.* We furthermore disclosed to the worker that our aim with this task was research.

Workers were required to be fluent in English reading and writing, and they were informed that every result was to be reviewed. A questionnaire displayed alongside the task description asked about worker demographics, such as age, gender, education, etc. In particular, we asked whether the worker has ever plagiarized. Completing the questionnaire was optional in order to minimize false answers, but still, these numbers have to be taken with a grain of salt: the Mechanical Turk is not the best platform for surveys. Table 4.2 overviews the worker demographics and task statistics. The average worker appears to be a well-educated male or female

in the twenties whose mother tongue is English. 16% of the workers claim to have plagiarized, and if at least the order of magnitude can be taken seriously this shows that plagiarism is a prevalent problem. A number of pilot experiments were conducted to determine the pay per task, depending on the text length and the task completion time: for 50 US-cents about 500 words get rewritten in about half an hour. We observed that changing the pay per task has proportional effect on the task completion time, but not on result quality. This observation is in concordance with earlier research [129].

The top row of Table 4.3 contrasts a source passage and its rewritten, plagiarized passage obtained via Mechanical Turk. As can be seen, the edits made are substantial while the semantics have been entirely preserved. In this example, the discourse remains unchanged, while expressions are less concise. The worker replaced certain concepts and phrases with synonyms or equivalent figures of speech. It must be noted, however, that a number of typos and grammar errors can be found (i.e., we copied the text without change from the results submitted). For example, the worker writes "enough man" instead of "enough men," "St. John" instead of "St. Johns," and "degrees to the north latitude" instead of "degrees north latitude." Partly, such errors are slips of the pen as workers often do not double-check their work, and partly they arise from a lack of knowledge about the concepts discussed in the source passage. Though we have manually reviewed all of the 4 000 texts that have been rewritten via Mechanical Turk, in each text, we have done only spot checks to verify that it is English and about the same topic as the original. We have not checked spelling and grammar so that it is yet unknown how prevalent such errors are in the corpus.

Table 4.3: Example of a simulated plagiarism case via Mechanical Turk, and two artificial plagiarism cases.

Source Passage	**Simulated Plagiarism** (AMT)
The emigrants who sailed with Gilbert were better fitted for a crusade than a colony, and, disappointed at not at once finding mines of gold and silver, many deserted; and soon there were not enough sailors to man all the four ships. Accordingly, the Swallow was sent back to England with the sick; and with the remainder of the fleet, well supplied at St. John's with fish and other necessaries, Gilbert (August 20) sailed south as far as forty-four degrees north latitude. Off Sable Island a storm assailed them, and the largest of the vessels, called the Delight, carrying most of the provisions, was driven on a rock and went to pieces. [Excerpt from "*Abraham Lincoln: A History*" by John Nicolay and John Hay.]	The people who left their countries and sailed with Gilbert were more suited for fighting the crusades than for leading a settled life in the colonies. They were bitterly disappointed as it was not the America that they had expected. Since they did not immediately find gold and silver mines, many deserted. At one stage, there were not even enough man to help sail the four ships. So the Swallow was sent back to England carrying the sick. The other fleet was supplied with fish and the other necessities from St. John. On August 20, Gilbert had sailed as far as forty-four degrees to the north latitude. His ship known as the Delight, which bore all the required supplies, was attacked by a violent storm near Sable Island. The storm had driven it into a rock shattering it into pieces.
Artificial Plagiarism (low obfuscation)	**Artificial Plagiarism** (high obfuscation)
Other Sable Island a storm assailed them, and the largest of the vessels, called the Delight, carrying most of the provisions, was driven on a rock and went to pieces. Accordingly, the immerse was sent back to England with the disappointed; and with the remainder of the fleet, soon supplied at St. John's with pisces and off necessaries, Gilbert (August 20) sailed south as sick as forty-four degrees north latitude. The emigrants who sailed with Gilbert were better fitted for a crusade than a colony, and, far at not at once finding excavation of gold and silver, many deserted; and well there were not enough sailors to man all the four ships.	The emigrants who sailed with John were better supplied for the gold than a colony, and, deserted at not at once happen excavation of crusade silver, many disappointed; and well there were not enough sailors to man all most four derelict. Soon, the swallow was assailed back to England with the other; and with the remainder of a fish, accordingly fitted at St. Gilbert' Room with fleet and sick necessaries, Gilbert (August 20) sailed south as far as forty-four degrees north. Off Sable Island a storm sent them, and the largest of the vessels, name the Delight, carrying the congregation of a provisions, was driven on the rock and went to pieces.

4.2.3 Creating Artificial Plagiarism

To create an artificial plagiarism case s_{plg} from a given source passage s_{src}, we employ three obfuscation strategies:

- *Random Text Operations.* s_{plg} is created from s_{src} by shuffling, removing, inserting, or replacing words or short phrases at random. Insertions and replacements are taken from the document d_{plg} where s_{plg} is to be inserted.
- *Semantic Word Variation.* s_{plg} is created from s_{src} by replacing words with their synonyms, antonyms, hyponyms, or hypernyms, chosen at random. A word is kept if none are available.
- *POS-preserving Word Shuffling.* The sequence of parts of speech in s_{src} is determined and s_{plg} is created by shuffling words at random while retaining the original POS sequence.

The strategies are adjusted by varying the number of operations made on s_{src}, and by limiting the range of affected phrases, allowing for different degrees of obfuscation. For our corpus, we adjusted them to match an intuitive understanding of a "low" and a "high" obfuscation. The bottom row of Table 4.3 shows two examples of artificial plagiarism which have been obfuscated using different combinations of the above strategies, when applying them repeatedly on the source passage. Some additional heuristics are used to ensure that the texts do not contain obvious defects, such as capitalized words in the middle of a sentence or punctuation marks following each other closely. Unsurprisingly, however, these strategies do not produce text which is grammatically correct or semantically equivalent to the source passage. Of course other obfuscation strategies are conceivable (e.g., based on automatic paraphrasing methods [11]), but the above strategies were preferred for reasons of performance and since many current plagiarism detection algorithms treat text as bag of words (i.e., disregarding word order).

Table 4.4: Corpus statistics of the PAN plagiarism corpus in the versions 2009, 2010, and 2011 (i.e., PAN-PC-09, PAN-PC-10, and PAN-PC-11). The corpora consist of 41 223, 27 073, and 26 939 documents as well as 94 217, 68 566, and 61 069 plagiarism cases, respectively.

Document Statistics				
Year / Version		2009	2010	2011
Document Purpose				
source documents		50%	50%	50%
suspicious documents				
– with plagiarism		25%	25%	25%
– w/o plagiarism		25%	25%	25%
Detection Task				
external detection		70%	100%	82%
intrinsic detection		30%		18%
Plagiarism per Document (fraction)				
hardly	(0.05-0.2)	45%	45%	57%
medium	(0.2-0.5)	15%	15%	15%
much	(0.5-0.8)	25%	25%	18%
entirely	(>0.8)	15%	15%	10%
Document Length (pages)				
short	(1-10)	50%	50%	50%
medium	(10-100)	35%	35%	35%
long	(100-1000)	15%	15%	15%

Plagiarism Case Statistics			
Year / Version	2009	2010	2011
Obfuscation (degree/tool)			
none	49%	40%	18%
artificial			
– paraphrase (low)	28%	22%	32%
– paraphrase (high)	14%	22%	31%
– translation (auto)	9%	12%	10%
simulated			
– paraphrase (AMT)	—	4%	8%
– translation (AMT)*	—	—	1%
Topic Match			
intra-topic cases	—	50%	—
inter-topic cases	—	50%	—
Case Length (words)			
short (50-150)	34%	34%	35%
medium (300-500)	33%	33%	38%
long (3000-5000)	33%	33%	27%

*machine translation, then manual correction

4.2.4 The PAN Plagiarism Corpus

We have employed the aforementioned strategies to generate simulated and artificial plagiarism for the PAN plagiarism corpus. A new version of this corpus was constructed for each of the annual PAN plagiarism detection competitions in 2009, 2010, and 2011, each one improving upon its predecessor. Besides the above obfuscation strategies, several other parameters have been varied (see Table 4.4 for a detailed overview). The documents used for the corpus are derived from books obtained from the Project Gutenberg.[1]

[1] http://www.gutenberg.org

Document Purpose Every document in the corpus serves one of two purposes: it is either used as a source for plagiarism or as a document suspicious of plagiarism. The latter divide into documents which actually contain plagiarism and documents that don't. The documents without plagiarism allow to evaluate whether a detector can distinguish plagiarism cases from natural, accidental overlaps between random pairs of documents.

Detection Task The corpus divides into two parts $(D_{\text{plg}}, D_{\text{src}}, S)$ and (D_{plg}, S), corresponding to the two paradigms of plagiarism detection mentioned in the introduction (see Section 1.1.1): the first part is used for the subtask of external plagiarism detection, the second part for intrinsic plagiarism detection. In external plagiarism detection, one is given a set D_{plg} of documents suspicious of plagiarism, a set D_{src} of potential source documents, and the task to identify all of the actual plagiarism cases S between the two. In intrinsic plagiarism detection, one is given a set D_{plg} of documents suspicious of plagiarism, and the task to identify all plagiarism cases S in them. This can be accomplished without referring to source documents by checking, based on a writing style analysis, whether all passages of a suspicious document have been written by the same author. If this is not the case, the respective passages are possibly plagiarized. Hence, the source documents used to generate the plagiarism cases are omitted in this part of the corpus. Also, the intrinsic plagiarism cases are not obfuscated to preserve the writing style of the original author; the percentages of unobfuscated plagiarism cases in the corpora include the the cases belonging to the intrinsic part. In the 2010 version of the corpus, the two parts of the corpus were merged in order to make the corpus more realistic: in practice, one cannot expect a clean separation of plagiarism cases whose sources can be found from those whose sources are unavailable. However, since merging the two subtasks turned out to be too difficult for the participants of the competition, the 2011 corpus was split up again.

Obfuscation Besides using the aforementioned obfuscation strategies, unobfuscated as well as translated plagiarism has been generated. The 2009 version contains only artificial plagiarism, the 2010 version for the first time simulated plagiarism, and the 2011 version in addition simulated translations. The artificially translated plagiarism was generated using Google Translate, while the simulated translations were generated by combining machine translation with a manual correction on AMT. This was done in order to foreclose workers on Mechanical Turk to just submit machine translations of texts to be translated.

Topic Match For the 2010 version of the corpus, we attempted to generate plagiarism cases between topically related documents. To this end, the source documents and the suspicious documents were clustered into $k = 30$ clusters using bisecting k-means [241]. Then an equal share of plagiarism cases were generated for pairs of source documents and suspicious documents within as well as between clusters. Presuming clusters correspond to (broad) topics, we thus obtained intra-topic and inter-topic plagiarism.

Plagiarism per Document, Document Length, and Case Length The corpora contain documents with different amounts of plagiarism as well as documents and plagiarism cases of different lengths. These parameters have not changed much, though, since they appear to have little influence on detection performance.

Discussion The corpus contains documents for almost all parameter combinations: short documents with an unobfuscated, short plagiarism case, resulting in a 5% fraction of plagiarism, but also large documents with several obfuscated plagiarism cases of varying lengths, copied from different source documents and resulting in fractions of plagiarism up to 100%. The fraction of plagiarism per

document and the degree of obfuscation per case determine the difficulty of the cases. Since the true distributions of these parameters in real plagiarism are unknown, sensible estimations were made for the corpus. For example, there are more simple plagiarism cases than complex ones, where "simple" refers to short cases, hardly plagiarism per document, and less obfuscation.

4.2.5 Corpus Validation

To validate the "quality" of the plagiarism cases created for our corpus, we report on a large-scale validation study. We compare both artificial plagiarism cases and simulated plagiarism cases to cases of the two corpora Clough09 and METER. Presuming that the authors of these corpora put their best efforts into the construction and annotation of plagiarism cases, the comparison gives insights whether our scale-up strategies are reasonable in terms of case quality. To foreclose the results, we observe that simulated plagiarism and, in particular, artificial plagiarism behave similar to the two handmade corpora. In the light of the employed strategies to construct plagiarism this result may or may not be surprising—however, we argue that it is necessary to run such a comparison in order to provide a broadly accepted evaluation framework in this sensitive area.

The experimental setup is as follows: given a plagiarism case $s = \langle s_{\text{plg}}, d_{\text{plg}}, s_{\text{src}}, d_{\text{src}} \rangle$, the plagiarized passage s_{plg} is compared to the source passage s_{src} using 10 different retrieval models. Each model is an n-gram vector space model (VSM) where n ranges from 1 to 10 words, employing stemming, stop word removal, tf-weighting, and the cosine similarity. Similarity values are computed for a sample of cases from each corpus, the size of which is upperbounded by the largest corpus: 100 similarities are sampled from each corpus.

The rationale of this setup is based on the well-known fact from near-duplicate detection that if two documents share only a few 8-grams—so-called shingles—it is highly probable that they are du-

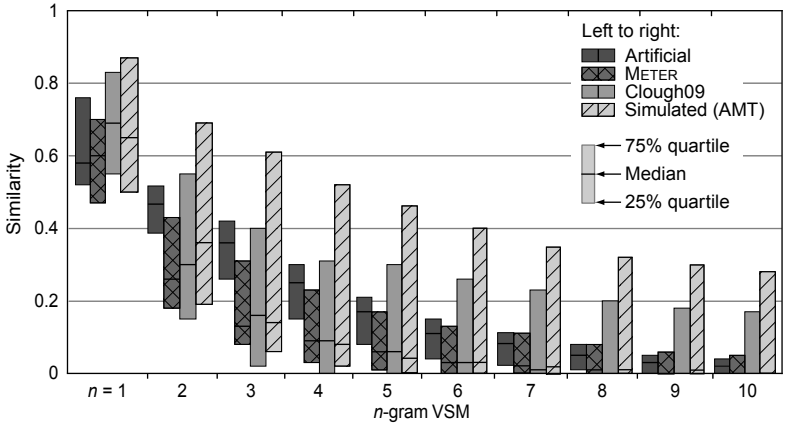


Figure 4.1: Comparison of four text reuse corpora: each box depicts the middle range of 100 similarities obtained from comparing source passages to their rewritten versions using an n-gram VSM ($n \in \{1, \ldots, 10\}$ words).

plicates [32]. Another well-known fact is that two documents which are longer than a few sentences and which are exactly about the same topic will, with a high probability, share a considerable portion of their vocabulary (i.e., they have a high similarity under a 1-gram VSM). It follows for plagiarism detection that a common shingle between s_{plg} and s_{src} pinpoints very accurately an unobfuscated portion of s_{plg}, while it is inevitable that even a highly obfuscated s_{plg} will share a portion of its vocabulary with s_{src}. The same holds true for all other kinds of text reuse.

Figure 4.1 shows the obtained similarities, contrasting each n-gram VSM and each corpus. The box plots show the middle 50% of the respective similarity distributions as well as median similarities. The corpora divide into groups with comparable behavior: in terms of the similarity ranges covered, the artificial plagiarism compares to the METER corpus, except for $n \in \{2, 3\}$, while the simulated

plagiarism from the Clough09 corpus behaves like that from our corpus, but with a different amplitude. In terms of median similarity, METER, Clough09, and our simulated plagiarism behave almost identical, while the artificial plagiarism differs. Also note that our simulated plagiarism as well as the Clough09 corpus contain some cases which are hardly obfuscated.

From this experiment it can be concluded that, under an n-gram model, artificial plagiarism shows a similar behavior compared to simulated plagiarism. This is despite the fact that artificial plagiarism is not human-readable. So if a plagiarism detection algorithm is capable of detecting artificial plagiarism when treating texts as bags of words, it will also be, to some extent, capable of detecting simulated, and possibly real plagiarism. Nevertheless, the similarity range covered by artificial plagiarism is narrower than that of simulated plagiarism, which may indicate that manually rewritten texts are more diverse in practice. This behavior is unsurprising, and it follows that simulated plagiarism should always form part of a plagiarism detection evaluation. The similarity range of the simulated plagiarism obtained from Mechanical Turk is much wider than that of the Clough09 corpus, which is probably due to the fact that many more people have been involved and because of the comparably wider range of topics. There is certainly a lot more that can be done in terms of generating more realistic artificial plagiarism, and in terms of creating simulated plagiarism with low error-rates, which will both require further research and development. But altogether, our strategies to scale-up the construction of plagiarism corpora seem to work well compared to existing corpora.

4.3 Three Evaluation Competitions

Using our performance measures and the plagiarism corpus, we organized the 1st, 2nd, and 3rd international competition on plagiarism detection in conjunction with the PAN workshop series in 2009, 2010, and 2011. Altogether 32 groups from all over the world participated, 9 of whom more than once. In this section, we survey their plagiarism detectors and the best practices employed for external and intrinsic plagiarism detection. The section concludes with a detailed evaluation of the achieved detection performances.

Given the two parts of the PAN plagiarism corpus $(D_{\text{plg}}, D_{\text{src}}, S)$ and (D_{plg}, S) for external and intrinsic plagiarism detection, the competitions divided into corresponding subtasks:

- *External Plagiarism Detection.*
  Given D_{plg} and D_{src}, the task is to identify the plagiarized passages S in D_{plg}, and their source passages in D_{src}.
- *Intrinsic Plagiarism Detection.*
  Given D_{plg}, the task is to identify the plagiarized passages S.

Note that in the 2010 competition, these tasks were merged into one. Moreover, each task of the competitions divided into two phases:

- *Training Phase.*
  Release of training corpora $(D_{\text{plg}}, D_{\text{src}}, S)$ and (D_{plg}, S) to allow for the development of a plagiarism detection system.
- *Testing Phase.*
  Release of test corpora $(D_{\text{plg}}, D_{\text{src}})$ and (D_{src}), omitting the plagiarism annotations S which are supposed to be detected and then submitted as plagiarism detections R.

Participants were allowed to compete in either of the two tasks or both. After the testing phase, the submitted detections were evaluated, and the winner was determined as the participant whose detections R best matched S on the respective test corpora.

4.3.1 External Plagiarism Detection

Most of the participants submitted notebooks describing their plagiarism detectors. Analyzing them, it became apparent that all of them implemented similar processes. We have unified and organized the various approaches in a "reference retrieval process" for external plagiarism detection. The process follows the basic steps of plagiarism detection outlined in the introduction (i.e., candidate retrieval, detailed analysis, and post-processing, as shown in Figure 1.3).

Candidate Retrieval Given a suspicious document d_{plg} from D_{plg} and the source documents D_{src}, in order to simplify the detection of cross-language plagiarism, non-English documents among D_{src} are translated to English using machine translation (services). Then a subset D'_{src} of D_{src} is retrieved that comprises candidates sources for d_{plg}. Basically, this is done by comparing d_{plg} with every document in D_{src} using a fingerprint retrieval model: d_{plg} is represented as a fingerprint $\mathbf{d}_{\text{plg}}$ of hash values from sorted word n-grams extracted from d_{plg}. Note that sorting the n-grams brings them into a canonical form which cancels out obfuscation locally. Beforehand, d_{plg} is normalized by removing stop words, by replacing every word with a particular word from its synonym set (if possible), and by stemming the remainder. Again, this cancels out some obfuscation.

Since all suspicious documents in D_{plg} are to be analyzed against D_{src}, the entire D_{src} is represented as fingerprints $\mathbf{D}_{\text{src}}$, stored in an inverted index. This way, postlists for the hash values of the fingerprint $\mathbf{d}_{\text{plg}}$ of any given d_{plg} from D_{plg} can be retrieved, and all documents from D_{src} referenced in at least k postlists are considered part of D'_{src}. Note that the value of k increases as n decreases. This approach is equivalent to an exhaustive comparison of $\mathbf{d}_{\text{plg}}$ with every fingerprint in $\mathbf{D}_{\text{src}}$ using the Jaccard coefficient, but optimal in terms of runtime efficiency when analyzing all of D_{plg}.

Detailed Analysis A suspicious document d_{plg} is compared in-depth with each candidate source document $d_{\text{src}} \in D'_{\text{src}}$. This is done as follows by means of a heuristic sequence alignment algorithm: first, the sorted word n-grams that match exactly between $\mathbf{d}_{\text{plg}}$ and $\mathbf{d}_{\text{src}}$ are extracted as seeds. Second, the seeds are merged stepwise into aligned passages by applying merge rules. A merge rule decides whether two seeds (or aligned passages) can be merged (e.g., in case they follow each other in both documents). Typically, a number of merge rules are organized in a precedence hierarchy: a superordinate rule is applied until no two seeds can be merged anymore, then the next subordinate rule is applied on the resulting aligned passages, and so on until all rules have been processed. Third, the obtained pairs of aligned passages r_{plg} and r_{src} are returned as plagiarism detections $r \in R$, where $r = \langle r_{\text{plg}}, d_{\text{plg}}, r_{\text{src}}, d_{\text{src}} \rangle$.

Post-Processing Before being returned, the set of plagiarism detections R is filtered in order to reduce false positive detections. In this respect, a set of "semantic" filter rules are applied: for example, detections below a certain length or detections whose passages r_{plg} and r_{src} do not exceed a similarity threshold under some retrieval model may be filtered. Moreover, ambiguous detections that report different sources for approximately the same plagiarized passage in d_{plg} are dealt with (e.g., by discarding the less probable alternative).

Discussion and Comparison of the Three Competitions In the course of the three competitions, the detectors submitted have matured and specialized to the problem domain. With regard to their practical use in a real-world setting, however, some developments must be criticized. In general, many of the approaches work only for plagiarism detection in local document collections, but cannot be applied easily for plagiarism detection against the web. A novelty since 2010 was that many participants approached cross-language plagiarism cases straightforwardly by automatically translating all

non-English documents to English. In cross-language information retrieval, this solution is often mentioned as an alternative to others, but it is hardly ever applied. Reasons for this include the fact that machine translation technologies are difficult to be set up, which is of course alleviated to some extent by online translation services like Google Translate. With regard to plagiarism detection, however, this solution can again only be applied locally, and not on the web.

Most of the retrieval models for candidate retrieval employ "brute force" fingerprinting, say, instead of selecting few n-grams from a document as is custom with near-duplicate detection algorithms like shingling and winnowing [32, 197], all n-grams are used. The average n is about 4.2 words, but ranges from 2 to 6 have been tried. A new development since PAN 2010 was that the n-grams are sorted before computing their hash values. Moreover, some participants put more effort into pre-processing (e.g., by performing synonym normalization). Altogether, these methods can be seen as *counter-obfuscation* heuristics. Fingerprinting cannot be easily applied when retrieving source candidates from the web, so some participants employ standard keyword retrieval technologies, such as Lucene and Terrier. All of them, however, first chunk the source documents and index the chunks rather than the documents, so as to retrieve plagiarized portions of a source document more directly. In any case, the use of inverted indexing to speed up candidate retrieval is predominant; only few participants still resort to a naïve exhaustive comparison of suspicious and source documents.

With regard to detailed analysis, one way or another, all participants employ sequence alignment heuristics, but few notice the connections to bioinformatics and image processing. Hence, due to the lack of a formal framework, participants come up with rather ad hoc rules. Finally, in order to minimize their granularity, some participants discard overlapping detections with ambiguous sources partly or altogether. It may or may not make sense to do so in a competition, but in a real-world setting this cannot hold.

4.3.2 Intrinsic Plagiarism Detection

Intrinsic plagiarism detection has been studied as an individual subtask in the 2009 and 2011 competitions, while in 2010 it was merged with external plagiarism detection. An analysis of the submitted notebooks reveals a generic set of building blocks all of which employ a chunking strategy, a writing style retrieval model, and an outlier detection algorithm; however, the specifics differ significantly. In all cases, the mentioned building blocks are arranged within a retrieval process similar to that described by the authors of [134, 205], the latter of which being the best performing detector of 2009: For a given suspicious document, (1) the document is chunked, (2) the chunks are represented under the style retrieval model, and (3) style differences are identified by means of outlier detection among the chunk representations. (4) After post-processing, the identified chunks are returned as potentially plagiarized passages.

Chunking All of the submitted detectors employ sliding window chunking with chunk sizes ranging from 200 to 1000 words. The slide stepping of the window ranges from 40 to 500 words.

Retrieval Model Retrieval models for intrinsic plagiarism detection are comprised of a model function that maps texts onto feature representations along with a similarity measure to compare representations. The submitted detectors use either character-based or word-based features: for example, char-3-grams as well as other well-known features that quantify writing style [205, 179], the 2500 most frequent char-3-grams [106], the 100 rarest words that appear in at least 5% of all chunks [1], or a standard vector space model [146]. Notice that the choice of features determines the least sensible chunk length, since some features require a minimum amount of text in order to provide robust results. Regarding similarity measures, most detectors employ measures similar to Stamatatos' nd_1 [205].

Outlier Detection Based on the style retrieval model, outlier detection attempts to identify chunks of the suspicious document that are noticeably different from the rest. The following two strategies have been applied: (1) measuring the deviation from the average document style, and (2) chunk clustering. The former strategy follows the original proposal of [134] by comparing each chunk representation with that of the whole suspicious document [146, 179]. Rationale of this approach is to measure the extent to which the style of a chunk matches the average style of the whole suspicious document. A significant deviation is interpreted as an indication of different authorship. Chunk clustering, on the other hand, compares the chunk representations and attempts to cluster them into groups of similar styles, whereas the chunks of each group may have been written by a different author [1, 106]. While a lot of finesse has to go into outlier detection, it is important to keep in mind that these algorithms also depend crucially on the choice of retrieval model.

Post-processing With regard to post-processing most detectors merge overlapping and consecutive chunks that have been identified as outliers in order to decrease detection granularity.

Discussion and Comparison of the Three Competitions Intrinsic plagiarism detection has received less attention than external plagiarism detection. One reason may be that intrinsic detection is still in its infancy compared to external detection, and so is research on combining the two. In the 2010 competition a combined plagiarism detection task was tried, yet the winning participant reports to have dropped developments in favor of external plagiarism detection [104]. The third winner has successfully combined intrinsic and external detection by employing the intrinsic detection algorithm only on suspicious documents for which no external plagiarism has been detected [141]. Hence, it was decided to reinstate intrinsic plagiarism detection in 2011 as a distinct subtask.

In 2009, only one detector achieved a performance above the baseline [205], whereas in 2011, this detector was outperformed significantly by another one [146]. This improvement, however, should be taken with a grain of salt: the detector's retrieval model quantifies the uniqueness of a word with regard to the whole suspicious document. However, during corpus construction, plagiarism cases have been inserted into the suspicious documents from randomly chosen source documents, so that no topic overlap between a suspicious document and its sources can be expected (i.e., with a high probability, words have been inserted into the suspicious documents that did not occur beforehand). A retrieval model which builds word uniqueness features hence benefits from this construction principle. Moreover, it is surprising that such a retrieval model performs that well, since models based on character n-grams have previously been shown to outperform word-based style quantification. Presumably, this performance may not be reproduced in different settings.

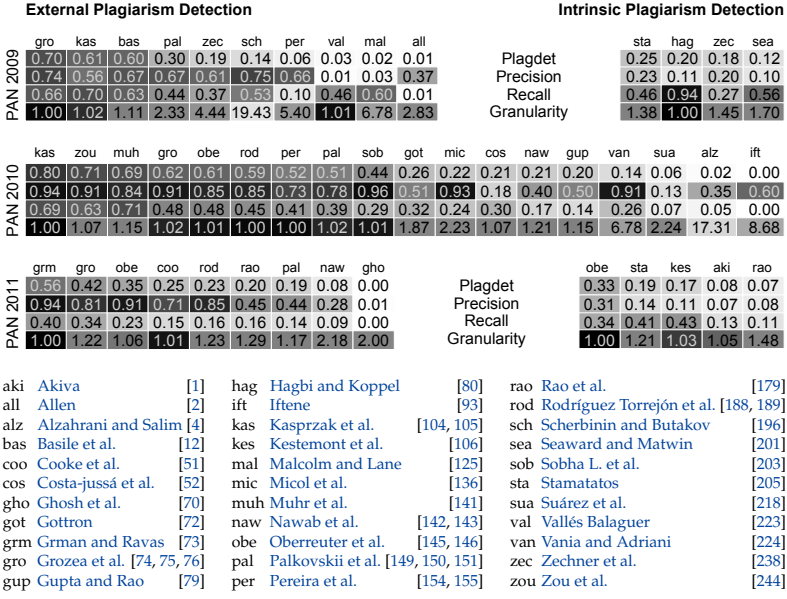
These results are nonetheless important as they pinpoint a problem with constructing a corpus for intrinsic plagiarism detection. Randomly inserting text into a document may preserve writing style, but it obviously does not represent plagiarist behavior, and it hence opens the door to detection approaches which may not be applicable in practice. Though, at the time of writing, no better way of constructing a corpus for intrinsic plagiarism detection evaluation is at hand, this will be an important subject for future research.

4.3.3 Detection Performance Evaluation

We have analyzed the detection results submitted by the participants of each competition using the performance measures precision, recall, granularity, and plagdet as introduced in Section 4.1. The latter served as a means to rank the detectors, while the former three shed light onto specific performance characteristics.

PAN 2009 The first row of Figure 4.2 shows the detection performances of the 13 detectors that took part in PAN 2009. Regarding external plagiarism detection, the approach of Grozea et al. [76] performs best in terms of plagdet and granularity. Yet, the approach with top recall is the one on rank 2, while the one with top precision is on rank 6. Otherwise, only the top three detectors achieved a reasonable performance on all four measures, while the others each perform poorly on at least one of them. Regarding intrinsic plagiarism detection, the approach of Stamatatos [205] performs best. Since intrinsic plagiarism detection is a one-class classification task, its baseline performance is not 0 but the performance obtained when detecting everything as plagiarism. This is almost exactly what the detector of Hagbi and Koppel [80] did. Interestingly, this detector is on rank 2 while the two remaining approaches perform worse.

PAN 2010 The second row of Figure 4.2 shows the detection performances of the 18 detectors that took part in PAN 2010. That year, external and intrinsic plagiarism detection have been treated as an integrated task. The best performing detector is that of Kasprzak and Brandejs [104], which outperforms both the second and the third detector by about 14% in terms of plagdet performance. The remaining detector's performances vary widely from good to poor performance. When looking at precision, the detectors roughly divide into two groups, namely detectors above and below a precision of 0.7. Apparently, almost all detectors with a precision above this threshold achieve top ranks. The recall is, with some exceptions, proportional to the plagdet ranking, while the top 3 detectors are set apart from the rest. Most notably, some detectors achieve a higher recall than their ranking suggests, which pertains particularly to the detector of Muhr et al. [141], which outperforms even the winning detector. With regard to granularity, again, two groups can be distinguished, namely detectors below and above a granularity of 1.5. Remember that a granularity close to 1 is desirable. Again, the



aki Akiva [1]
all Allen [2]
alz Alzahrani and Salim [4]
bas Basile et al. [12]
coo Cooke et al. [51]
cos Costa-jussá et al. [52]
gho Ghosh et al. [70]
got Gottron [72]
grm Grman and Ravas [73]
gro Grozea et al. [74, 75, 76]
gup Gupta and Rao [79]
hag Hagbi and Koppel [80]
ift Iftene [93]
kas Kasprzak et al. [104, 105]
kes Kestemont et al. [106]
mal Malcolm and Lane [125]
mic Micol et al. [136]
muh Muhr et al. [141]
naw Nawab et al. [142, 143]
obe Oberreuter et al. [145, 146]
pal Palkovskii et al. [149, 150, 151]
per Pereira et al. [154, 155]
rao Rao et al. [179]
rod Rodríguez Torrejón et al. [188, 189]
sch Scherbinin and Butakov [196]
sea Seaward and Matwin [201]
sob Sobha L. et al. [203]
sta Stamatatos [205]
sua Suárez et al. [218]
val Vallés Balaguer [223]
van Vania and Adriani [224]
zec Zechner et al. [238]
zou Zou et al. [244]

Figure 4.2: Plagiarism detection performances at PAN 2009-2011. Grid columns show the performances of a plagiarism detector in terms of plagdet, precision, recall, and granularity. The columns are ordered by plagdet scores. Cell shading serves as visualization, where black indicates maximum, and white minimum performance.

detectors below the threshold tend to be ranked high, whereas the detectors on rank two and three have a surprisingly high granularity when compared to the others. Altogether, a lack of precision and/or high granularity explains why some detectors with high recall get ranked low (and vice versa), which shows that there is more than one way to excel in plagiarism detection. Nevertheless, the winning detector does well in all respects.

From the groups that took part in PAN 2009, 5 participated again. Note in this connection that the performances cannot be compared directly across years, since the 2010 test corpus was constructed differently than that of 2009, removing a number of errors. Nevertheless, it can be observed that many improved their detectors' relative performance: the detectors of Kasprzak and Brandejs [104] and Muhr et al. [141] outperformed the 2009 winner, while the one of Pereira et al. [155] achieved mid range performance. The detector of Palkovskii et al. [149] maintained its relative position, and the one of Grozea and Popescu [74] report that they have not changed their detector but instead reused the one that won 2009.

PAN 2011 The third row of Figure 4.2 shows the detection performances of the 12 detectors that took part in PAN 2011. Regarding external plagiarism detection, the best performing detector of Grman and Ravas [73] dominates all others. The second and third detector of Grozea and Popescu [75] and Oberreuter et al. [146] achieve 33% and 60% less plagdet performance. The precision performance of the top five detectors is very high, while recall varies from medium to poor. The granularity of the top five detectors is not always close to 1, leaving room for improvement. Regarding intrinsic plagiarism detection, the detector of Oberreuter et al. [146] performs best overall, while that of Kestemont et al. [106] performs best in terms of recall. Interestingly, both detectors achieve their performances based on different outlier detection paradigms, namely chunk-document comparison and chunk-chunk comparison. With the latter, it appears to be more difficult to achieve a good tradeoff between precision and recall. The 2009 winning detector from Stamatatos [205] serves as a baseline for comparison. It still outperforms all others, except that of Oberreuter et al. which performs more than 40% better.

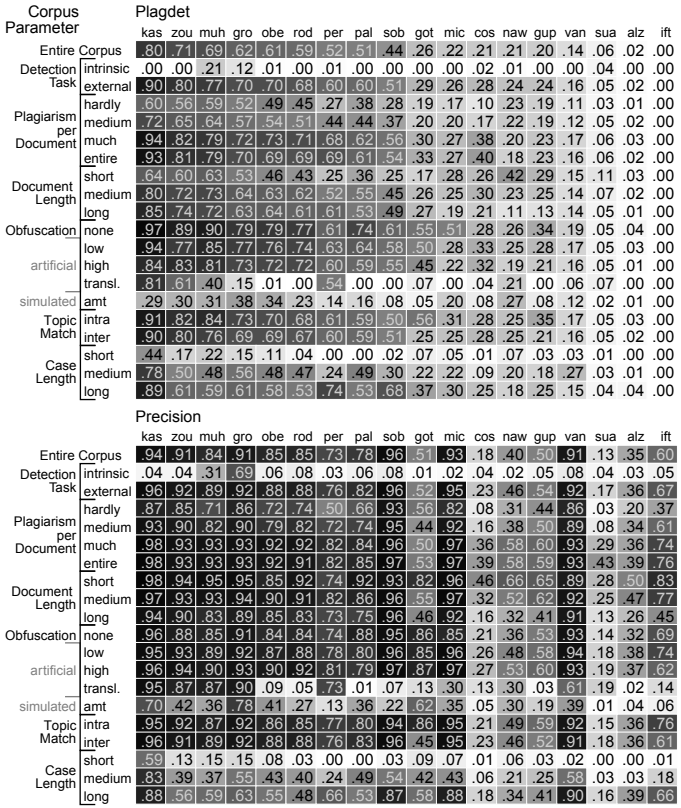
From the groups that took part in the previous competitions, 5 participated again, 2 of them for the third time. Again, note that the performances cannot be compared directly across years, since

the 2011 test corpus was constructed to be much more difficult than those of previous years, containing a lot more obfuscated plagiarism cases at the expense of unobfuscated ones. This is why the absolute recall and plagdet performance values are a lot lower than before.

4.3.4 Detection Performance Evaluation per Corpus Parameter

The test corpora used for the competitions comprise a number of parameters and thus a high diversity of plagiarism cases (see Table 4.4 for an overview). We have analyzed the detectors evaluated at PAN 2010 and 2011 with regard to their detection performance of plagiarism cases for each corpus parameter. The test corpus used for PAN 2009 comprises many of these parameters as well, but a construction weakness rendered such an evaluation impossible. Figures 4.3, 4.4, and 4.5 show the parameter-wise performances of the detectors of PAN 2010 and PAN 2011, respectively.

Obfuscation Perhaps one of the most interesting corpus parameters is obfuscation (i.e., the strategies to modify a plagiarized passage). The respective rows in the aforementioned figures show the detection performances achieved per strategy. Unsurprisingly, many detectors detect unobfuscated plagiarism rather well. In PAN 2010, the top detectors also achieve good performance on artificial plagiarism with low obfuscation, but less so in 2011: here, the recall difference compared to unobfuscated plagiarism is higher, whereas precision and granularity are unaffected. Plagiarism with high obfuscation is detected less well than that with low obfuscation, but due to the increased difficulty of the 2011 test corpus, the performance gap towards unobfuscated plagiarism became a lot wider. The detection performance of translated plagiarism was already quite high in 2010 and improved in 2011. No doubt, the use of machine translation services to unify the languages of the test corpus to English works, but as mentioned earlier, this is not applicable in practice.



alz Alzahrani and Salim [4]
cos Costa-jussá et al. [52]
got Gottron [72]
gro Grozea and Popescu [74]
gup Gupta and Rao [79]
ift Iftene [93]
kas Kasprzak and Brandejs [104]
mic Micol et al. [136]
muh Muhr et al. [141]
naw Nawab et al. [142]
obe Oberreuter et al. [145]
pal Palkovskii et al. [149]
per Pereira et al. [155]
rod Rodríguez Torrejón et al. [188]
sob Sobha L. et al. [203]
sua Suárez et al. [218]
van Vania and Adriani [224]
zou Zou et al. [244]

Figure 4.3: Plagiarism detection performances at PAN 2010. Grid columns show the performances of a plagiarism detector in terms of plagdet and precision, grid rows show performances dependent on corpus parameters. The columns are ordered by plagdet scores. Cell shading serves as visualization, where black indicates maximum and white minimum performance.

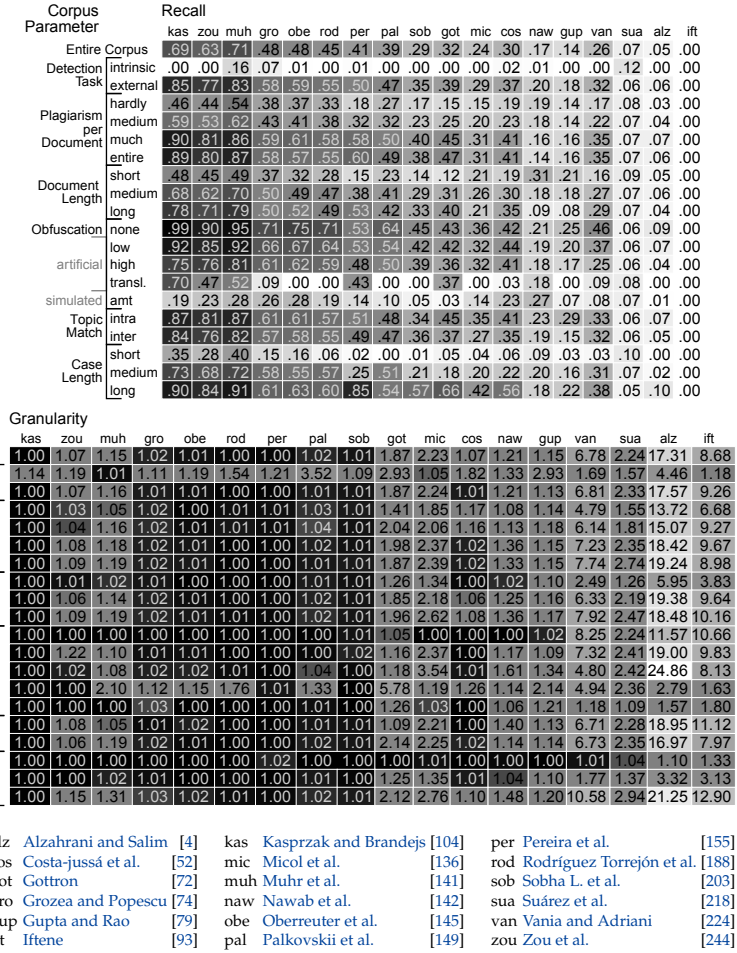
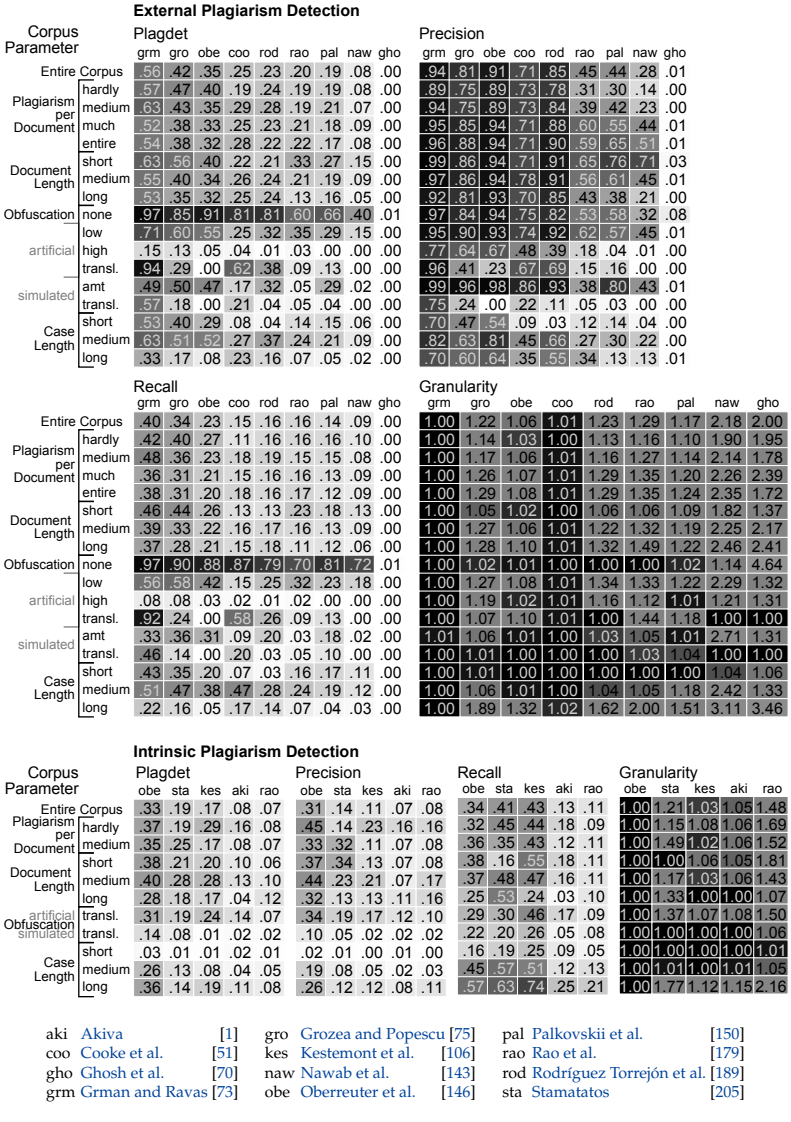


Figure 4.4: Plagiarism detection performances at PAN 2010. Grid columns show the performances of a plagiarism detector in terms of recall and granularity, grid rows show performances dependent on corpus parameters. The columns are ordered by plagdet scores. Cell shading serves as visualization, where black indicates maximum and white minimum performance.



aki Akiva [1]
coo Cooke et al. [51]
gho Ghosh et al. [70]
grm Grman and Ravas [73]
gro Grozea and Popescu [75]
kes Kestemont et al. [106]
naw Nawab et al. [143]
obe Oberreuter et al. [146]
pal Palkovskii et al. [150]
rao Rao et al. [179]
rod Rodríguez Torrejón et al. [189]
sta Stamatatos [205]

Figure 4.5: Plagiarism detection performances at PAN 2011. The columns show the plagdet, precision, recall, and granularity performances of plagiarism detectors, the rows show performances dependent on corpus parameters. The columns are ordered by plagdet scores. The cell shading visualizes the range from maximum (black) to minimum (white) performance.

Simulated plagiarism, however, appears to be much more difficult to detect regarding both precision and recall. Interestingly, the best performing detectors on simulated plagiarism in 2010 are not the overall best performing detectors: Muhr et al. [141], Grozea and Popescu [74], and Oberreuter et al. [145] achieved rank 3, 4, and 5, respectively. Also the detector of Nawab et al. [142] must be mentioned in this connection, which also achieves high recall performance, but suffers from medium precision. It can hence be hypothesized that the approaches implemented by the participants possess different capabilities in detecting the different kinds of obfuscated plagiarism. Conversely, it might also be the case that some participants overfitted their detector to a certain kind of plagiarism, which may have had negative effects on detecting other kinds of plagiarism. In light of this fact, the introduction of simulated plagiarism into the 2010 test corpus helps preventing such developments. Indeed, the overall best performing detectors in 2011 are also the ones which best detect simulated plagiarism. Furthermore, simulated translations have been introduced in 2011, which were detected well only by the winning detector of Grman and Ravas [73], achieving perfect granularity, and good precision at medium recall.

Finally, the only kinds of obfuscation that formed part of the 2011 test corpus for intrinsic plagiarism detection have been artificial and simulated translations. It can be seen that the machine translations have been with a low recall at very low precision, and that simulated translations are detected much worse.

Detection Task and Topic Match Two corpus parameters are unique to the 2010 test corpus: detection task and topic match. Since external and intrinsic plagiarism detection were treated as a single task, the respective plagiarism cases had to be detected at the same time. The two rows entitled "Detection Task" of Figures 4.3 and 4.4 show the detectors' performances at detecting external and intrinsic plagiarism: since most of the participants focused on external pla-

giarism detection, the trends that appear on the entire corpus can be observed here as well. The only difference is that the recall values are between 20% and 30% higher than on the entire corpus, which is due to the fact that about 30% of all plagiarism cases in the corpus are intrinsic plagiarism cases. Only Muhr et al. [141] and Suárez et al. [218] made serious attempts to detect intrinsic plagiarism; their recall is well above 0, yet only Muhr et al.'s precision is reasonable as well. Combining intrinsic and external detection pays off overall for Muhr et al., and the intrinsic-only detection of Suárez et al. even detects some of the external plagiarism cases. Grozea and Popescu [74] tried to exploit knowledge about the corpus construction process to detect intrinsic plagiarism, which is of course impractical and renders their performance negligible.

The two rows entitled "Topic Match" of the figures show the detection performances with regard to whether or not the topic of a plagiarized document matches that of its source documents. It can be observed that this appears to make no difference at all, other than a slightly smaller precision and recall for inter-topic cases compared to intra-topic cases. However, since many participants did not implement a retrieval process similar to that of web search engines, some doubts remain whether these results hold in practice.

Plagiarism per Document, Document Length, and Case Length All of the Figures 4.3, 4.4, and 4.5 also show detection performances with regard to the length of a plagiarism case, the length of a plagiarized document, and the percentage of plagiarism per plagiarized document. Regarding 2010, the general rule was the longer a case and the longer a document, the easier it is to detect plagiarism. This can be explained by the fact that long plagiarism cases in the 2010 test corpus are less obfuscated than short ones, assuming that a plagiarist does not spend much time on long cases, and since long documents contain more of the long cases on average than short ones. Also, the more plagiarism per plagiarized document the better the detection,

since a plagiarism detector may be more confident with its detections if much plagiarism is found in a document. Regarding 2011, the picture changes as a result of our changes of the corpus construction. Long cases have been obfuscated more so that consequently, the correlations between these parameters observed in 2010 were not reproduced. Altogether, however, the plagiarism detectors within each year behave similarly with regard to these corpus parameters.

4.3.5 Conclusions and Future Work

Until recently, the evaluation methodologies in the field of plagiarism detection had conceptual shortcomings, allowing only for limited comparability. Our research over the past three years has contributed right here: we have introduced tailored performance measures for plagiarism detection and the large-scale PAN plagiarism corpus for the controlled evaluation of detection algorithms. The corpus features various kinds of plagiarism, including obfuscated cases generated automatically and manually. The corpus design has been successfully validated in relation to previous corpora.

Until now, 32 plagiarism detectors have been compared using our evaluation framework, many of them repeatedly. This high number of systems has been achieved based on three evaluation workshops in which the framework was employed and developed, namely PAN 2009 [169], PAN 2010 [170], and PAN 2010 [175]. A number of lessons learned can be derived from the evaluation workshops: research and development on external plagiarism detection focuses too much on retrieval from local document collections instead of web retrieval, while the more challenging intrinsic plagiarism detection gets less attention. Besides web retrieval, another challenge in external plagiarism detection is obfuscation: while artificial obfuscation appears to be detectable relatively easy if a plagiarism case is long, short plagiarism cases as well as simulated obfuscation is not. Regarding translated plagiarism, again, automatically generated

cases pose no big challenge in a local document collection, while simulated cross-language plagiarism does. Future competitions will have to address these shortcomings.

The evaluation results show that the detection task and obfuscation strategies are key parameters of the corpus that determine detection difficulty: the most difficult cases to be detected are those without source, and those comprising simulated obfuscation. The difficulty to detect simulated plagiarism in general is of particular interest, as it shows that plagiarists have the opportunity to hide their plagiarism effectively from automatic detection. We hope that our framework will be beneficial as a challenging and yet realistic test bed for researchers in order to pinpoint the room for the development of better plagiarism detection systems.

Part II

Language Reuse

Chapter 5

Web Comments for Multimedia Retrieval

Numerous websites invite visitors to comment on their content. To this end, comment boards are provided at the bottom of webpages where submitted comments are shown in chronological order. Comments are one of the first kinds of user-generated web content, and virtually all types of items are being commented on, be them texts, images, songs, videos, products, ideas or personal profiles. Comment boards have not changed much since their début in web-based guest books. Their very purpose is to collect user feedback, but they provide practical value to visitors as well. Hence, we consider comment boards a form of social software which create a community centered around the commented item. Comment boards serve as a paradigm to exploit the wisdom of the crowds since, ideally, commenters share their opinion, their criticism or extraneous information. Unlike tagging, blogging and "wiki-ing," commenting may not be considered work. In practice, however, comment boards appear less useful to the naked eye: popular webpages get flooded with up to thousands of comments, an amount impossible to be browsed by an individual user. Moreover, many comments are utterly irrelevant, spam or replications, which is why comments are often neglected as a source of useful information. It is all of these observations that formed the starting point of our research.

This chapter presents a comprehensive study of the "commentsphere,"[1] giving an in-depth overview of our respective publications [162, 164, 172, 35, 182]. In Section 5.1, we begin with a survey of existing comment retrieval research: we identify filtering, ranking, and summarization as important comment retrieval tasks, which in turn can be classified as comment-targeting or comment-exploiting. Section 5.2 reports on three experimental case studies of comment-targeting retrieval tasks, and Section 5.3 reports on two case studies of comment-exploiting retrieval. The latter demonstrate language reuse of comments in order to compare web items across media.

Our contributions are the following: a unifying survey of comment-related research, a comment filtering model based on writing style features, a comment ranking model based on novelty detection by similarity reduction, a comment summarization model based on a sentiment word cloud visualization, and a retrieval model for measuring cross-media similarity using comments.

5.1 A Survey of Comment-related Research

This section presents a survey of research related to retrieval tasks in the commentsphere. Based on an analysis of 59 relevant papers, we identify three main retrieval tasks where comments are successfully utilized: filtering, ranking and summarization. Other tasks that have attracted less attention include comment discourse extraction and popularity prediction of web items. We distinguish the mentioned tasks with respect to their retrieval targets, which are either the comments themselves or the commented items. We term the underlying paradigms as comment-targeting and comment-exploiting.

[1] The term "commentsphere" is derived from the term "blogosphere", i.e., the commentsphere is made up of all user-generated comments on web items. The term was coined in a comment on a blog post asking how blogs may be improved, in which one commenter suggested "*permalinks in the commentsphere*" [157]. To the best of our knowledge, the first scientific paper mentioning the term is [138]; a variant is the term "commentosphere" [198].

5.1.1 Comment-targeting Retrieval

Rationale of Comment-targeting Retrieval Let d_q be a webpage or an item (e.g., a text, an image, a video) about a particular topic and let D bet the set of comments on d_q. In terms of regular information retrieval, comment-targeting retrieval is organized as follows:

- *Information Need.* A user's interest in d_q is understood as an information need q that targets comments on d_q and d_q's topic. It is not assumed that d_q covers a topic exhaustively.
- *Query.* Formulating a keyword query that targets comments (in the sense of *"Retrieve all comments that contain more information on d_q's topic."*) is hardly possible. However, a good characterization of d_q's topic—and hence the information need—is d_q itself, which hence should be used as query document.
- *Relevance.* A relevant comment on d_q is a comment that *complements* the information of d_q in some respect. It is an information retrieval challenge to develop retrieval models that capture this kind of relevance.

Why is it unreasonable to formulate keyword queries against comment sets, just as we do against the web? The answer is not straightforward: a well-known principle of information retrieval states that nothing can be retrieved about an interesting topic without having a-priori knowledge about it [18]. Within web search tasks, users formulate queries based on their incomplete knowledge about the topic for which they seek documents. The same holds true for retrieving comments, but at a much finer granularity. Since comments are short, the amount of knowledge in a comment is limited to a few facts. For example, to retrieve a comment, the user requires at least partial knowledge of the facts within. There is a (fuzzy) bound for the amount of a-priori knowledge about a fact x to be exceeded before users will start to search for x, and comments

may be considered well below this bound. Another practical aspect prevents users from searching comments manually: why bother to search for a fact in a small, arbitrary set of comments when a web search engine is only one click away?

A good description of what users actually want to find in a set of comments D on d_q is some kind of "surprise." This can be complementary information but also jokes. Either way the invariant is that the comments to be retrieved are on the same topic as d_q. Classical retrieval models (e.g., algebraic models like the vector space model, latent semantic indexing, explicit semantic analysis, or probabilistic models like binary independence, the unigram language model, or latent Dirichlet allocation) are not well-suited to measure such connections: when computing the relevance between d_q and some comment d, term overlap or concept overlap is measured in first place, which is obviously inappropriate for our purpose. In its extreme form, the classical models consider a comment as most relevant which duplicates d_q, although it contributes nothing.

Survey Research which targets comments is organized in the Tables 5.1 and 5.2. Special emphasis is placed on the underlying retrieval models. Two-thirds of the surveyed papers address only one specific *comment type*, namely reviews of products or movies. Reviews play an important role in decision-making when buying something online, which renders this comment type particularly interesting. Also, reviews are rated by other users, which means that they can be used for evaluation purposes. Research on reviews is known as opinion mining, and it relies on technologies developed for sentiment analysis. We emphasize the following distinction: while the very purpose of sentiment analysis is the identification of subjectivity and polarity, opinion mining employs sentiment as a single feature among many for the analysis of reviews.

All comment retrieval models rely on a feature-based vector representation, where a feature is a possibly elaborate function that quantifies a certain aspect of a comment. Table 5.1 organizes the variety of the found features according to nine research fields. Table 5.2 overviews the analyzed papers, whereas the comment retrieval model (multi-)column illustrates the feature usage by referring to the nine research fields of Table 5.1: the •-symbol indicates the most important feature group of a model, which is sometimes extended by features from other feature groups (indicated by the ◦-symbol). We also analyzed the related work with respect to the different approaches for relevance quantification:

1. In comment filtering, the quality of a comment is quantified, where "good quality" refers to good writing style, and the absence of vandalism and extreme sentiments. Also, the reputation of a commenter is taken into account, presuming that good commenters will not write low-quality comments.

2. In comment ranking, the relevance of a comment is put on a level commensurate with its helpfulness as perceived by other users. The existing retrieval models try to capture the concept of helpfulness from the human-labeled product reviews in order to predict the degree of helpfulness of a new review. This approach is related to the learning-to-rank paradigm.

3. In comment summarization, the prevalent approach is the extraction of sentences that express an opinion. The relevance of a full comment is not considered, but the importance of sentences in describing the content of a comment or all comments.

It can be observed that the commenters on comment boards begin to discuss or even argue about the commented item d_q. Most comment boards, however, do not support discussion threading or record the reply-to structure. In this regard, Mishne and Glance [138] tried to predict whether a dispute is happening on a comment board.

Table 5.1: Overview of features used within comment retrieval models.

Group	Features
IR *(Information Retrieval)*	– [sum of] word [n-gram] weights (Boolean, tf, $tf \cdot idf$) – character n-grams ($n \in [1,5]$) – comment-article similarity – word entropy – unigram language model
NLP *(Natural Language Processing)*	– number of dependent tokens – part-of-speech tokens (percentage of verbs, nouns, etc.) – number of verb / noun phrases
Style *(Writing Style)*	– word length (avg. number of syllables) – sentence length (avg. number of words) – paragraph length (avg. number of words) – comment length (number of words, sentences, or paragraphs) – highlightings (e.g. bold, italic) – punctuation (questions, exclamations) – reader's grade level or readability measures (e.g., Flesch Kincaid Grade)
Vand *(Vandalism Detection)*	– non-words, or gobbledygook – letter repetition – upper case words or spelling
Sent *(Sentiment Analysis)*	– word polarity (positive, neutral, negative) – sentence polarity (percentage of positive / negative words) – document polarity (percentage of positive / negative sentences) – word subjectivity (percentage of subjective / objective relations) – sentence subjectivity (percentage of subjective / objective words)
Com *(Comment-specific Analysis)*	– time of posting (absolute, relative) – user rating of the commented object – common debate phrases – user-feedbacks on helpfulness, comment quality (absolute, relative) – appearance of product-/movie-properties (in the comment, in sentences) – product information (price, sales rank, rating) – appearance of brand/product names (in the comment [title, body])
User *(User Modeling)*	– serial sharing commenter quotients (quickness, amount) – behaviors (percentages in rating others good or bad) – popularity (percentages of being rated good or bad, other's clicks/replies) – user identification (nickname, join date, role) – user relationships (friends, foes) – user expertise (activities on the same topic) – user success (top-rated contributions)
Other	– blog title string – use of genre core vocabulary (measured as mutual information)

Table 5.2: Overview of comment-targeting IR research.

Comment retrieval model								Evaluation corpus		Classifier	Reference
IR	NLP	Style	Vand	Sent	Com	User	Other	Relevance basis	Type	Source(s)		
Retrieval task: comment filtering												
	○	●	○					quality	generic	Slashdot	Naïve Bayes	**Section 5.2.1**
○					●			quality, reputation	generic	Slashdot	association rule learning	[225]
●						○	○	misuse, spam	generic	Rediff.com	Naïve Bayes, perceptron, logistic regression, SVM	[200, 199]
●								spam	generic	–	KL-divergence threshold	[139]
○		○	○	○	●	○		quality, reputation	reviews	Amazon	logistic regression	[100, 101]
		○			●	○		quality	reviews	Amazon	SVM	[121]
Retrieval task: comment ranking												
●	○							novelty	generic	Slashdot	–	**Section 5.2.2**
○	○	○	○		○	●		user preference	generic	Digg.com	support vector regression	[87, 107]
	●				○	○		helpfulness	reviews	Amazon, CNET	radial basis function	[122]
		●			○	○		helpfulness	reviews	Amazon	multiple regression	[69]
○	○	●			○	○		helpfulness	reviews	Amazon	SVM	[108]
○	○				●			helpfulness	reviews	Amazon	linear regression, SVM	[240]
					●			increasing positivity	reviews	Ebert's Movies, ConsumerReports.org	custom	[114]
					●	○		–	reviews	–	–	[160]
						●		–	reviews	eBay	stochastic simulation	[92]
Retrieval task: comment summarization												
					●			word frequency	generic	YouTube	–	**Section 5.2.3**
	○				●	○		sentence importance	reviews	Rotten Tomatoes, Amazon, CNET, IMDB	–	[16, 90, 120, 242]
					●	○		sentence importance	reviews	CNET, Epinions, PriceGrabber	–	[116, 115]
○					●	○		sentence importance	reviews	Env. Prot. Agency	SVM	[112, 113]
	○				●			sentence importance	reviews	NHK BS debate, ewoman.co.jp	–	[65]
○	●					○		property frequency	reviews	eBay	–	[124]
					●	○		property frequency	reviews	eOpinion.com	–	[232]

They represent its comments D as a feature vector using features from almost all groups mentioned in Table 5.1, train a decision tree classifier based on 500 annotated comment boards and achieve a classification accuracy of 0.88. The authors of [61, 198] go one step further and attempt to extract the discussion hierarchy from D.

Remarks The wide range of features employed in comment retrieval models reveals the different views that can be taken on the data which makes comment retrieval an interdisciplinary research field. Regarding comment filtering, many models employ user-centered features and domain knowledge about reviews. These features can not be applied on other comment types since comments are often posted anonymously and do not necessarily review something.

Most of the comment ranking models use a comment's helpfulness as relevance measure, which appears to be a reasonable choice. It must be noted, though, that helpfulness is hardly ever defined but derived from the ground truth of the evaluation corpora. If, for example, a model is trained on Amazon reviews it is an open question whether a "domain transfer" to other comment types works. Moreover, it stands to reason that the initiating (query) document d_q for a set of comments D introduces a bias in the relevance assessments, which may not be crucial for reviews but for comments in general. The existing models measure the relevance of comments based on a static, predefined information need, since they do not consider d_q.

Comment summarization is done by extracting the "important" sentences from comments. Again, this can only be done reliably for reviews but not for comments in general: unlike reviews, comments tend to be short and messy, rendering sentence extraction and the quantification of their importance difficult. Altogether we observe that, while being an active research field, comment-targeting retrieval currently focuses too much on reviews. I.e., most likely the proposed models are not adequate for the wider commentsphere.

5.1.2 Comment-exploiting Retrieval

When retrieving web items for which a sufficient number of comments are available, the comments can be used to raise the retrieval recall. This was first observed by Mishne and Glance [138], who performed a large-scale analysis on the importance of comments within the blogosphere. They found that comments account for up to 30% of its size and that the use of comments improves the recall of blog search by 5%-15%, indicating that comments are a vital part of the blogosphere. This work is also the first to assess the intrinsic value of comments. In blog retrieval, in order for a blog post to be relevant to a query, it suffices if one of its comments is relevant to the query, which was also the modus operandi of the TREC blog track [148]. Recently, the same idea has been applied to video retrieval, in which comments provide a rich text resource as well—significantly larger than user-supplied tags or video titles [55, 231, 236]. There is no doubt that this idea can be applied to image retrieval, music retrieval or other types of web items as well. In Section 5.3, we show that comments can be used to compare web items across media.

Comments are also exploited for the summarization of web items [58, 88, 89, 152]: given an item d_q and comments D on that item, the task is to generate a summary of d_q. The comments on d_q are evaluated to find the often referred to parts of d_q. These parts are then used for the summary, circumventing the problem of identifying them solely based on d_q.

Similarly, filtering by comment exploitation is a viable monitoring and maintenance technology. Given a web item d_q about which commenters are outraged, which may be detected using the aforementioned dispute classification approach of [138], the web item could be automatically selected for reexamination by a site administrator. Though we have not found research addressing this, it is very likely that such measures are already being taken on sites such as YouTube, for example.

Another important task in this regard is the prediction of the popularity of a web item based on its comments [99, 103, 138, 219, 221, 234]. Here, the comments are treated like time series data, using features, such as the increase of comments per time frame, to predict whether the commented item will become popular.

Remarks Research on comment-exploiting retrieval is more diverse compared to the research targeting comments. This is in the nature of things as the ways in which comments can be exploited for different purposes cannot be enumerated. Approaches to comment-exploiting retrieval cannot be compared across different retrieval tasks. Within each comment-exploiting task, however, the literature is few and far between, which indicates that comments have not yet been adopted as a source of valuable information about the commented item. By highlighting comment-exploiting retrieval as a paradigm, we hope to foster research in this direction.

5.1.3 Evaluation Corpora

The fifth column in Table 5.2 shows places where comments can be found for evaluation purposes. Since most of the research is about reviews, Amazon is used most often as a corpus. However, there are plenty of other websites which may be useful in this respect (e.g. YouTube, Flickr, Last.fm, Digg, Picasa or news paper sites). Although there is no lack of comments in general, comments with human annotations are rare; exceptions include Amazon, Digg and Slashdot. For our studies, we have compiled two evaluation corpora based on comments from Slashdot and YouTube; both are available to other researchers upon request.

Slashdot Corpus Slashdot is a website for publishing and commenting technology-related news. The publishing process is based on a moderation system in which users can submit an article d_q, and

Table 5.3: Characterization of comment categories on Slashdot.

Category	Description of a comment d	Frequency	Avg. score
Negative Categories			
offtopic	d does not discuss d_q's topic	4%	-0.6
flamebait	d is meant to pick a fight	3%	-0.5
troll	d is a prank to waste other people's time and effort in responding	5%	-0.6
redundant	d repeats what has been said before	2%	-0.3
Positive Categories			
insightful	d makes d_q more accessible with new understandings, analogies, or examples	32%	3.1
informative	d adds new information or a new angle	14%	3.1
interesting	d does not fit in any other category	24%	2.9
funny	d is humorous respecting d_q's topic	16%	3.2

Slashdot's editors decide whether or not d_q will be published. For each published article a comment board D is available, many of whose comments are categorized by Slashdot's comment moderators. Eight predefined comment categories are used: four of which are considered "positive" and four "negative" (see Table 5.3 for a short characterization). Based on the categories assigned by different moderators, an integer score is computed for each comment. The accounting of all assessments is mapped onto a range from -1 (negative) to +5 (maximum positive). Unfortunately, the scores do not reflect how many moderators are involved in an assessment.

We have downloaded all Slashdot articles from January 2006 to June 2008, including all comments. In total, 17 948 articles were published during this period, and about 3.8 million comments were posted. Comments are organized as discussion threads, which means that a large fraction of the comments are not direct responses to an article, but responses to other comments. Only a small fraction of all comments has been categorized by moderators. Our experiments are based on the 311 167 categorized, direct responses. Together, the second and third quartile of the articles get between 16

to 41 direct comments, while the second and third quartile of the comment lengths range from 1 to 45 words. With respect to the distribution of the comments on the categories, there seem to be only very few low-quality comments on Slashdot (see Table 5.3). However, one should be careful to consider this result as an accurate picture considering most comments are not categorized and Slashdot policies encourage moderators to categorize positive rather than negative (i.e. moderators may spend time finding good comments instead of wasting time reading bad ones).

YouTube Corpus YouTube is a video sharing website for homemade videos. The comments on videos are typically very short, and quite often thousands of comments per single video can be found. Since only a single video is associated per YouTube page, and since most comments are very short, we assume that most of them are some kind of opinion expression regarding the respective video. Explanations or discussions are less frequently observed than on Slashdot, for example. This makes YouTube comments especially interesting for opinion summarization. We downloaded 9.8 million comments from YouTube which were posted on 64 830 videos that appeared on several YouTube feeds at the end of 2008. Due to limitations of the YouTube API, only up to 1 000 comments per video could be retrieved, and it was not possible to adjust the time frame in which a comment or a video has been posted.

5.2 Filtering, Ranking, and Summarizing Comments

In this section, we present three exploratory studies which relate to the comment-targeting retrieval tasks discussed above: the filtering of low-quality comments on Slashdot (Section 5.2.1), the ranking of comments on Slashdot (Section 5.2.2), and the summarization of mass opinion in YouTube comments (Section 5.2.3).

5.2.1 Case Study: Comment Filtering

Given a set of comments D, the task is to filter all comments of extremely low quality, particularly comments from spammers and vandals. The case study investigates whether comment filtering on Slashdot can be done on the basis of a writing style analysis. This analysis is interesting since existing retrieval models for this task depend primarily on user modeling [225].

Retrieval Model We assess a comment's quality by its readability which, in turn, depends to some extent on its writing style. User-generated content on the web often lacks in this respect since users tend to use common speech, do not revise their writing for grammatical and spelling errors and often neglect punctuation and capitalization. By contrast, many users seem to prefer good writing over bad writing, since comments with better style achieve consistently higher ratings on Slashdot. For this reason, as well as to ensure generalizability, our feature selection comprises features from linguistic stylometry and vandalism detection. The latter is an especially important feature class targeting low-quality and ill-intentioned comments and was proposed for use in the detection of vandalism on Wikipedia [167]. We use the following features (see Table 5.1):

- *NLP.* The frequency of prepositions and interjections indicates common speech.
- *Style1.* The comment length indicates whether a commenter makes an effort. This feature achieves a remarkable performance in discriminating high-quality Wikipedia articles [25].
- *Style2.* Readability formulas, such as the DC Formula [56, 39], the FK Grade [64, 109] and the GF Index [77] indicate the

sophistication of language use:

$$\begin{aligned} \text{DC} &= 0.1579 \cdot \text{PDW} + 0.0496 \cdot \text{ASL} + 3.6365, \\ \text{FK} &= 0.39 \cdot \text{ASL} + 11.8 \cdot \text{ASW} - 15.59, \\ \text{GF} &= (\text{ASL} + 100 \cdot \text{R3SW}) \cdot 0.4, \end{aligned}$$

where
- PDW = ratio of d's words a 4th-grader understands (based on a dictionary),
- ASL = d's average sentence length,
- ASW = d's average number of syllables per word,
- R3SW = ratio of d's words with at least 3 syllables.

- *Vand1.* The compression rate of a comment's text.
- *Vand2.* The deviation of a comment's letter frequency distribution from the expectation. This as well as the first vandalism feature indicate bad writing or non-writing (e.g. when a commenter hits the keyboard randomly).
- *Vand3.* The frequency of vulgar words in a comment.

Experiments Based on a set D of categorized comments from the Slashdot corpus, a dichotomous classifier $c : \mathbf{D} \rightarrow \{0, 1\}$ is trained on the feature representations $\mathbf{D}$ of D. Naïve Bayes is used as the classification technology. We have also experimented with SVM classifiers but despite their otherwise good performance, Naïve Bayes could not be outperformed. The performance is measured as precision and recall with respect to each class $\in \{0, 1\}$, indicating the negative and the positive comment category on Slashdot. The results shown in Table 5.4 (Experiment 1) are obtained from a ten-fold cross-validation. At the bottom of the table, a baseline is given in which all comments are classified as positive.

Two issues render our classification approach particularly difficult: the class imbalance and the short length of the comments. Keeping these problems in mind, the results of the first experiment are promising but not overwhelming: only a small portion of negative

Table 5.4: Filtering performance of the comment quality model.

Feature(s)	Class (Categories)	Precision	Recall	*F*-Measure
Experiment 1				
All	positive	0.86	0.90	0.88
	negative	0.43	0.33	0.37
Experiment 2: Swapping "funny"				
All	positive exclusive funny	0.74	0.88	0.80
	negative inclusive funny	0.74	0.51	0.60
Experiment 3: Dropping "funny"				
All	positive exclusive funny	0.85	0.91	0.88
	negative	0.61	0.45	0.52
Experiment 4: Third class "funny"				
All	positive exclusive funny	0.76	0.87	0.81
	funny	0.41	0.46	0.43
	negative	0.52	0.17	0.26
Experiment 5: Individual features				
NLP	positive	0.82	0.98	0.89
	negative	0.35	0.06	0.10
Style1	positive	0.81	1.00	0.90
	negative	0.00	0.00	0.00
Style2	positive	0.84	0.94	0.89
	negative	0.43	0.19	0.26
Vand1	positive	0.82	1.00	0.90
	negative	0.00	0.00	0.00
Vand2	positive	0.84	0.96	0.90
	negative	0.50	0.18	0.26
Vand3	positive	0.83	0.97	0.90
	negative	0.50	0.12	0.19
Baseline: All comments positive				
–	positive	0.81	1.00	0.90
	negative	0.00	0.00	0.00

comments are classified as such. Another issue which needs to be addressed in this regard is the category "funny." Funny comments are a vital part of Slashdot but presumably neither our features nor any of those surveyed are capable of capturing funniness. One of the few publications targeting humor retrieval is [137], wherein the authors try to distinguish humorous texts from other texts. But the particular case where a humorous text is a response to another not necessarily humorous text has not been studied. The Slashdot corpus appears as a valuable resource for humor retrieval.

We have conducted three additional experiments in which the "funny" category was either swapped from positive to negative, dropped or considered as a third class altogether. The results are also shown in Table 5.4. As is evident, when considering funny comments as negative the classification performance is significantly improved, which may be an indication that funny comments look similar to negative comments. Note that dropping funny comments results in a better performance as well. Considering funny comments as a third class does not work since they cannot be significantly separated from negative comments using our retrieval model. In [182] we investigate this issue further.

Compared to the results of Veloso et al. who study the same classification task, we achieve a similar classification accuracy. Note however, that we employ an entirely different feature set: the retrieval model of Veloso *et al.* is specific to Slashdot since it is based primarily on a user model, whereas our model can be considered as domain-independent, more robust and applicable to anonymous comments. Finally, Experiment 5 measures the classification performance of single features. In contrast to the findings of [25], the comment length feature Style1 does not improve over the baseline, which is also the case for the vandalism feature Vand1.

5.2.2 Case Study: Comment Ranking

Given a document d_q and a set of comments D, the task is to rank the comments according to their novelty with respect to d_q. Ordinary novelty detection identifies sentences in a document stream which complement facts already known to the user [204]. Analogously, d_q encodes a user's a-priori knowledge before exploring D. The case study investigates the applicability of novelty to comment ranking. This analysis is interesting since none of the existing approaches include d_q in their retrieval models, which may be acceptable for review ranking but not for general comment ranking.

Retrieval Model We analyze whether the well-known maximal marginal relevance (MMR) model works for Slashdot comments. Moreover, we propose a new model, called ESA_Δ, and compare it to MRR. Both MMR and ESA_Δ are meta-models, since they build upon a generic retrieval model in order to boost their performance. Here, the generic retrieval model is a *tf*-weighted vector space model, which hence is also a good baseline for comparison.

Under the MMR model, the most relevant comment which complements a given d_q is computed iteratively from the comments D on d_q, based on a subset $S \subset D$ that the user already knows [36]. In the i-th step, the comment d_i at rank i is computed as follows:

$$d_i = \underset{d_x \in D \setminus S}{\text{argmax}}[\lambda \cdot \varphi(\mathbf{d}_x, \mathbf{d}_q) - (1-\lambda) \cdot \max_{d_y \in S}[\varphi(\mathbf{d}_x, \mathbf{d}_y)]],$$

where $S = \{d_q, d_1, \ldots, d_{i-1}\}$ are the a-priori known comments, φ is the cosine similarity, and λ adjusts the trade-off between d_i's similarity to d_q and the novelty of d_i in D. Initially, S contains only d_q. Note that the relevance of d_i to d_q is quantified as the value that maximizes the right-hand side of the equation. In accordance with the literature, we chose $\lambda = 0.8$.

ESA_Δ is based on the explicit semantic analysis paradigm [66, 67, 68]. The original ESA model represents a document d as a vector $\mathbf{d}_{|D_I}$ that comprises the similarities of d to documents from a collection D_I, referred to as index collection. The similarities of d to D_I are computed using the vector space model. Each document from D_I is considered as the description of a particular concept, and documents from Wikipedia have been successfully applied in this respect. The supposed rationale of ESA is to represent d in a concept space that is defined by the index collection. Within ESA, two documents d_q and d are compared by computing the cosine similarity between the concept vectors $\mathbf{d}_{q|D_I}$ and $\mathbf{d}_{|D_I}$.

The ESA model's index collection D_I introduces a level of indirection to the similarity computation. This way, connections between d_q and d may become apparent which are not obvious when looking at vocabulary overlap only. In ESA_Δ we intend to extract exactly this portion of similarity by first reducing the overlap between d_q and d: all terms from d that appear in d_q are removed. What remains in the reduced comment $d_\Delta = d \setminus d_q$ is considered as the comment's "visible novelty." To quantify the relatedness of d_q with regard to d_Δ, the ESA vectors $\mathbf{d}_{q|D_I}$ and $\mathbf{d}_{\Delta|D_I}$ are compared.

Experiments Two experiments were conducted on the Slashdot corpus: a comparison of comment rankings obtained with the two aforementioned models with (1) the reference ranking induced by the Slashdot's comment scores, and (2) the less fine-grained ranking induces by Slashdot's comment categories.

The comment scores on Slashdot define a ranking that can be used to evaluate a retrieval model with respect to its ability to capture relevance. For the comments D of each article d_q, the average relevance value of all comments $D_i \subset D$ with score $i \in \{-1, 0, \ldots, 5\}$ to d_q was computed. The left plot in Figure 5.1 shows the results. The standard deviations for the models are as follows: $\sigma_{\text{VSM}} = 0.13$, $\sigma_{\text{ESA}_\Delta} = 0.16$, and $\sigma_{\text{MMR}} = 0.08$. The vector space model and the

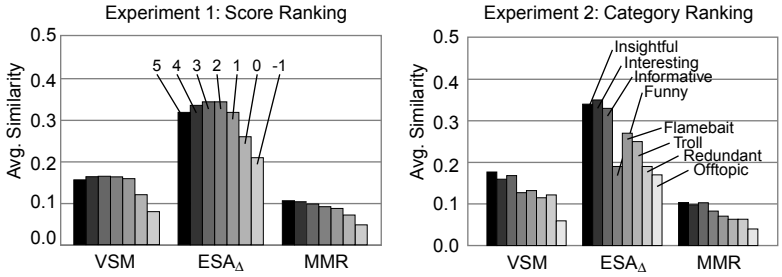


Figure 5.1: Evaluation results for the ranking study: the plots show the average similarity of a comment d to d_q per comment score (left) and per comment category (right). VSM, ESA$_\Delta$, and MMR denote the vector space model, the new similarity-reduced explicit semantic analysis, and maximal marginal relevance.

ESA model show a comparable similarity distribution in which medium scores are ranked highest on average, while MMR places the comments in a natural order. However, the ESA$_\Delta$ model achieves significantly higher similarity values than the vector space model, while the relevance values computed with MMR appear to be rather small. The latter is not a problem as long as the desired ordering of the comments is achieved. To determine whether this is indeed the case, we have also computed the graded relevance scores NDCG, ERR, and Kendall's τ, both on the entire rankings produced by the three models and restricted to the respective top 10 comments only. In addition, these scores have also been computed for a random ranking as a second baseline. Table 5.5 shows the results. As can be seen for the complete rankings, neither of the models clearly outperforms the other, and what is more, neither of the models outperforms the random baseline. On the top 10 comments, at least the latter is achieved, while the three models remain almost indistinguishable.

Table 5.5: Ranking performance of the comment retrieval models. The values in brackets denote standard deviation.

Measure	**Retrieval Models**			**Baseline**
	VSM	ESA_Δ	MMR	Random
NDCG	0.71 (0.08)	0.71 (0.08)	0.70 (0.08)	0.70 (0.08)
ERR	0.54 (0.24)	0.54 (0.24)	0.55 (0.26)	0.53 (0.25)
Kendall's τ	0.09 (0.13)	0.08 (0.12)	0.06 (0.13)	0.01 (0.11)
NDCG@10	0.72 (0.13)	0.73 (0.13)	0.73 (0.13)	0.70 (0.12)
ERR@10	0.58 (0.24)	0.58 (0.24)	0.58 (0.26)	0.55 (0.25)
Kendall's τ@10	0.09 (0.12)	0.08 (0.12)	0.06 (0.13)	0.01 (0.11)

Following the design of the former experiment, we also compute the average relevance value of a comment to d_q per category. Hence, based only on positive and negative judgements, a less fine-grained ranking is demonstrated (e.g. if a user wants negative comments to be presented after the positive comment). The right plot in Figure 5.1 shows the results. Observe the difference in the distributions of the vector space model and MMR to the ESA_Δ model: the latter achieves significantly higher similarities for the positive categories (except for "funny") than for the negative categories, whereas the vector space model and MMR show almost no discriminative behavior.

This above results should be taken with a grain of salt: sensible rankings are induced only in terms of averages while the models' graded relevance scores are inconclusive. Undesired rankings occur with a non-negligible probability. We conclude that the task of ranking comments remains unsolved and needs further research.

5.2.3 Case Study: Comment Summarization

Given a set of comments D, the task is to generate a short text or a visualization that overviews the contents or the opinions of D: users will quickly get an idea about the comments without having to read everything. The case study investigates the use of word clouds for

opinion summarization of comments on YouTube. This analysis is interesting since existing comment summarization approaches rely on sentence extraction and are prevalently applied to distill customer product reviews. The respective technology cannot directly be applied to comments, which are significantly shorter than reviews—most of the comments found on media sharing sites do not even contain one sentence. It is unlikely that relevant information can be found in such comments except the opinion of the commenters. Short comments in particular are tedious to read which is why a suitable summarization for them is desired. While a single opinion may not be very useful (especially if no argument is provided), the fact that popular items inspire thousands of people to share their opinions allows us to generate a representative opinion summary.

Retrieval Model The summarization of a comment set D divides into an offline step and an online step. Suppose that two dictionaries V^+ and V^- are given, comprising human-annotated terms that are commonly used to express positive and negative opinions respectively [217]. In the offline step, the well-known sentiment analysis approach described by [222] is used to extend V^+ and V^- to the application domain. The extension is necessary in order to learn terms that are not covered by the dictionaries, and it is not feasible to do this manually. The semantic orientation SO of an unknown word w is measured by the degree of its association with known words from V^+ and V^-:

$$\mathrm{SO}(w) = \sum_{w^+ \in V^+} \mathrm{assoc}(w, w^+) - \sum_{w^- \in V^-} \mathrm{assoc}(w, w^-),$$

where $\mathrm{assoc}(w, w')$ maps two words to a real number indicating their association strength. If $\mathrm{SO}(w)$ is greater than a threshold ε (less than $-\varepsilon$), w is added to V^+ (V^-); otherwise w is considered neutral. The point-wise mutual information statistic is applied as an

Figure 5.2: Opinion summaries which contrast positive and negative words (left) and all kinds of words (right).

association measure:

$$\text{assoc}(w, w') = \text{PMI}(w, w') = \log_2 \frac{p(w \wedge w')}{p(w) \cdot p(w')},$$

where $p(w \wedge w')$ denotes the probability of observing w together with w', and $p(w)$ is the probability of observing w. In the online step, when a comment set D is observed, two summary term vectors $\mathbf{s}_D^+$ and $\mathbf{s}_D^-$ are constructed each counting the absolute frequencies of positive and negative terms in D, using the dictionaries V^+ and V^-. Words not occurring in one of them are considered neutral.

Visualization We visualize $\mathbf{s}_D^+$ and $\mathbf{s}_D^-$ as word clouds. Figure 5.2 shows examples where $\mathbf{s}_D^+$ and $\mathbf{s}_D^-$ are contrasted. Word clouds are arrangements of words in 2 or 3 dimensions in which important words are highlighted [202]. Here, the words are colored according to their sentiment polarity, and they are scaled according to their term frequency. The font size of a word is computed as follows:

$$\text{size}(w) = \text{max\_size} \cdot \frac{tf(w)}{\max_{w^* \in V} (tf(w^*))},$$

where max_size is a predefined maximum font size and $tf(w)$ is the term frequency of w as counted in $\mathbf{s}_D^+$ or $\mathbf{s}_D^-$, and $V = V^+ \cup V^-$.

A few more issues need to be addressed in practice: emoticons and exclamations (such as ":-)" or "lol") require an additional set of detection rules since commenters often vary their spelling, spelling errors (which are abundant in comments) need to be corrected on the fly, and negations in front of opinion words need to be detected as a heuristic to determine the orientation of a word in context.

An important part of the visualization is the percentages of positive and negative words found on top of the word cloud (see Figure 5.2). For a quick overview these numbers are sufficient; however, when a user wants to know more about what other commenters thought, a click on any word in the cloud produces a list of comments containing it. As mentioned above, we have implemented a browser add-on that summarizes YouTube comments and Flickr comments on-the-fly. A lot of user feedback was obtained this way; from which it became clear that users find the summary interesting and useful, yet they criticize that sometimes words from comments are considered negative (positive) although they have been used in a positive (negative) way. For future developments, it is planned to incorporate sentiment classification of whole comments.

Finally, it is noteworthy that the word cloud shown right in Figure 5.2 inspired us to investigate cross-media retrieval (see the next section). In this particular case, the top 3 neutral words perfectly explain what the video is all about: a cat playing the piano.

5.3 Measuring Cross-Media Item Similarity

Our final two case studies investigate whether a set of comments D can be used for retrieval purposes (i.e., whether the combined text of D tells us something about the commented item d_q), thus allowing for the comparison of items across media. To the best of our knowledge, we are the first to analyze this possibility. Cross-media retrieval is a sub-problem of multimedia retrieval, which again divides into various sub-tasks. Here, we consider the following: given

a set of items of different media types, the task is to pair those items which are similar with respect to their topic, regardless of their media type. A primary goal of cross-media retrieval is the construction of retrieval models that bridge the gap between different media types by means of identifying correlations between low-level features and semantic annotations. We approach this problem from a different perspective through the use of comments in lieu of the commented item. This way, model construction is not an issue since well-known text retrieval models can be directly applied. Although the text surrounding a non-textual item has always been used to extract annotations in multimedia information retrieval [60, 98, 118], comments in particular have not been considered in this respect.

5.3.1 Case Study: Comment Descriptiveness

A premise of our approach is that comments describe the commented item to some extent, which is analyzed first off. Under the assumption that the activity of commenting on text is not fundamentally different from that of commenting on non-textual items, we restrict our experiments to the text domain for now. If the assumption holds and if comments on texts prove to be descriptive, it follows that comments on non-textual items are descriptive as well.

Retrieval Models Two basic retrieval models are employed in our experiments: the well-known vector space model (VSM) and the explicit semantic analysis model (ESA). In short, ESA is a collection-relative generalized VSM [208, 68]. It represents a document d as vector $\mathbf{d}_{|D_I}$ of d's similarities to the documents of an index collection D_I. Our index collection contains $|D_I| = 10\,000$ randomly selected Wikipedia documents, and the similarity of d to every index document is computed in turn using the VSM. Both models use the cosine similarity to compute the similarity of two document representations. Note that we choose *basic* retrieval models to ascertain whether comment descriptiveness can be measured reliably.

Experiments We have conducted three experiments on the Slashdot evaluation corpus. In Experiment 1, we determine the descriptiveness of a set of comments: a given document d_q is compared with the combined text of its comments D. As a baseline, d_q is compared once with the comments of another, randomly selected document. The obtained similarity values are depicted in Figure 5.3 as similarity distributions (i.e., the ratio of all similarities per similarity interval of 0.1 resolution). In Experiment 2, we determine if the combined text of a set of comments D can replace the commented item d_q in a ranking task, by ranking the remaining corpus documents twice: (1) wrt. their similarity to d_q, and (2) wrt. their similarity to D as a whole. The top 100 ranks of these two rankings are compared using the rank correlation coefficient Spearman's ρ, which measures their (dis-)agreement as a value from $[-1, 1]$. The experiment has been repeated with randomly selected documents d_q from the corpus until the averaged correlation value converged (cf. Figure 5.3). In Experiment 3, we determine whether or not the observed similarities between d_q and D as a whole depend only on text which has been copied from d_q into one of D's comments: we apply our aforementioned similarity reduction technique by first removing all terms from D which also occur in d_q, and then by exploiting the fact that ESA, unlike the VSM, has the capability to measure more than just the overlap similarity between d_q and D. Figure 5.3 shows the obtained similarity distributions. It is our goal to determine the amount of comments on a document d_q necessary to reach a certain degree of descriptiveness. Hence, we use 5 subsets of the evaluation corpus which comprise only documents which got at least $|D| \geq i \in \{1, 10, 100, 500, 1000\}$ comments. The experiments were repeated for each subset (= table rows), and, each experiment was repeated for every d_q in a given subset. If a d_q got $|D| > i$ comments a random subset $D_i \subset D$, $|D_i| = i$, was chosen for the respective experiment.

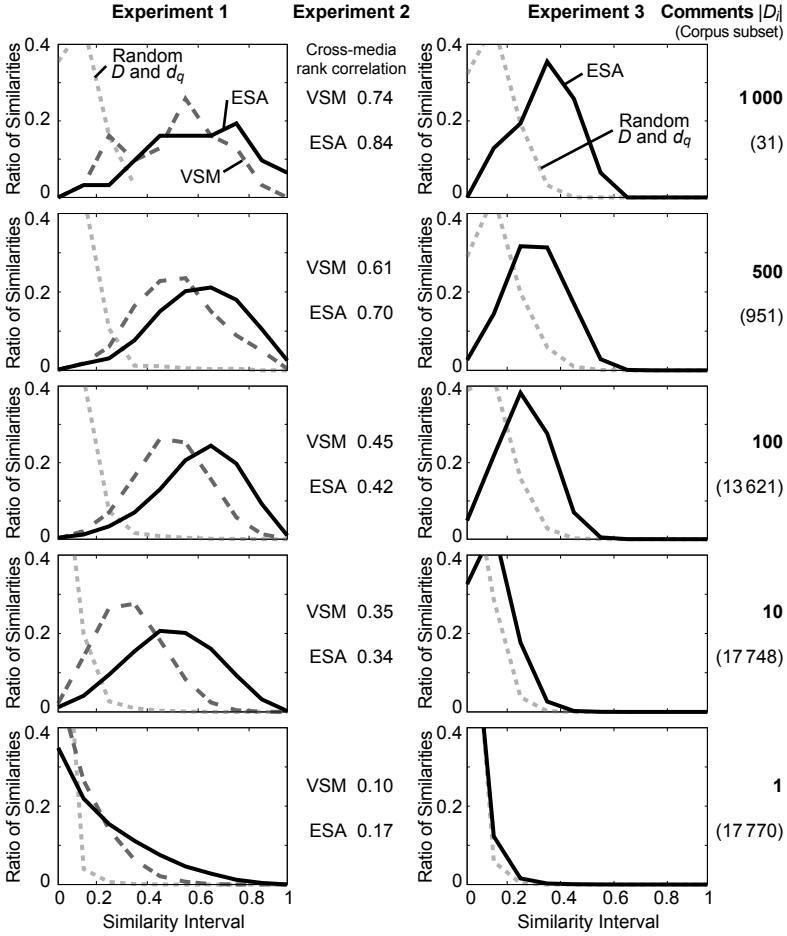


Figure 5.3: Comment descriptiveness experiments dependent on the number of comments.

Experiment 1 reveals that 10 comments on a document are sufficient to reach a considerable similarity between them compared to the baseline, Experiment 2 reveals that 100 to 500 comments are sufficient to reach a moderate rank correlation, and Experiment 3 reveals that 100-500 comments comprise a measurable commenter contribution not contained in the original document. Experiment 3 also demonstrates ESA's capability to measure more than just overlap similarity and shows that the similarities measured in Experiment 1 cannot be attributed solely to duplicated text. Further, Experiment 3 may be interpreted as an indicator of the amount of comments necessary to capture the topics of non-textual objects. In all experiments, the retrieval quality increases with the number of comments per document $|D_i|$. Hence, comments on text documents can be called descriptive and it remains to be investigated whether our hypothesis holds that commenting is not entirely media-dependent.

5.3.2 Case Study: Cross-media Retrieval

Having established that a sufficiently high number of comments on a text describe it well, we proceed in this case study toward reusing comments for the comparison of web items of different media types.

Retrieval Model A standard vector space model with $tf \cdot idf$ term weighting is used as our retrieval model. Given a web item d_q and its associated set of comments D, d_q is represented as a term vector $\mathbf{d}_q$ based on the index terms found in D, while applying stop word reduction and stemming. In the case that d_q is a text document, as in the Slashdot corpus, the index terms found in d_q are also included in $\mathbf{d}_q$. The representations of two items, $\mathbf{d}_q$ and $\mathbf{d}'_q$, are compared using the cosine similarity. Though nearly every retrieval model can be employed for this task, we resort to a simple vector space model in order to determine how robust a cross-media similarity assessment can be accomplished.

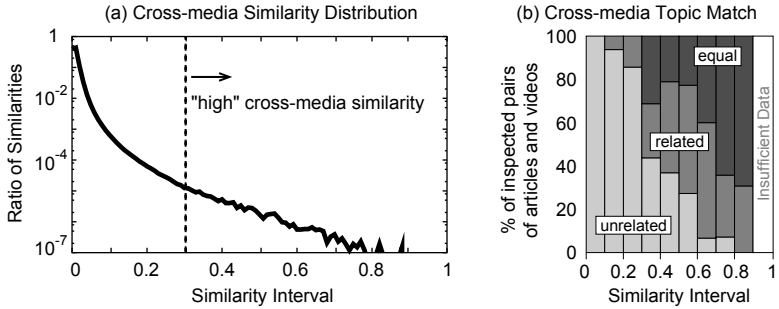


Figure 5.4: (a) Distribution of cross-media similarities between YouTube videos and Slashdot articles per similarity interval of resolution 0.01. (b) Percentage of pairs of articles and videos with unrelated topics, related topics, and equals topics per similarity interval of resolution 0.1, based on a stratified sample of 150 manually inspected item pairs.

Experiments Given the evaluation corpora described above, 6 000 videos from the YouTube corpus were sampled and compared to each of the 17 948 Slashdot articles. This resulted in about 107.7 million similarities. Slashdot and YouTube are similar in that both are community-driven websites, so that at least some topical overlaps can be expected. However, since both corpora have been compiled independently, we were not aware of existing overlaps. Figure 5.4a shows the obtained similarity distribution as a percentage of similarities over similarity intervals with an interval resolution of 0.01.

From all pairs of articles and videos compared, we have sampled a total of 150 for manual inspection by means of stratified sampling from similarity intervals of resolution 0.1. The sampled pairs were then classified into categories of topical match, namely pairs with equal topic, related topics, and unrelated topics. Figure 5.4b shows the obtained results. Topic overlaps start appearing at similarities above 0.3, which may already be considered a "high" cross-media similarity for its considerable positive deviation from the expecta-

Table 5.6: Overview of the top 100 inspected cross-media similarities.

Topic Match	Share	Similarity				Avg. # Comments		Title Match
		min	avg.	max	stdev	Slashdot	YouTube	
equal	36%	0.71	0.78	0.91	0.06	53	927	72%
related	55%	0.71	0.76	0.91	0.04	81	683	62%
unrelated	9%	0.72	0.78	0.87	0.05	104	872	–
Σ	100%	0.71	0.77	0.91	0.05	74	790	60%

Table 5.7: Selection of matching web items found with comment-based cross-media retrieval.

Similarity	Comments		URLs
	Slashdot	YouTube	
0.91	83	950	http://slashdot.org/story/07/03/15/2056210 http://www.youtube.com/watch?v=RuWVMB7OxbM
0.82	69	950	http://slashdot.org/story/08/02/05/1511225 http://www.youtube.com/watch?v=Z_gKOCb4QBA
0.81	102	950	http://slashdot.org/story/08/01/02/1611240 http://www.youtube.com/watch?v=tLlHibrFATg
0.76	41	950	http://slashdot.org/story/07/10/16/1526257 http://www.youtube.com/watch?v=TluRVBhmf8w
0.74	40	950	http://slashdot.org/story/07/07/11/1246250 http://www.youtube.com/watch?v=DLxq90xmYUs
0.74	79	766	http://slashdot.org/story/07/08/13/1347253 http://www.youtube.com/watch?v=BWQ5ZMnz25I
0.74	66	78	http://slashdot.org/story/06/02/02/0024235 http://www.youtube.com/watch?v=F0uq21xjMCw
0.73	75	950	http://slashdot.org/story/08/06/04/1159207 http://www.youtube.com/watch?v=adc3MSS5Ydc

tion. As the comment-based cross-media similarity increases, more and more items with related or equal topics can be observed. Within high similarity ranges, pairs of articles and videos with equal topic appear most often. In addition to this manual inspection, the top 100 most similar pairs were evaluated with respect to their topical match. Table 5.6 gives a detailed overview of these item pairs, and Table 5.7 shows a selection of matching item pairs: 91% of the top item pairs have equal or related topics. The similarity values in the table give an idea about the measured similarities and their standard deviation (stdev). Few false positives achieve high similarities, but are based on a lot more comments on the side of Slashdot. Observe that the number of comments appears to correlate with the similarity, and that more comments possibly result in a "topic drift." Often, the title of a YouTube video is descriptive, and hence the percentage of pairs where the video title overlaps with the Slashdot article was determined. This is the case in 60% of the examined item pairs, which in turn means that, in this experiment, 40% of the top 100 item pairs would not have been identified based on their titles.

5.3.3 Conclusion and Future Work

In this chapter, we study the commentsphere from an information retrieval perspective. We present a comprehensive survey of related work and, based on this survey, identify the most relevant retrieval tasks with respect to research effort and user impact: the filtering, the ranking and the summarization of comments as well as their exploitation for the same tasks on web items. In addition, there are a number of secondary retrieval tasks that are no less exciting, including the prediction of a web item's popularity based on comments. We conducted a case study for four of the tasks mentioned above. For this purpose, we compiled adequate corpora which are available to other researchers in order to foster the research activities in this field. Within the case studies, special attention was paid to

the used retrieval models: our objective was to point out differences to existing retrieval models and to provide a better understanding of the challenges for retrieval tasks in the commentsphere. Moreover, we developed new retrieval model variants to address some of these challenges. Our contributions from a retrieval model perspective:

- *Feature Overview.* We compile an overview of features used to represent comments.
- *Retrieval Models for Filtering and Ranking.* We propose a domain-independent model to filter low-quality comments which competes with other models. With ESA_Δ we present a new retrieval model to measure novelty for comment ranking.
- *Cross-media Retrieval.* We introduce a new paradigm to measure cross-media item similarity.

From a research perspective, we consider the following directions as promising:

- *Retrieval Models.* Current research focuses on product and movie reviews, but web comments in general are much more diverse and require new tailored retrieval models.
- *Humor Retrieval.* If surprises are what people expect from comments, they may be funny at the same time: humor retrieval has not recognized comments as a research subject.
- *Multimedia Annotation.* If comments capture a commented item's topic well, it should be possible to extract tags and annotations for the commented item from its comments.
- *Ranking and Summarization.* Comment boards show comments in chronological order, but with increasing comment number the overview gets lost. Hence, ranking comments by their relevance and novelty, as well as summarizing comments will continue to be an important direction for research.

Chapter 6

Web N-Grams for Keyword Retrieval

The interfaces of today's web search engines are mainly keyword-based: users submit queries by typing several keywords into a search box. It is not uncommon that queries comprise phrases and compound concepts; take `times square` as an example. A search engine that is informed about such phrases and concepts by means of proper quotation may consider them as indivisible units and use them to improve retrieval precision (e.g., by excluding documents that do not contain words in the exact same order of the phrases). Other server-side algorithms that benefit from quotation include query reformulation, which could be done on the level of phrases instead of keywords, and query disambiguation, which has to cope with tricky queries like `times square dance`. Without quotes, it is difficult to know whether the user intends to find newspaper articles on `square dance` in the London `times` or rather `dance` events at `times square`, New York (locating the user might also help, of course). Skilled web searchers surround phrases with quotes, but experience shows that most searchers are not even aware of this option. Hence, search engines apply pre-retrieval algorithms that automatically divide queries into segments in order to second-guess the user's intended phrases and to improve the overall user satisfaction.

This chapter studies query segmentation algorithms, giving an in-depth overview of our respective publications [81, 82]. Section 6.1 surveys existing approaches to query segmentation. Section 6.2 presents the basic notation of our query segmentation framework and introduces two query segmentation algorithms. An empirical evaluation in Section 6.3 shows the segmentation accuracy of our method on the current gold standard, compared to existing approaches. Furthermore, this section introduces our new corpus, outlines its construction with the aid of crowdsourcing, and compares the segmentations obtained this way to expert segmentations.

Our contributions are the following: a new and robust approach to the task of query segmentation which relies on reusing the web in the form of *n*-grams and their frequencies. The performance our approach achieves is competitive with state-of-the-art algorithms on a widely used evaluation corpus of 500 queries. As part of our evaluation, we compare our algorithm with seven others proposed in the literature. Our second contribution relates to evaluation and verifiability: a new query segmentation corpus comprising 50 000 queries. Our corpus subsumes the current standard corpus and, unlike that, meets the requirements of representative large-scale evaluations, which was one of the main points raised at the SIGIR 2010 workshop on "Query Representation and Understanding" [54].

6.1 A Survey of Query Segmentation

Recent research suggests a variety of approaches to query segmentation. For instance, Guo et al. [78], Yu and Shi [237], and Kiseleva et al. [110] use methods based on conditional random fields (CRF). Guo et al. evaluate their method on a proprietary query corpus and tackle the broader problem of query refinement that simultaneously involves spelling correction, stemming, etc. Hence, their approach is not entirely comparable to ours, since we assume that spelling correction is done prior to query segmentation—an assumption shared

by most other query segmentation studies. The other CRF-based methods by Yu and Shi and Kiseleva et al. also address query segmentation in settings different from ours. Yu and Shi focus on query segmentation in the context of text stored in relational databases and use database-specific features not available in web search. Kiseleva et al. focus on product queries and aim to improve the user experience in web shops.

One of the earliest approaches to *web* query segmentation is by Risvik et al. [185]. They segment queries by computing so-called connexity scores that measure mutual information within a segment and the segment's frequency in a query log. Jones et al. [102] also use a mutual information-based scoring that finds segments in which adjacent terms have high mutual information. However, neither Risvik et al. nor Jones et al. evaluate the segmentation accuracy of their approaches. In a very recent paper, Huang et al. [91] also use segment-based pointwise mutual information scores obtained from web-scale language models. For a given query, they derive a tree of concepts. The tree is then used to obtain a final query segmentation. However, Huang et al. evaluate their method only on a proprietary query corpus without comparing it to other approaches. Note that in many query segmentation studies, mutual information-based segmentation is used as a baseline, often performing worse than the more involved methods.

One of the earliest methods that does not rely only on mutual information is the supervised learning approach by Bergsma and Wang [23]. Bergsma and Wang incorporate many features: statistical ones like phrase frequencies on the web and in query logs, as well as dependency features that focus on noun phrases. They also established the first gold standard corpus of 500 queries, each segmented by three human annotators. Subsequent work has adopted this gold standard [29, 81, 220, 239]; as do we in our evaluations in order to ensure comparability. However, Bergsma and Wang's evaluation corpus is rather small, so that we decided to introduce a

larger and more representative corpus. As for the query segmentation, Bergsma and Wang's supervised learning method is trained on queries segmented by a single annotator who also segmented the gold standard. This leaves some doubts with regard to the generalizability of the results. Nevertheless, Bendersky et al. [21] successfully use a version of Bergsma and Wang's method as a sub-procedure in their two-stage query segmentation approach.

Instead of the supervised approach that requires training data, Tan and Peng [220] and Zhang et al. [239] suggest unsupervised methods. Zhang et al. compute segment scores from the eigenvalues of a correlation matrix corresponding to a given query. Tan and Peng's method, like ours, uses only n-gram frequencies from a large web corpus as well as Wikipedia. However, Tan and Peng state that raw n-gram frequencies by themselves cannot be used for query segmentation. Hence, they build a language model from the n-gram frequencies via expectation maximization. In a second step, Tan and Peng boost a segment's score derived from the language model if it is used prominently in Wikipedia. Our new method uses the same features as Tan and Peng's but with superior segmentation accuracy (cf. Section 6.3). Moreover, in contrast to Tan and Peng's assumption, our naïve query segmentation method [81] shows how raw n-gram frequencies can be exploited for query segmentation using an appropriate normalization scheme (cf. Section 6.2).

The snippet-based method by Brenes et al. [29] is quite simple compared to the aforementioned approaches: it segments a query based on search result snippets for the unquoted query. Brenes et al. evaluate different techniques of deriving a query's segmentation from snippets. Obviously, the main concern with this approach is runtime efficiency. Most queries have to pass the retrieval pipeline twice before results are returned: once unquoted to obtain the snippets used for segmentation, and once more with quoted segments.

Bendersky et al. [20] also suggest a method that involves time consuming double-retrieval for most queries. Their method intro-

duces a segment break between two words whenever a likelihood ratio is below some threshold. The likelihood ratios are obtained by combining web *n*-gram probabilities and pseudo-relevance feedback (PRF) from the top-ranked documents for the original, unquoted query. The PRF-based method achieves a promising experimental segmentation accuracy. However, Bendersky et al.'s evaluation corpus is rather small (250 queries) and it has been segmented by only one annotator, which suggests a bias in the segmentations. Furthermore, the PRF-based segmentation method is not compared with state-of-the-art approaches.

Our first naïve query segmentation method [81] scores all segmentations for a given query by the weighted sum of the frequencies of contained *n*-grams, obtained from a large web corpus. Besides raw *n*-gram frequencies, no other features are involved, making this approach easy to explain and straightforward to implement. The weighting scheme of naïve query segmentation aims at "normalizing" the *n*-gram frequencies, so that a longer segment like "`toronto blue jays`" has a chance to achieve a higher score than shorter sub-segments like "`blue jays`". With respect to segmentation accuracy, the naïve approach performs comparable to the other approaches that use, supposedly, more sophisticated features. Furthermore, storing the *n*-gram frequencies in a large hash table ensures very competitive runtime on machines with sufficient main memory. However, until now, no explanation was given why the exponential normalization scheme of naïve query segmentation performs so well. We close this gap with an in-depth analysis in Section 6.2. Our new segmentation method, which is inspired by naïve query segmentation, is introduced in [82].

Finally, the very recent approach of Mishra et al. [140] compares with our method in terms of feature complexity. But instead of web *n*-gram frequencies, Mishra et al. exploit *n*-gram frequencies from a large query log. Their experiments on a proprietary query corpus indicate an improvement over a mutual information baseline.

6.2 Two Query Segmentation Algorithms

We regard a keyword query q as a sequence $(w_1, w_2, \ldots, w_k)$ of k keywords. A valid segmentation S for q is a sequence of disjunct segments s, each a contiguous subsequence of q, whose concatenation equals q. There are 2^{k-1} valid segmentations for q, and $(k^2 - k)/2$ potential segments that contain at least two keywords from q.

The basic and major assumption of both our approaches is that phrases contained in queries must exist on the web—otherwise they cannot increase retrieval performance. The idea then is to use the web as a corpus of potential query phrases. The largest collection of web phrases obtainable, besides the web itself, is the Google n-gram corpus [26]. It contains n-grams of length 1 to 5 from the 2006 Google index along with their occurrence frequencies. For n-grams up to $n = 5$, the frequencies can be directly retrieved from the corpus; for longer n-grams up to $n = 9$, estimations can be made similar to the set-based method described in [220]. For example, the frequency of a 6-gram $ABCDEF$ can be lower-bounded by the sum of the Google n-gram frequencies of the 5-grams $ABCDE$ and $BCDEF$ decreased by the frequency of their 4-gram intersection $BCDE$. In practice, however, despite being set-theoretically sound, these lower bound estimations often evaluate to 0 (i.e., the intersection's frequency is too large to be compensated by the frequencies of the two longer n-grams). That said, there are still situations where non-zero lower bound estimations appear and do some good.

Using the web occurrence frequencies from the Google n-gram corpus and the set-based estimations, both our approaches score and rank all possible segmentations of a query. They apply normalization schemes to the raw n-gram frequencies and allow long segments to achieve scores comparable to their shorter sub-segments. For example, `blue jays` will always have a larger raw n-gram frequency than `toronto blue jays`, but the latter should be the preferred segmentation in queries concerning the baseball team. In the following

section, we detail two normalization approaches: a naïve normalization scheme as well as a normalization scheme which incorporate knowledge from Wikipedia.

6.2.1 Naïve Normalization

Naïve query segmentation computes a score for each valid segmentation S of a query q as follows. The n-gram frequencies $freq(s)$ of all potential segments s with at least two words are retrieved. The frequencies at hand, all valid segmentations are enumerated, and each segmentation S is scored according to the following function:

$$score(S) = \begin{cases} \sum_{s \in S, |s| \geq 2} |s|^{|s|} \cdot freq(s) & \text{if } freq(s) > 0 \text{ for all } s \in S, |s| \geq 2 \\ -1 & \text{else.} \end{cases}$$

The weight factor $|s|^{|s|}$ is a means of normalizing $freq(s)$ to compensate the power law distribution of web n-gram frequencies. This way, a long segment s has a chance of being selected compared to its shorter sub-segments. For a query q, the valid segmentations are ranked according to their scores and naïve query segmentation selects the one that maximizes $score(S)$.

For example, `toronto blue jays` has an n-gram frequency of 0.8 million, which is smaller than the 1.4 million of `blue jays`; simply using the length $|s|$ of s as weight factor does not help to prefer the three-word segment (*score* of 2.4 million vs. 2.8 million). However, using the exponential weight factor $|s|^{|s|}$, the segmentation `"toronto" "blue jays"` with a *score* of 5.2 million is discarded in favor of `"toronto blue jays"` with a *score* of 21.6 million.

In the above scoring function, the n-gram frequencies of single word segments are implicitly set to 0, so that the "null"-segmentation that consists of only single word segments—the unquoted query—gets a *score* of 0 (the else-case does not match and the summation is

empty). A segmentation gets a negative score of -1 if it contains a segment (with at least two words) that does not exist in the web n-gram corpus. Such a segmentation cannot help to improve retrieval performance since the non-existent segment is not found on the web at all, that is, in our n-gram representation of the web. Scoring such segmentations with a negative score ensures that the unquoted query will be chosen as a fallback solution in case all other valid segmentations contain non-existent phrases.

Compared to other methods, naïve query segmentation is the most basic approach as it uses only raw n-gram frequencies instead of many different features. Also, the scoring can be explained (and implemented) within a few lines (of code). What remains to be illustrated is why the exponential scoring performs well in practice, and not only for the `toronto blue jays`.

Empirical Justification In what follows, we provide empirical evidence that the exponential scoring scheme of naïve query segmentation reproduces human preferences of segmenting queries. This sheds light on why the method achieves its convincing segmentation accuracy reported in [81] and in Section 6.3. To this end, we take a closer look at the underlying data used in the naïve scoring scheme and its evaluation: the Google n-gram frequencies and the 500 human-annotated queries from the Bergsma-Wang-Corpus.

The Google n-grams contain occurrence frequencies for all 1- to 5-grams that appeared more than 40 times within Google's web index as of 2006. Some overview figures are given in Table 6.1, including the average and the median frequencies for all 2- to 5-grams in the original corpus (rows "all") as well as for a cleaned version of the corpus (rows "cleaned") from which all n-grams were removed that contain non-alphanumeric characters. We assume that n-grams occurring in real web queries are better represented by the cleaned corpus. Note that the resulting average and median frequencies are quite low and thus illustrate the presumed long tail distribution of n-gram frequencies (most n-grams have very low frequencies).

Table 6.1: Overview of the Google n-gram corpus.

Corpus Subset		Entries	Frequency	
			Average	Median
2-grams	all	314 843 401	2 893	110
	clean	215 114 473	2 065	108
3-grams	all	977 069 902	756	91
	clean	492 457 391	608	89
4-grams	all	1 313 818 354	387	80
	clean	556 957 524	330	79
5-grams	all	1 176 470 663	300	75
	clean	421 372 587	258	73

To experimentally justify naïve scoring, we address the situation of a query in which a long n-gram should be favored over its shorter prefixes in a segmentation. The long n-gram then has to obtain a better score than any of its shorter prefixes. To simulate this scenario, we sampled 10 million 5-grams from the cleaned 5-gram corpus. The sampling probability is chosen proportional to a 5-gram's frequency, so that frequent 5-grams are favored. We speculate that favoring frequent 5-grams for our sample makes it more representative of the most frequent segments observed in web queries. From all of the sampled 5-grams, we computed all prefixes of length 2 to 4 words.

Table 6.2 gives an overview of the 5-gram sample and their prefixes. Observe that the median frequencies of the sample show an exponential decrease. In order to further compare this decrease to

Table 6.2: Overview of the sample of 10 million 5-grams.

$\lvert s\rvert$-grams	Unique Entries	Median Freq.	$\frac{\text{2-gram Freq.}}{\lvert s\rvert\text{-gram Freq.}}$	$\lvert s\rvert^{\lvert s\rvert}$
2-grams	2 409 063	3 461 030	1	4
3-grams	5 431 544	78 733	44	27
4-grams	8 073 863	7 356	470	256
5-grams	10 000 000	1 129	3 065	3 125

Table 6.3: Expected *score* in the 5-gram sampling experiment.

$\|s\|$-grams	Median Freq.	$\|s\|^{\|s\|}$-scoring	"Hard-Wired"
2-grams	3 461 030	13 844 120	3 461 030
3-grams	78 733	2 125 791	3 464 252
4-grams	7 356	1 883 136	3 457 320
5-grams	1 129	3 528 125	3 460 385

the $|s|^{|s|}$-scoring, we also provide the ratio of the median 2-gram frequency to the median s-gram frequency. Note that the last two columns of Table 6.2 show that the $|s|^{|s|}$-compensation for the exponential frequency decrease is always in the correct order of magnitude. In this connection, one may wonder why not to choose the observed ratios instead of $|s|^{|s|}$-scoring (i.e., multiplying $freq(s)$ of a 3-gram s with 44 instead of 3^3, etc.). To test this possibility, we performed a pilot study where the ratios from Table 6.2 were used as "hard-wired" weight factors instead of $|s|^{|s|}$: the achieved segmentation accuracy dropped significantly.

An explanation for this behavior can be found when comparing the expected median *score*-values of our sample with the query corpus used in our evaluation. As can be seen in Table 6.3, the hard-wired scoring achieves a very good normalization of the expected *score*-values for the 5-gram sample in the sense that all median *score*-values have the same order of magnitude. Then again, $|s|^{|s|}$-scoring clearly favors 2-grams, whereas the longer n-grams are somehow

Table 6.4: Segment length distribution of human segmentations.

Annotator	Segment Length					
	1	2	3	4	5	6
A	451	699	74	14	2	
B	351	541	113	77	9	2
C	426	588	100	51	5	1
Agree	151	318	31	9		

"on par." To compare this with human segments, Table 6.4 compiles the segment length distribution of the 3 human annotators of the widely used Bergsma-Wang-Corpus. The last row of Table 6.4 contains only queries that all three annotators segmented the same way. Observe that human annotators also favor 2-grams instead of longer segments, especially in queries they all agree upon. The $|s|^{|s|}$-scoring of naïve query segmentation reproduces this behavior on our 5-gram sample, which explains, to some extent, why it performs so well.

6.2.2 Wikipedia-Based Normalization

In pilot experiments, the aforementioned approach which normalizes each segment's frequency with hard-wired average frequencies shows an inferior segmentation accuracy compared to the naïve approach that involves a similar, but less even normalization scheme. Nevertheless, normalizing segment frequencies with average frequencies also bears the idea of normalizing in a segment-specific way, which is exactly what we propose for our Wikipedia-based normalization scheme. As the name suggests, the new scheme involves another feature besides the raw n-gram frequencies, namely a Wikipedia title dictionary obtained from a dump of the English Wikipedia. Note that Tan and Peng [220] also use this feature in their segmentation approach. Our dictionary contains all Wikipedia titles and their respective disambiguations, but no words from within articles. Otherwise, the new normalization scheme is just as simple as naïve query segmentation.

Again, the n-gram frequencies $freq(s)$ of all potential segments s with at least two words are retrieved. For each segment s, we check whether it is present in the Wikipedia title dictionary. If a segment s appears in the dictionary, we replace its frequency $freq(s)$ with the maximal Google n-gram frequency found among the sub-2-grams

$s' \sqsubset s$, given by the following *count*:

$$count(s) = \begin{cases} |s| + \max\limits_{\substack{s' \sqsubset s \\ |s'|=2}} freq(s') & \text{if } s \text{ is a title in Wikipedia} \\ freq(s) & \text{else.} \end{cases}$$

This way, the normalization of a segment's frequency is segment-specific in that every potential segment's *freq*-value is treated separately rather than normalizing it with some average frequency. Note that $|s|$ is added to the maximal sub-2-gram frequency for convenience reasons, as it allows us to prove that queries that consist of a single Wikipedia title will not be split into sub-segments (shown in the next subsection). Otherwise, adding $|s|$ had no measurable effect in our experiments so that a query segmentation system "in the field" could safely omit it. Based on the Wikipedia-normalized *count*-values, a valid segmentation S of q is scored as follows:

$$score(S) = \begin{cases} \sum\limits_{s \in S, |s| \geq 2} |s| \cdot count(s) & \text{if } count(s) > 0 \text{ for all } s \in S, |s| \geq 2 \\ -1 & \text{else.} \end{cases}$$

Again, for a query q we choose from all valid segmentations the one that maximizes $score(S)$.

Remarks The Wikipedia-based normalization can run into the special case of encountering a "new" concept s that is present in the Wikipedia titles but not in the n-gram corpus (e.g., some new product or brand that did not exist back in 2006). If all the sub-2-grams of s exist in the n-gram corpus, this is not an issue since the Wikipedia-based normalization still works. Otherwise, $count(s)$ would be set to $|s|$, although s is prominently placed in Wikipedia, which is not satisfactory. We tackle this problem by a simple additional rule for

the more general case that only some sub-2-grams of s are not yet present in the n-gram corpus. In that case, we set the missing 2-gram frequencies to the median 2-gram frequency 3 461 030 from Table 6.2. As before, $count(s)$ is set to the maximal frequency found among the sub-2-grams of s (which now also could be the median frequency).

Another straightforward method of circumventing the situation of Wikipedia titles being out of sync with the n-gram corpus exists at search engine site: to update the n-gram corpus in real-time. But we expect that n-gram updates will still be done less frequently compared to updating the Wikipedia title dictionary. And since Wikipedia typically contains pages on new, important "concepts" very quickly, setting the corresponding frequencies to the median 2-gram frequency is a reasonable heuristic. As a side note, in our experiments we did not find any "new" Wikipedia concepts since the Bergsma-Wang-Corpus and our newly developed corpus both stem from the AOL query log of 2006 and thus fit the 2006 Google n-gram corpus very well.

Theoretical Justification The idea of Wikipedia-based normalization is similar to that of the naïve normalization approach, namely giving longer segments a chance to be preferred over their shorter sub-segments. But now this goal is achieved by using Wikipedia for segment re-weighting instead of some "magic," yet powerful exponential factor. Wikipedia serves as a database of named entities that helps to identify widely known concepts like names, places, etc. These concepts should not be split up when segmenting a query. In case a concept forms an entire query, it can be proven that the above scoring function will prefer the complete concept as one segment.

Lemma 1. *Let s be a sequence of words found in the Wikipedia title dictionary. Whenever a query $q = s$ shall be segmented, the Wikipedia-based scoring function will choose the segmentation "s" without any intermediate quotes.*

Proof. Let ℓ be the largest *freq*-value of any sub-2-gram of s. As s is found in the Wikipedia title dictionary, the segmentation "s" gets $score(\text{"}s\text{"}) = |s| \cdot (\ell + |s|)$.

Now consider any other valid segmentation S of s that splits s into sub-segments and gets a positive score by the Wikipedia-based normalization scoring function. The worst possible case is that all sub-segments of s also are Wikipedia concepts and that all sub-2-grams of s have ℓ as their *freq*-value. Note that this worst case scenario is independent of whether all sub-2-grams of s are in the n-gram corpus or not: it does not matter whether ℓ is the median 2-gram frequency from Table 6.2 or not.

As S divides s into non-overlapping sub-segments (otherwise S is not valid), the sum of the length-weight-factors of the sub-segments cannot be larger than $|s|$. Furthermore, the Wikipedia-normalized *count*-value of each of these sub-segments cannot be larger than $\ell + |s| - 1$ as the largest sub-segments of s have size $|s| - 1$. Basic calculus yields $score(S) \leq |s| \cdot (\ell + |s| - 1) < score(\text{"}s\text{"})$ so that the segmentation "s" will be chosen. □

As an example, consider the query `new york yankees`—to stay within the baseball domain. The relevant Google n-gram frequencies for this query are 165.4 million for `new york` and 1.8 million for `new york yankees`. Note that naïve query segmentation would segment the concept as "`new york`" "`yankees`" since this achieves a naïve scoring of 661.6 million compared to 28.8 million for "`new york yankees`". However, with Wikipedia-based normalization, `new york yankees` gets as *count* the 165.4 million frequency of `new york`. Hence, "`new york yankees`" achieves a 496.2 million *score* compared to 330.8 million for "`new york`" "`yankees`".

The only way that a Wikipedia title s is split up during segmentation is when a word in front of or after s co-occurs very often with a prefix or a suffix of s, respectively. This is especially the case when two Wikipedia titles are interleaved within a query. However, s is

only split up if the *score*-value of some segmentation containing a sub-segmented s is superior to all the segmentations containing s as a complete segment. This is in line with the rationale that larger *score*-values represent better segmentations.

An example where a Wikipedia concept is split, is the query `times square dance` from the introduction. The relevant segments that appear in the Wikipedia title dictionary are `times square` and `square dance`. Based on the Google n-gram corpus, `times square` gets a 1.3 million *count* and `square dance` achieves 0.2 million. Our new segmentation scoring function then assigns `"times square" dance` the highest *score* of 2.6 million while `times "square dance"` achieves 0.4 million—either way, a Wikipedia concept is split.

It is of course debatable whether the segmentation `"times square" dance` is a good choice, since we have pointed out the ambiguity of this query. This situation can only be resolved by incorporating information about the user's context. For instance, if the user stems from London, reads "The Times" and is a passionate folk-dancer, this might make the alternative segmentation `times "square dance"` preferable. If no such context information is at hand, there is still another option: the search engine may present the results of the best scoring segmentation to the user and offer the second best segmentation in a "Did you mean" manner.

Besides the theoretical justification for Wikipedia-based normalization, our new method also shows very promising experimental performance with respect to segmentation accuracy against human-segmented queries, as the next section shows.

6.3 Evaluating Query Segmentation Algorithms

In this section we report on an evaluation that compares our approach to 7 others proposed in the literature. We employ the performance measures proposed in [23, 220], use a mutual information-based method as baseline, and evaluate against three corpora: the

Bergsma-Wang-Corpus 2007 [23], an enriched version of that corpus, and a new, significantly larger corpus, called Webis Query Segmentation Corpus 2010. The latter two have been compiled with the aid of Amazon's Mechanical Turk. In this connection, we also compare segmentations from experts with those of laymen.

6.3.1 Performance Measures

Performance evaluation of query segmentation algorithms is twofold. On the one hand, *runtime* is crucial since query segmentation must be done on-the-fly during retrieval. Runtime is typically given as throughput (i.e., the number of queries segmentable per second). On the other hand, the computed segmentations should be as accurate as possible. There are three levels on which to measure segmentation accuracy in a supervised manner:

Query Level. As a whole, a computed segmentation of a query is correct iff it contains exactly the same segments as a human reference segmentation. Hence, the *query accuracy* is the ratio of correctly segmented queries for a given corpus.

Segment Level. Let S denote the set of segments of a human segmentation of a query q. A computed segmentation S' can be evaluated using the well-known measures precision and recall. *Segment precision* and the *segment recall* are defined as follows:

$$\text{seg prec} = \frac{|S \cap S'|}{|S'|} \qquad \text{and} \qquad \text{seg rec} = \frac{|S \cap S'|}{|S|}.$$

Both values can be combined into a single score by computing their harmonic mean, called *segment F-Measure* :

$$\text{seg F} = \frac{2 \cdot \text{seg prec} \cdot \text{seg rec}}{\text{seg prec} + \text{seg rec}}.$$

Break Level. Note that query segmentation can also be considered a classification task in which, between each pair of consecutive words in a query, a decision has to be made whether or not to insert a segment break. This allows for $k-1$ potential break positions in a query with k keywords. For every break position, the computed segmentation may either decide correctly or not, according to a reference segmentation. The *break accuracy* measures the ratio of correct decisions.

As an illustration, consider the query `san jose yellow pages` with the reference segmentation `"san jose" "yellow pages"`. A computed segmentation `"san jose" yellow pages` is not correct on query level, resulting in a query accuracy of 0. However, on segment-level, `"san jose" yellow pages` at least contains one of the two reference segments, yielding a segment recall of 0.5. But since the other two single word segments are not part of the reference segmentation, precision is 0.333, yielding a segment F-Measure of 0.4. The break accuracy is 0.666, since `"san jose" yellow pages` decides incorrectly only for one of the three break positions.

6.3.2 Baseline: Mutual Information

As a baseline for query segmentation we adopt the mutual information method (MI) used throughout the literature. A segmentation S for a query q is obtained by first computing the pointwise mutual information score for each pair of consecutive words (w_i, w_{i+1}) in q, with $i \in \{1, \ldots, k-1\}$ and $k = |q|$:

$$\mathrm{PMI}(w_i, w_{i+1}) = \log \frac{p(w_i, w_{i+1})}{p(w_i) \cdot p(w_{i+1})},$$

where $p(w_i, w_{i+1})$ is the joint probability of occurrence of the 2-gram (w_i, w_{i+1}), and $p(w_i)$ and $p(w_{i+1})$ are the individual occurrence probabilities of the two words w_i and w_{i+1} in a large text

corpus. Second, segment breaks are introduced into q whenever the pointwise-mutual information score of two consecutive words is below a pre-specified threshold τ (i.e., when $\text{PMI}(w_i, w_{i+1}) < \tau$).

Within our evaluation, the probabilities for all words and 2-grams have been computed using the Microsoft Web N-Gram Services [91].[1] More specifically, the language model of webpage bodies from April 2010 has been used. We recorded all probabilities for all our corpora in order to ensure replicability. For our experiments, we chose $\tau = 0.894775$, which maximizes the MI method's break accuracy on the Bergsma-Wang-Corpus.

6.3.3 The Bergsma-Wang-Corpus 2007

The Bergsma-Wang-Corpus 2007 (BWC07) consists of 500 queries which have been sampled from the AOL query log dataset [153]. The sample was chosen at random from the subset of queries that satisfy all of the following constraints:

- A query consists of only determiners, adjectives, and nouns.
- A query is of length 4 words or greater.
- A query has been successful (i.e., the searcher clicked on one of the search result URLs returned by the search engine).

The query sample was then segmented independently by 3 annotators, who were instructed to first guess the user intent based on the query in question and the URL the user clicked on, and then segment the query so that it describes the user intent more clearly. In 44% of the queries, all three annotators agree on the segmentation, and in about 60% of the cases, at least two annotators agree.

[1] http://web-ngram.research.microsoft.com

Remarks The BWC07 has a number of shortcomings that render evaluations based on this corpus less insightful: the query sample of the corpus is not representative, partly because of the small number of queries and partly because of the sampling constraints. The first constraint particularly raises concerns since its motivation was to accommodate design limitations of the proposed query segmentation algorithm; the authors stated that "*as our approach was designed particularly for noun phrase queries, we selected for our final experiments those AOL queries containing only determiners, adjectives, and nouns*" [23]. The other two constraints are less of a problem, though one might argue that queries of length 3 should also be subject to segmentation, and that, unless query quality predictors are employed, a search engine cannot know in advance whether a query will be successful. Moreover, the number of annotators per query appears to be too small, since in 40% of the queries no agreement was achieved. This, in turn, tells something about the difficulties involved in manually segmenting queries. On a minor point, there are also a few duplicate queries, spelling errors, and character encoding errors. The former have a measurable effect on segmentation performance, given the small size of the corpus, while the latter two should not be part of a query segmentation corpus. Query segmentation does not necessarily involve spell checking and encoding normalization, as those may be treated as separate problems. However, none of the above should be held to the disadvantage of the corpus' authors, since their intentions were first and foremost to evaluate their approach on a new retrieval problem, and not to construct a reference corpus, which it became nonetheless.

Experiments Several previous studies evaluate their query segmentation algorithms against the BWC07 [23, 29, 81, 220, 239]. We follow suit, to ensure comparability to the existing evaluations. However, prior to that we chose to correct the aforementioned minor errors and remove duplicate queries; the cleaned corpus consists of 496 queries.

Since the existing studies do not mention any problems with the BWC07, we have asked the respective authors for their segmentations of the BWC07 queries. Our rationale for this is to avoid an error-prone reimplementation of the existing algorithms, since their output for the BWC07 suffices to recalculate the accuracy measures. The authors of [23, 29, 81, 239] kindly provided their segmentations. Based on this data, we have been able to verify the performances reported in the respective papers, and to reevaluate the algorithms on the cleaned version of the BWC07.

Table 6.5 shows the results of our evaluation. Each column presents the segmentation accuracies of one algorithm. The rightmost two columns show the results of our naïve query segmentation [81] and those of Wikipedia-based normalization [82]. The table should be read as follows: three annotators—A, B, and C—independently segmented the 496 queries of the BWC07 and they agreed on 218 of them, denoted in the rows "Agree." The rows "Best of A, B, C" evaluate simultaneously against the up to three reference segmentations of A, B, and C, choosing the one that maximizes a computed segmentation's break accuracy. Note that, compared to the originally published results, the segmentation accuracies of most systems slightly decrease in our evaluation. This is due to the removed duplicate queries that are rather "easy" and correctly segmented by most systems. An exception is the remarkable performance of our mutual information baseline with a significant improvement over previously reported values. One reason for this is that the language model underlying the Microsoft Web N-Gram Services presumably gives more accurate probabilities than those used in previous evaluations. However, more importantly, note that the baseline's threshold τ was derived on the BWC07 queries with the explicit aim of maximizing the resulting break accuracy. Hence, it represents the best case MI approach for the BWC07 queries and yields a challenging baseline.

Table 6.5: Segmentation performance on the Bergsma-Wang-Corpus.

Annotator	Performance Measure	Algorithm							
		MI	[23]	[220]	[239]	[29]	[140]	[81]	[82]
A	query	0.407	**0.609**	0.526	0.518	0.540	0.256	0.597	0.573
	seg prec	0.553	**0.748**	0.657	0.673	0.686	0.476	0.724	0.706
	seg rec	0.550	**0.761**	0.657	0.650	0.672	0.566	0.711	0.679
	seg F	0.552	**0.754**	0.657	0.662	0.679	0.517	0.717	0.692
	break	0.761	**0.859**	0.810	0.810	0.797	0.681	0.826	0.826
B	query	0.413	0.435	0.494	0.504	0.383	0.185	0.438	**0.508**
	seg prec	0.539	0.582	0.623	**0.637**	0.527	0.369	0.577	0.635
	seg rec	0.548	0.608	**0.640**	0.632	0.533	0.450	0.583	0.627
	seg F	0.544	0.595	0.631	0.634	0.530	0.405	0.580	**0.631**
	break	0.765	0.783	0.802	**0.811**	0.741	0.598	0.777	0.803
C	query	0.417	0.472	0.494	0.484	0.440	0.224	0.480	**0.508**
	seg prec	0.553	0.623	0.634	0.627	0.580	0.424	0.618	**0.647**
	seg rec	0.557	**0.643**	0.642	0.613	0.575	0.515	0.613	0.632
	seg F	0.555	0.633	0.638	0.620	0.578	0.465	0.615	**0.640**
	break	0.764	0.795	**0.796**	0.789	0.755	0.639	0.777	0.795
Agree	query	0.555	0.688	0.671	0.670	0.615	0.294	0.693	**0.720**
	seg prec	0.659	0.792	0.767	0.787	0.731	0.491	0.789	**0.810**
	seg rec	0.665	**0.809**	0.782	0.770	0.724	0.575	0.782	0.792
	seg F	0.662	0.800	0.774	0.779	0.727	0.529	0.785	**0.801**
	break	0.828	**0.889**	0.871	0.883	0.834	0.699	0.868	0.885
Best of A, B, C	query	0.583	0.702	0.692	0.694	0.629	0.333	0.700	**0.726**
	seg prec	0.693	0.812	0.797	0.811	0.749	0.558	0.800	**0.820**
	seg rec	0.697	**0.831**	0.807	0.801	0.746	0.649	0.796	0.807
	seg F	0.695	**0.821**	0.801	0.806	0.747	0.600	0.798	0.814
	break	0.849	0.899	0.891	0.897	0.857	0.736	0.889	**0.900**

Since we did not get the query segmentations of Tan and Peng [220], we include the values they published in their paper (in gray). However, the removal of duplicate queries would most likely decrease their performances as well. Since some previous studies did not report all measures for their algorithms, Table 6.5 contrasts for the first time all performance values for all algorithms that have been evaluated against the BWC07.

The results show that the approach of Bergsma and Wang [23] performs very well on annotator A as well as on the queries all annotators agree upon. However, this is not surprising as their approach is based on a supervised learning algorithm that was explicitly trained on queries segmented by annotator A (the agreement queries also match A's segmentation). This leaves doubts with regard to generalizability, underpinned by the inferior performance of their approach on the two other annotators B and C. As for the other systems, note that our naïve query segmentation algorithm with n-gram frequency normalization achieves a performance comparable to the best approaches of Brenes et al. [29], Tan and Peng [220], and Zhang et al. [239]. Furthermore, note that our Wikipedia-based frequency normalization method outperforms all other methods with respect to query accuracy and segment precision, while being on par with the best performing system at break accuracy.

An Excursus on Retrieval Performance Besides measuring segmentation accuracy against a gold standard, one might as well evaluate whether a segmentation actually yields an improvement in retrieval performance (i.e., retrieving results that are more relevant). To this end, we have conducted the following small-scale experiment to compare the two best performing approaches: Bergsma and Wang's supervised learning approach and our Wikipedia-based normalization scheme. For each of the 218 BWC07 queries on which all three annotators agree, we submit 4 queries to the Bing web search engine, each time storing the top-50 results: the BWC07 segmentation, the

computed segmentation of Bergsma and Wang, the computed segmentation of our method, and the unquoted query. We consider the top-50 results obtained by the BWC07 segmentation as "relevant," which allows us to measure the recall of the other three queries. Averaged over all 218 trials, the supervised learning approach achieves a recall of 0.844, our Wikipedia-based normalization achieves 0.836, whereas unquoted queries achieve a recall of 0.553.

Presuming the user intent is well captured in the segmented queries of the BWC07 (three annotators agreed upon these segmentations), the results might be interpreted as indication that segmenting queries improves recall over unquoted queries. Comparing the average recall of Bergsma and Wang's segmentations and ours also suggests that their worse query accuracy is compensated by their better segment recall so that the overall retrieval recall is comparable. It is important to note that this small experiment is not meant to replace a large-scale TREC-style evaluation, since the results have not been judged manually as relevant or not. Instead, this experiment is meant as a first step toward not just comparing segmentation accuracy. After all, the ultimate goal of query segmentation is to improve retrieval performance.

6.3.4 The Enriched Bergsma-Wang-Corpus

One point of criticism about the BWC07 is the non-agreement of its annotators in 40% of the queries. We attribute this problem to the fact that not all queries are the same, and that some are more ambiguous than others. A search engine user who poses an ambiguous query would know of course, if being asked, how exactly to segment it, whereas an annotator has a hard time figuring this out afterwards. One way to overcome this problem is to collect more opinions from different annotators and then make a majority decision. Recently, paid crowdsourcing has become an important tool in this respect, enabling researchers to significantly scale up corpus construction.[2]

[2]See [3], and our previous works [163, 171].

Table 6.6: Segmentation performance of Amazon's Mechanical Turk (AMT) workers on the Bergsma-Wang-Corpus.

Annotator	**Performance Measure**	**"Algorithm"** AMT
Best of A, B, C	query	0.821
	seg prec	0.892
	seg rec	0.883
	seg F	0.887
	break	0.941

Amazon's Mechanical Turk (AMT) is a well-known platform for paid crowdsourcing. In short, it acts as a broker between workers and so-called requesters, who offer tasks and payment for their successful completion. Since real money is involved and since workers remain anonymous, the platform attracts scammers who try to get paid without actually working. Hence, requesters get the opportunity to check submitted results and reject those that are unsatisfactory. Besides saving money, rigorous result checking is of course a necessity to ensure quality.

In an effort to enrich the BWC07, we have offered a query segmentation task on AMT in which we asked to segment the queries of this corpus. At least 10 valid segmentations per query were collected, while manually checking and rejecting invalid ones. To measure the success of our initiative and to learn about how much laymen agree with experts, we compute all of the aforementioned performance measures, treating the segmentations obtained from AMT as if they were segmentations returned by an algorithm. For each query, the segmentation that was submitted most often by the workers was used. The results of this comparison are shown in Table 6.6. Again, "Best of A, B, C" means that from the three reference segmentations of A, B, and C the one is chosen for comparison that maximizes the AMT workers' break accuracy. Note that the workers of AMT outperform the algorithmic query segmentation approaches

Table 6.7: Segmentation performance on the *enriched* Bergsma-Wang-Corpus.

Annotator	Performance Measure	Algorithm							
		MI	[23]	[220]	[239]	[29]	[140]	[81]	[82]
Best of AMT	query	0.738	0.812	n/a	0.831	0.794	0.546	**0.859**	0.857
	seg prec	0.802	0.890	n/a	0.891	0.869	0.758	**0.909**	0.908
	seg rec	0.806	0.904	n/a	0.889	0.868	0.830	**0.909**	0.908
	seg F	0.804	0.897	n/a	0.890	0.868	0.792	**0.909**	0.908
	break	0.916	0.944	n/a	0.945	0.924	0.850	0.950	**0.952**
Best of A, B, C and AMT	query	0.754	0.825	n/a	0.849	0.802	0.546	**0.873**	0.867
	seg prec	0.809	0.897	n/a	0.910	0.876	0.753	**0.921**	0.917
	seg rec	0.813	0.910	n/a	0.910	0.874	0.825	**0.921**	0.914
	seg F	0.811	0.903	n/a	0.910	0.875	0.787	**0.921**	0.916
	break	0.920	0.947	n/a	0.954	0.928	0.847	**0.957**	0.956

(cf. Table 6.5) on all accounts. However, the AMT workers cannot achieve perfect segmentation accuracy when compared to the expert segmentations of BWC07, since even experts make errors.

Having established that the segmentations obtained via AMT are indeed valid, we reevaluated all of the segmentation algorithms against the AMT segmentations as well as against the combination of the BWC07 segmentations and the AMT segmentations. The results of this evaluation are shown in Table 6.7. As can be observed, our naïve query segmentation algorithm [81] performs best on the enriched BWC07. Our Wikipedia-based normalization comes in second. A speculative explanation for the remarkable performance of the naïve $|s|^{|s|}$-scoring might be that web n-gram frequencies are correlated with the n-grams an average AMT worker knows.

6.3.5 The Webis Query Segmentation Corpus

Given the encouraging results achieved with enriching the Bergsma-Wang-Corpus by means of crowdsourcing, the logical next step is to build a larger corpus, thus addressing the remaining points of criticism about the BWC07, namely its small size and the sampling

Table 6.8: Segmentation performance on the Webis Query Segmentation Corpus.

Annotator	Performance Measure	Algorithm		
		MI	[81]	Our
Best of AMT	query	0.598	0.599	**0.616**
	seg prec	0.727	0.736	**0.744**
	seg rec	0.738	0.733	**0.739**
	seg F	0.732	0.734	**0.742**
	break	0.844	0.842	**0.850**

constraints. Following a new sampling strategy (detailed below), we have sampled 50 000 queries from the AOL query log. These queries were checked for correctness of spelling and encoding, and then segmented by workers recruited on Amazon's Mechanical Turk. We followed the same strategy as before, however, this time using the enriched BWC07 as check queries to identify workers who perform poorly. As a result, we present the new Webis Query Segmentation Corpus 2010 (Webis-QSeC-10).[3] Finally, we have evaluated our query segmentation algorithms as well as the baseline algorithm against this corpus, the results of which are shown in Table 6.8. Unsurprisingly, the absolute performance values are far below those on the BWC07, since our new corpus contains the whole spectrum of queries and not only noun phrase queries. While the mutual information baseline allows for some comparability to the earlier results, a complete comparison of all algorithms against our new corpus is still missing. So far, our Wikipedia-based normalization scheme performs best on this corpus, but once again, mutual information serves as a reasonable and challenging baseline.

Corpus Construction The 50 000 queries for our corpus were chosen in three steps from the AOL query log: first, the raw query log was filtered in order to remove ill-formed queries; second, from the

[3] http://www.webis.de/research/corpora

remainder, queries were sampled at random; and third, the sampled queries were corrected. In the filtering step, queries were discarded according to the following exclusion criteria:

- Queries comprising remnants of URLs (navigational queries) or URL character encodings.
- Queries from searchers having more than 10 000 queries.
- Queries from searchers whose average time between consecutive queries is less than 1 second.
- Queries from searchers whose median number of letters per query is greater than 100.
- Queries that contain non-alphanumeric characters except for dashes and apostrophes in-between characters.
- Queries that are shorter than 3 words or longer than 10.
- Queries from searchers that duplicate preceding queries of themselves (result page interaction).

Of the 36 389 567 queries in the raw AOL query log, 6 027 600 queries remained after filtering. The majority of the filtered queries (22.8 million) were removed by the criteria pertaining to special characters and query length (e.g., queries shorter than 3 words).

In the sampling step, 50 000 queries were chosen at random from the filtered query log, while maintaining the original distributions of query frequency and query length. To accomplish this, the log was divided into query length classes, where the i-th class contains all queries of length $i \in \{3, \ldots, 10\}$, keeping duplicate queries from different searchers. Then, the query length distribution was computed and the amount of queries to be expected for each length class in a 50 000 query sample was determined (see Table 6.9). Based on these expectations, for each length class, queries were sampled without replacement until the expected amount of distinct queries was reached. Hence, our sample represents the query length distribution

Table 6.9: Overview of the Webis-QSeC-10 query sample.

Length	AOL Queries	Distribution	Sample
3	2 750 697	45.64%	22 820
4	1 620 818	26.89%	13 445
5	846 449	14.04%	7 020
6	418 621	6.95%	3 475
7	202 275	3.36%	1 680
8	102 792	1.70%	850
9	55 525	0.92%	460
10	30 423	0.50%	250
Σ	6 027 600	100.00%	50 000

of the filtered AOL log. And since each length class in the filtered log contained duplicate queries according to their frequency, our sample also represents the log's query frequency distribution.

In the final step, we attempted to correct spelling errors present in the sampled queries by means of semi-automatic spell checking. We collected a 1 million word dictionary of words by combining various dictionaries and other sources for often-checked words available online, such as aspell, WordNet, Wiktionary, and Wikipedia titles, to name only a few. Using this dictionary as well as their Google *n*-gram counts, we applied the statistical spell checker proposed by Peter Norvig[4] to the query sample. However, we did not follow the spell checker blindly, but reviewed each replacement manually. This way, about 14% of the queries in our sample have been corrected. It must be mentioned, though, that not all errors can be identified this way, and that correcting the queries will remain an ongoing task.

Corpus Anonymization The AOL query log has been released without proper anonymization, other than replacing the searchers' IP addresses with numerical IDs. This raised a lot of concerns among researchers as well as in the media, since some AOL users could be

[4] http://norvig.com/spell-correct.html

personally identified by analyzing their queries. We address this problem in our corpus by removing the searcher IDs entirely. This way, only queries from our sample that are unique in the raw log may be mapped back onto their original searcher IDs, if someone would choose to do so.

6.3.6 Runtime

As mentioned at the outset, it is important to achieve a low runtime per query in order for a query segmentation algorithm to be practical. Since our method heavily relies on looking up potential segments in the Google n-gram corpus, an efficient implementation of an external hash table is required. We have used the approach described in [27], which employs a minimal perfect hash function, to implement a hash table that maps n-grams to their frequencies. This hash table fits in 13 GB of main memory. The implementation of our query segmentation method is capable of segmenting about 3 000 queries per second on a standard PC. Unfortunately, for lack of implementations of the other query segmentation algorithms, we cannot yet report their runtime performances. However, given the sometimes high number of features proposed in the literature and their complexities, we doubt that any of these approaches can beat ours in a fair comparison.

6.3.7 Conclusion and Future Work

We introduced new approaches to query segmentation that are competitive with state-of-the-art algorithms in terms of segmentation accuracy, while simultaneously being more robust and less complicated. These approaches can be understood as normalization schemes for n-gram frequencies, one of which is based on Wikipedia background knowledge. All relevant feature values are preprocessed and stored in a hash table, rendering the approach very

fast and efficient. We provided theoretical and empirical arguments to better understand the rationale of our approach. For this purpose, a much larger evaluation corpus became necessary. We developed a new query segmentation corpus with 50 000 queries, which is two orders of magnitude larger than the reference corpus used in earlier publications. We introduced this new corpus and its construction via Amazon's Mechanical Turk. We are planning to release the corpus in the course of an open query segmentation competition.

Furthermore, we pointed out several directions for future research. The current evaluation of query segmentation algorithms relies solely on comparing segmentation accuracy against corpora of human-segmented queries, which is fine in itself, but which cannot tell whether query segmentation is actually useful in practice. Hence, future evaluations should also focus on the question whether query segmentation leads to a significant improvement of retrieval performance. In this paper, first steps in this direction were taken.

Another promising future research task is to analyze more elaborate performance measures for query segmentation algorithms. Since many queries are ambiguous, measures that quantify rankings of alternative segmentations for a given query should be investigated. In practice, it is an interesting option for a search engine to deal with ambiguous segmentations by presenting the results of the top-ranked segmentation, but offering also the second-ranked segmentation in a "Did you mean" manner. During our experiments, we observed that whenever our approaches did not match the BWC07 segmentation for the queries on which all three annotators agreed, the systems' second-ranked segmentation often did.

Chapter 7

Web N-Grams for Writing Assistance

Writers who are in doubt about a certain expression often ask themselves: *"What would others write?"* This question can be answered statistically when given a huge corpus of written text from which examples can be retrieved. In this chapter, we study how the whole web in the form of *n*-grams can be reused as a collection of examples of language use, giving an overview of our respective publications [173, 183, 214]. We introduce NETSPEAK, a new kind of search engine for commonly used language.[1]

Our contributions are the following: first, we identify two paradigms by which text production may be supported, namely by increasing text correctness and text commonness. The latter forms the basis for NETSPEAK as a tool to support choosing commonly used words. Then, we present NETSPEAK's retrieval engine which is capable of processing wildcard queries on a large corpus of *n*-grams in a split second. Last, we discuss two user interfaces for this retrieval engine; a textual web interface and the interactive NETSPEAK word graph visualization.

[1] NETSPEAK is available at http://www.netspeak.org.

7.1 Text Correctness Versus Text Commonness

Computer-aided writing has a long history dating back to the very beginning of personal computing. Error checking with respect to spelling and grammar, choosing words, style checking, discourse organization, and text structuring are the central research topics in this domain. Today, numerous word processors are available, ranging from simple editors to full-fledged office environments. However, spell checkers, thesauri and, to a certain extent, grammar checkers are currently the only technologies mature enough in order to be included in such commercial products.

An important observation to be made about most of the available tools and research contributions is that they are solely concerned with the question of whether or not a piece of text is *correct*. The correctness of a text is of course important for it to be understood by its audience, however, there is another factor which affects the understandability of a text, namely its *commonness*. There are numerous ways to correctly write something, but the goal of (non-fictional) writing usually is to maximize understanding in the target audience. This can be accomplished by choosing a wording which is commonly understood by most of these people.

7.1.1 NETSPEAK—A Search Engine for Common Language

There are certainly many different approaches one might think of to support writers with identifying uncommon language and choosing common language instead. We make a first step in this direction by implementing a search engine for common language called NETSPEAK. Our search engine indexes all n-grams found on the web, and it allows its users to pose wildcard queries of literal words and wildcard operators, wherein the literal words must occur in the expression sought after, while the wildcard operators allow to specify uncertainties. For example, a user who is looking for the most com-

mon alternative to be written between the words "rotate" and "axis" can enter these words with a wildcard in between them, and the search engine then retrieves all *n*-grams that match this pattern (see Figure 7.1). The retrieved matching *n*-grams are ranked by their commonness, that is by their occurrence frequency on the web. This way, the user can find confidence in choosing a particular phrase by judging both its absolute and relative frequencies. For example, a phrase may have a low relative frequency but a high absolute frequency, or vice versa, which in both cases indicates that the phrase is not the worst of all choices. Furthermore, example sentences for each phrase are offered, which are retrieved on demand when clicking on the plus sign next to a phrase. This allows users who are still in doubt to get an idea of the larger context of a phrase.

NETSPEAK's intended audience is people who often have doubts about how certain phrases are commonly formed. In particular, language learners and second-language speakers face difficulties in this respect, since their vocabulary as well as their feeling for usage is often not sufficiently developed. They often ask themselves how others would have written what they intend to write; a piece of information that is generally hard to come by. NETSPEAK implements a statistical solution to answer such information needs. Choosing more common phrases over uncommon ones may hold an additional value to this group of people by improving the readability and writing style of a text and by reducing the risk of making errors. Obviously, this is debatable and does not apply to well-versed native writers, but as non-native speakers we found this information immensely helpful in all our daily writing tasks.

Related Work Writing assistance is one of the primary research subjects of computer linguistics. A lot of papers about each of the error detection topics mentioned above have been published, but from a practical point of view, only basic tools such as spell checkers and thesauri are widely used, with the exception of the grammar checker implemented in Microsoft Word.

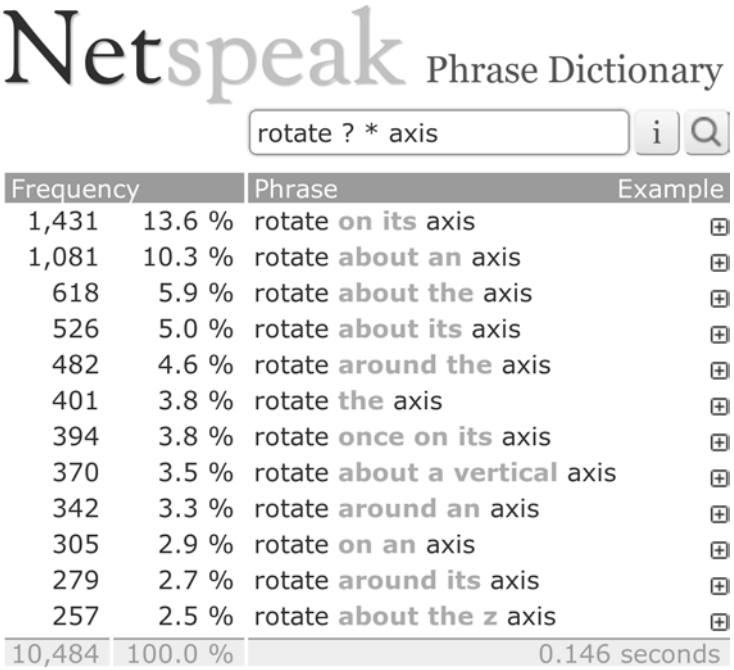


Figure 7.1: NETSPEAK's web interface. Shown are the results for the wildcard query "`rotate ? * axis`", where the `?`-wildcard matches exactly one word and the `*`-wildcard any number of words.

Linguists use search engines comparable to ours (called concordancers) in order to study word semantics and usage. Some are publicly available, for instance WEBCORP, WEBASCORPUS, PHRASESINENGLISH, and LSE.[2] However, since their target audience are linguists and not the average writer, they are not well-suited for our purposes. Moreover, we found in informal comparisons that NETSPEAK outperforms these tools in terms of both retrieval speed and the extent of the indexed language resources.

Corpora of *n*-grams are frequently used in natural language processing and information retrieval for training purposes [127] (e.g., for natural language generation, language modeling, and automatic translation). In particular, there is research on automatic translation within a language in order to correct writing errors [31]. We want to point out that our research is not directed at a complete writing automation since we expect a semi-automatic, interactive writing aid to be more promising in the foreseeable future.

7.2 N-Gram Retrieval with Wildcard Queries

The main building blocks of NETSPEAK are (1) an index of frequent *n*-grams on the web, (2) a query language to formulate *n*-gram patterns, and (3) a wildcard query processor which finds *n*-grams that match a given query and which allows to trade recall for time.

7.2.1 Web Language Index

To provide relevant suggestions in as many contexts as possible, a wide cross-section of written text on the web—if not the entire web—is required as a source of language. This is why we resort to the Google *n*-gram corpus which is currently the largest corpus of its

[2] See http://www.webcorp.org.uk, http://webascorpus.org, http://phrasesinenglish.org, and [181].

Table 7.1: The Google n-grams before and after post-processing.

Corpus Subset	Original Corpus # n-grams	Original Corpus Size	Case Reduction	Vocabulary Filtering
1-gram	13 588 391	177.0 MB	81.34 %	3.75 %
2-gram	314 843 401	5.0 GB	75.12 %	43.26 %
3-gram	977 069 902	19.0 GB	83.24 %	48.65 %
4-gram	1 313 818 354	30.5 GB	90.27 %	49.54 %
5-gram	1 176 470 663	32.1 GB	94.13 %	47.16 %
Σ	3 354 253 200	77.9 GB	88.37 %	54.20 %

kind [26]; it has been compiled from approximately 1 trillion words extracted from the English portion of the web as of 2006 and consists of over 3 billion n-grams along with their occurrence frequencies. Columns 2 and 3 of Table 7.1 give a detailed overview of the corpus. When taking the doubling time of the web into account, which is estimated to be in the order of months while still accelerating, this corpus may already be called "old." But since language evolves in time frames of decades rather than months, the only deficiency of using this corpus is that newly appearing words and company names which become popular are missing. We applied two post-processing steps to the corpus at our site: case reduction and vocabulary filtering. For the latter, a white list vocabulary V was compiled and only these n-grams whose words appear in V were retained. V consists of the words found in the Wiktionary and various other dictionaries, as well as of the words from the 1-gram portion of the Google corpus whose occurrence frequency is above 11 000. See Columns 4 and 5 of Table 7.1 for the size reductions after each post-processing step compared to the original corpus. After post-processing, the corpus was checked for consistency (i.e., whether, for a given n-gram d, all $n-1$-grams are part of the corpus and whether the occurrence frequency of d is greater than or equal to the sum of the frequencies of all $n+1$-grams beginning with d).

Table 7.2: EBNF grammar of NETSPEAK's query language.

Production Rule		
query	=	$\{$ word | wildcard $\}_1^5$
word	=	([" ' "] (letter { alpha })) | " , "
letter	=	" a " | ... | " z " | " A " | ... | " Z "
alpha	=	letter | " 0 " | ... | " 9 "
wildcard	=	" ? " | " * " | synonyms | multiset | optionset
synonyms	=	" ~ " word
multiset	=	" { " word { word } " } "
optionset	=	" [" word { word } "] "

In NETSPEAK the n-gram corpus is implemented as an inverted index μ, which maps each word $w \in V$ onto a postlist π_w. π_w is a list of tuples $\langle d\hat{}, f(d) \rangle$, where $d\hat{}$ refers to an n-gram d on hard disk that contains w, and where $f(d)$ is the occurrence frequency of d reported in the n-gram corpus. A tuple also stores information about w's position as well as other information omitted here for brevity. Note that, since none of the available open source implementations of inverted indexes were capable to index the n-grams in a reasonable amount of time and space, we have implemented a new inverted index. For this purpose, we employ a minimal perfect hash function based on the CHD algorithm which makes our index is space optimal [17].

7.2.2 Query Language

The query language of NETSPEAK is defined by the grammar shown in Table 7.2. A query is a sequence of literal words and wildcard operators, where the literal words must occur in the expression sought after, while the wildcard operators allow to specify uncertainties. Currently, five operators are supported: the question mark, which matches exactly one word; the asterisk, which matches any sequence of words; the tilde sign in front of a word, which matches any of

the word's synonyms; the multiset operator, which matches any ordering of the enumerated words; and the optionset operator, which matches one of the contained words or the empty word. Of course other sensible operators are conceivable, such as constraints on particular parts of speech, person names, places, dates, and times. A less obvious but very interesting enhancement is to constrain words with respect to the expressed positive or negative opinion. Overall objective is to make the query language more powerful yet keeping it simple. The latter is important since the target audience of the service are non-professional writers.

7.2.3 Wildcard Query Processing

Given the n-gram index μ and a wildcard query q, the task is to retrieve all n-grams D_q from μ that match q according to the semantics defined above. This is achieved within two steps: (1) computation of the intersection postlist $\pi_q = \bigcap_{w \in q} \pi_w$, and (2) filtering of π_q with a pattern matcher that is compiled at run-time from the regular expression defined by q. Reaching perfect precision and recall is no algorithmic challenge unless retrieval time is considered. Note in this respect that the length of a postlist often amounts up to millions of entries, which is for instance the case for stop words. If a query contains only stop words, the retrieval time for D_q may take tens of seconds up to a minute, depending on the size of the indexed corpus. From a user perspective this is clearly unacceptable. In cases where a query also contains a rare word w', it is often more effective to apply the pattern matcher directly to $\pi_{w'}$, which is possible since $\pi_q \subseteq \pi_w$ holds for all $w \in q$. But altogether this and similar strategies don't solve the problem as the frequency distribution of the words used in queries will resemble that of written text, simply because of the NETSPEAK use case. Note that web search engines typically get queries with (comparatively infrequent) topic words.

To allow for an adjustable retrieval time at the cost of recall we have devised a query processor, which incorporates rank-awareness within the postlists. Our strategy hence is a special kind of a top-k query processing technique [13, 94]. The strategy requires an offline pre-processing of μ, so that (1) each postlist is sorted in order of decreasing occurrence frequencies, and (2) each postlist is enriched by quantile entries κ, which divide the word-specific frequency distribution into portions of equal magnitude. Based on a pre-processed μ, the retrieval algorithm described above is adapted to analyze postlists only up to a predefined quantile. As a consequence, the portion of a postlist whose frequencies belong to the long tail of the distribution is pruned from the search. Note that the retrieval precision remains unaffected by this.

An important property of our search strategy is what we call *rank monotonicity*: given a pre-processed index μ and a query q, the search strategy will always retrieve n-grams in decreasing order of relevance, independently of κ. This follows directly from the postlist sorting and the intersection operation. An n-gram that is relevant for a query q is not considered if it is beyond the κ-quantile in some $\pi_w, w \in q$. The probability for this depends, among other things, on the co-occurrence probability between q's words. This fact opens up new possibilities for further research in order to raise the recall (e.g., by adjusting κ in a query-specific manner). Such options, however, have not yet been explored.

7.2.4 Evaluation

To evaluate the retrieval quality of our query processing strategy, we report here on an experiment in which the average recall is measured for a set of queries Q, $|Q| = 55\,702$, with respect to different pruning quantiles. The queries originate from the query logs of NETSPEAK; the service is in public use since 2008. We distinguish between

macro-averaged recall and micro-averaged recall:

$$\begin{aligned} \underset{\text{macro}}{rec}(\mu, q) &= \frac{|D_q \cap D_q^*|}{|D_q^*|} \\ \underset{\text{micro}}{rec}(\mu, q) &= \frac{\sum_{\langle d, f(d)\rangle \in (\pi_q \cap \pi_q^*)} f(d)}{\sum_{\langle d, f(d)\rangle \in \pi_q^*} f(d)} \end{aligned}$$

As described above, D_q and π_q are the results retrieved from μ for query q under a top-k strategy, whereas D_q^* and π_q^* are the results obtained when evaluating μ's postlists completely. While rec_{macro} considers only result list lengths, rec_{micro} allots more weight to n-grams with high occurrence frequencies, since they are more relevant to the user. Figure 7.2 shows the results for different query sizes.

The macro-averaged recall differs significantly from the micro-averaged recall, which indicates that most of the relevant n-grams are retrieved with our strategy. The current NETSPEAK quantile of $\kappa = 0.5$ marks the best trade-off between recall and retrieval time. At quantile 0.5 only 1.47% of a postlist is evaluated on average, which translates into a retrieval speedup of factor 68. The average retrieval time at this quantile seems to leave much room in terms of user patience to evaluate more of a postlist, however, it does not include the time to generate and ship the result page. Short queries are more difficult to answer because the size of the expected result set is much larger on average than that of a long query. From an evaluation standpoint the micro-averaged view appears to be more expressive. Altogether, our retrieval strategy makes NETSPEAK a fast and reliable writing assistant.

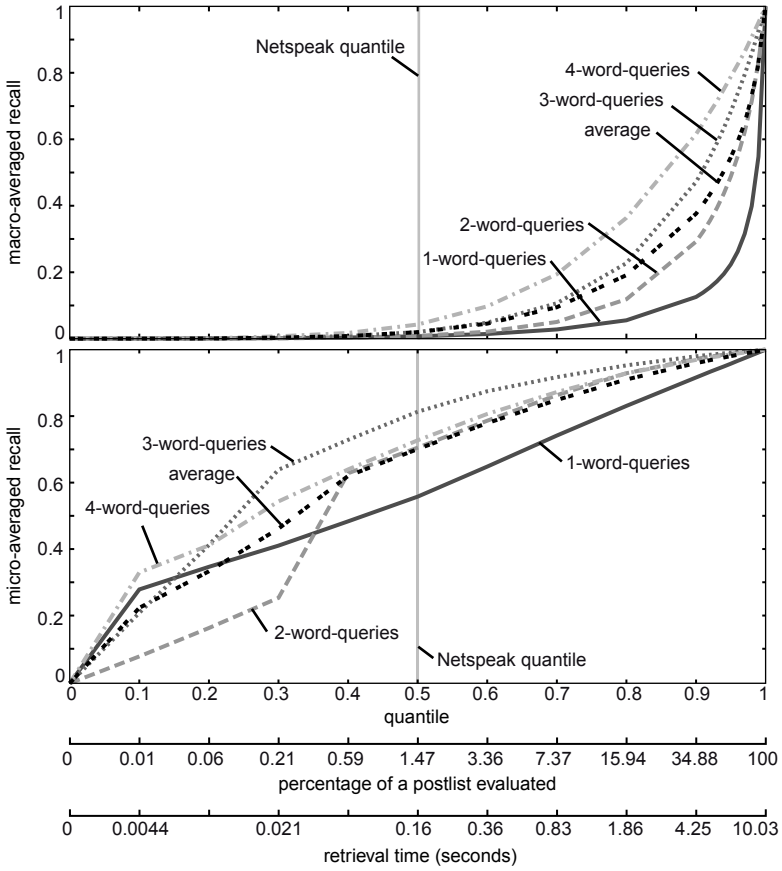


Figure 7.2: Macro-averaged recall (left) and micro-averaged recall (right) over quantiles. The additional axes indicate how much of a postlist is evaluated and the required processing time.

7.3 Visualizing Wildcard Search Results

NETSPEAK can be accessed via a classic web interface which presents search results as a list of *n*-grams ordered by their occurrence frequencies. Although straightforward and simple, for certain queries, this interface has its limitations with respect to showing the true picture of which choice of words is most common. These shortcomings can be addressed by means of a graph visualization of the search results: the NETSPEAK WORDGRAPH.[3] It turns out, however, that neither the web interface nor the graph visualization is perfect. In what follows, the problems associated with both the textual web interface and the WORDGRAPH are detailed.

7.3.1 The Problem of Interdependent Wildcards

The query language of NETSPEAK is powerful in that it allows to specify rather complex patterns of *n*-grams to be retrieved. But the results of a query with more than one wildcard cannot be easily interpreted since the words which are commonly used in place of a given wildcard depend on the words used in place of all other wildcards in the query. This interdependence of wildcards in a query can lead a user to misjudge which phrase is truly common and which isn't. Users who insert more than one wildcard into a query are less confident about how to write a certain phrase and seek to generalize the query in order to cover more of the possible alternatives. This, in turn, yields a longer result lists, which aggravates the difficulty to overviewing them. Figure 7.1 shows an example, where `about` appears in 5 of the *n*-grams, which indicates that this word should most likely follow `rotate`. The web interface, however, obscures this fact and the user is forced to scan the entire result list several times to grasp the true relationships.

[3] The WORDGRAPH is joint research with our colleagues from the Virtual Reality Systems Group at Bauhaus-Universität Weimar. We have published a joint paper about NETSPEAK and the WORDGRAPH visualization which was awarded the Best Paper Award at the 4th IEEE Pacific Visualization Symposium (PacificVis) [183].

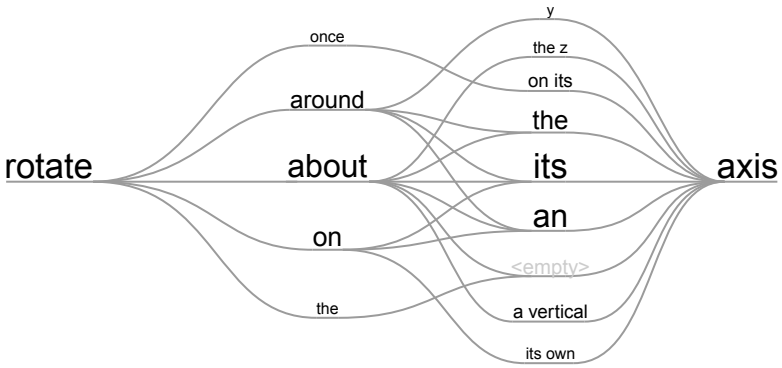


Figure 7.3: The NETSPEAK WORDGRAPH visualization. Visualized are the results for the wildcard query "`rotate ? * axis`" [183].

The NETSPEAK WORDGRAPH helps to overcome this problem. Figure 7.3 shows an example graph. The graph layers follow the structure of a query, showing one layer for every literal word and wildcard, which are filled dynamically with the results obtained from NETSPEAK's retrieval engine. A graph node corresponds to a word found in the result n-grams, and an edge represents the connection between two subsequent words of an n-gram. Consequently, each n-gram of a result set is represented as a path through the graph. The layers of the graph are arranged in vertical columns to facilitate reading, while multiple occurrences of the same word in a column are merged into a single node, and shared path segments of an n-gram are merged into a single edge.

The layout of the graph encodes the occurrence frequencies of the result n-grams. The vertical arrangement of nodes is done using a center spread ordering, which places words in a column, starting from the center with the most frequently appearing word and alternating the placement between above and below. The decrease of font size hints the decrease of occurrence frequency, where the

occurrence frequency of a word in a column is the sum of the occurrence frequencies of all *n*-grams in which it occurs. Edges between words are drawn as Bézier curves, while the ports for connecting edges with a word node are placed at either end of the word's baseline. The word itself is underlined, so that an incoming edge seemingly passes below the word to the other port and further on into outgoing edges. Connecting incoming and outgoing edges by underlining words significantly contributes to the readability of phrase fragments compared to interrupting the edges.[4]

Coming back to the problem outlined above, merging multiple occurrences of the same word within a column into a single node, effectively solves the problem of overviewing the results of queries with more than one wildcard. The WORDGRAPH shows the most likely alternatives for every wildcard at a glance.

7.3.2 The Result Capacity Problem

Merging duplicate words as well as merging shared path segments of *n*-grams bring about a new problem. If more than one literal word and more than one wildcard are interleaved in a query, the paths that can be seen in the above graph visualization may not always correspond to an *n*-gram from the result set. In this case, the graph has an increased result capacity which may lead the onlooker to false conclusions about what to write. However, since merging duplicate words and shared paths in the graph have been found superior in terms of legibility to all other alternative layouts, several interaction techniques were developed to counter this problem.

Figure 7.4 illustrates the result capacity problem. The graph on top visualizes the result set of the query "`? waiting * ~answer`". Note that the the *n*-gram "`without waiting for a response`" was not part of the result set although a path corre-

[4] In [183], several layout variants are compared of which the one described here turned out to be the most legible.

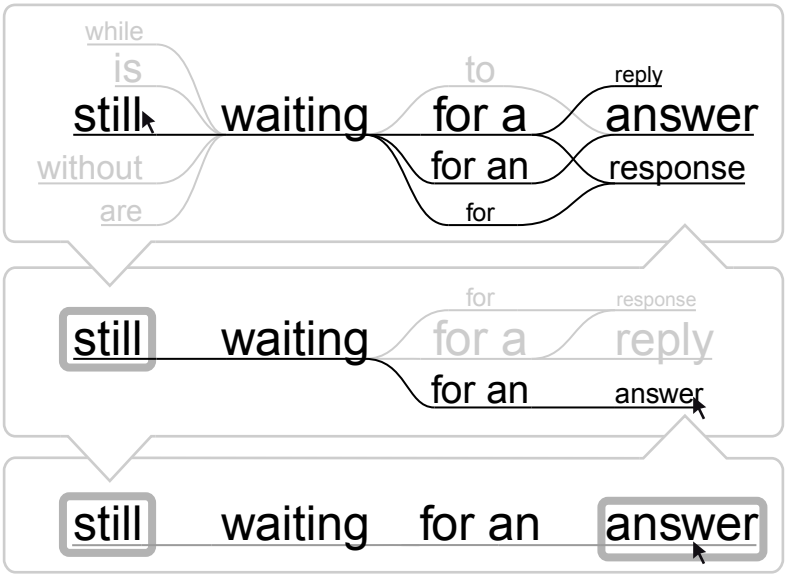


Figure 7.4: The top WORDGRAPH visualizes the results for the query "`? waiting * ~answer`". To counter the result capacity problem, the user may apply filter operations by hovering over a word, and by (repeatedly) selecting a word [183].

sponding to that *n*-gram can be visually traversed. In fact, paths for all combinations of the words found immediately left and right of `waiting` can be visually traversed. To check whether an *n*-gram is actually part of the result set, the user may hover with the mouse pointer over of word in order to highlight all paths passing through the word's node for which *n*-grams exist in the result set. By selecting a word, all other paths fade out. This filtering operation can be applied repeatedly. Deselecting a word word undoes the filter. The filter operations help to solve the result capacity problem, but this solution comes at the price of speed since the user always has to

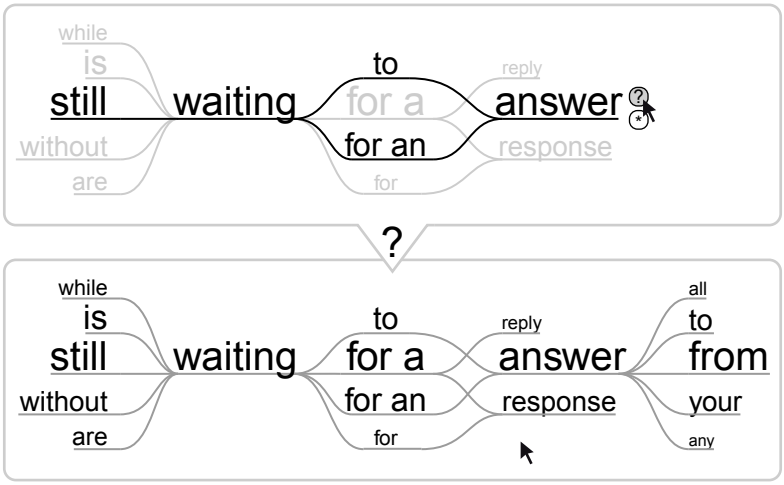


Figure 7.5: Query expansion: by clicking on the wildcard icon next to the word `answer`, a set of queries is generated for all n-grams whose paths pass through this word's node, complemented by the respective wildcard. The n-grams retrieved with these queries are integrated into the graph which results in a new column [183].

check the existence of an n-gram first. Also, this necessity may be difficult to be communicated to the user, since the graph gives no clue that it may show more alternatives than are actually there.

7.3.3 Query Expansion

An analysis of NETSPEAK's query logs shows that many of its users pose not one but many queries in a session in order to refine their search. The WORDGRAPH supports this explorative user behavior by allowing for the visualization of the result sets of multiple queries at the same time. By clicking on the expansion icons shown next to a word while hovering over it, new queries are constructed for

all n-grams whose paths go through the word's node, using up to four preceding words and appending the respective wildcard ? or $\star$. The union of the result sets of all these queries is then integrated into the existing graph structure, adding new columns as needed. The n-grams formed in this way, where $n > 5$, may be incorrect or meaningless since the n-gram collection indexed by NETSPEAK consist of only n-grams up to a length of 5 words; yet, sensible results have been observed in many cases.

7.3.4 Conclusion and Future Work

NETSPEAK answers complex word sequence queries that are formulated in an expressive query language. The system is designed for efficiency and allows for real-time querying of a 42 GB text data base. The result set is explored via a textual web interface or the graphical WORDGRAPH interface. Our analysis shows that neither interface provides for a perfect result visualization, yet. Further research in this direction will be required.

We see NETSPEAK as an educational tool for improving the knowledge of a second language. Additional operators for the query language such as wildcards for parts of speech or semantic constraints (e.g., person names, places, dates and times) and support for further languages besides English would broaden the scope of NETSPEAK. Also, an extension toward specific genres might be worthwhile in order to help inexperienced writers become familiar with the terminology and writing style in a specific field.

Chapter 8

Conclusion

In this thesis we study technologies that deal with reusing texts and language. Our contributions relate toward the automatic detection of text reuse within large document collections as well as technologies that reuse language to support certain retrieval tasks. In what follows, we summarize our contributions and give a brief outlook.

In Chapter 2 we study the application of fingerprinting as a technology for the detection of cases of reused text which are nearly identical to their original sources. We present and evaluate two different types of fingerprint algorithms which specialize on this kind of text reuse. Our findings are that fingerprint algorithms, despite their remarkable runtime properties, are difficult to be employed in practice. Choosing a set of parameters for which a reasonable tradeoff between retrieval precision and retrieval recall is obtained requires a lot of manual probing. Moreover, since hard disk storage is comparably cheap these days, the use case of text fingerprinting may be amiss: major search engines store copies of the whole web and might simply apply the brute force strategy of indexing all n-grams of all texts for a small value of n, which is necessary for the purpose of web search, anyway. Besides these problems, another important insight from our studies is that the low-dimensional

representations built during fingerprint construction are of use by themselves. We found that representations with as little as 50 dimensions achieve a high correlation to the standard vector space model. These low-dimensional representations can be built in linear time in the number of documents.

In Chapter 3 we introduce a new retrieval model for the quantification of the topical similarity of texts across languages, which can be applied for the detection of cross-language text reuse. The most salient property of our model is that it relies on comparable corpora instead of parallel corpora or translation dictionaries. The latter two are difficult to be obtained in practice at the necessary scale, while the former are readily available for a large number of language pairs in the form of Wikipedia. Another important insight from our evaluation is that our baseline model which measures cross-language similarity based on shared character 3-grams between documents is competitive to our model on pairs of languages which have syntactical similarities. Our model, however, also works well on languages without syntactical overlap.

In Chapter 4 we introduce new performance measures for text reuse and plagiarism detectors that balance three different aspects of detection quality, namely precision, recall, and granularity. Moreover, we introduce a new evaluation corpus consisting of automatically generated and manually written plagiarism cases. In this connection, we are the first to study crowdsourcing as means to scale up corpus construction. Both the measures and the corpus have been successfully employed in the course of three evaluation competitions in which 32 plagiarism detectors have been compared.

In Chapter 5 we introduce a new retrieval model for the quantification of the topical similarity of web items across media. Our model reuses web comments as a description of the commented item's topic, thus allowing for the use of standard text retrieval models instead of resorting to low-level feature representations of multimedia items. By analogy, what Wikipedia is to our aforemen-

tioned cross-language text similarity model, web comments are to our cross-media item similarity model. Both form a connection between different document spaces which are otherwise hard to be compared. Another important insight from this chapter is that web comments in general are a neglected source of information in research, with the exception of product reviews. We hence introduce the rationale of comment retrieval, important retrieval tasks therein, and their connections to existing information retrieval research.

In Chapter 6 we demonstrate how the whole web can be reused to solve a retrieval task. We study the problem of segmenting unquoted keyword queries into segments which could reasonably be surrounded with quotes, and we introduce two new algorithms to do so. The hypothesis underlying both algorithms is that quoted segments within a query should exist on the web; otherwise searching for them does not yield any results. To choose among the many alternative quotations for a given keyword query, our algorithms exploit the web in the form of n-grams along with their respective occurrence frequencies, favoring quotations that consist of frequent n-grams. It turns out that our comparably simple algorithms compete well with all state-of-the-art algorithms from the literature. In this connection, we contribute further by constructing a new, large-scale evaluation corpus of manually quoted queries via crowdsourcing.

In Chapter 7 we demonstrate another way of reusing the whole web, this time in order to assist writers. We hypothesize that aside from a text's correctness also its commonness is an important property. Choosing commonly written words to say something over more uncommon alternatives has the advantage of maximizing the number of people within a text's target audience who understand its message. While tools that help with correcting a text are being researched and developed, tools that help with increasing a text's commonness are not. We go a first step in this direction by implementing NETSPEAK, a search engine for commonly used language. NETSPEAK indexes the whole web in the form of n-grams along with

their occurrence frequencies, and it allows for wildcard queries. This way, a user may query short phrases of text while inserting wildcards at positions where he or she is uncertain about what words are most commonly used. Besides maximizing the retrieval speed of NETSPEAK's query processor, there are still open challenges for development: the wildcard query language is rather difficult to understand for laymen users, so that most use only a small subset of the available operators. A more sophisticated user interface should be developed which helps users with learning the query language. Moreover, visualizing the search results of a query turns out to be problematic for some queries with regard to showing a user the true picture about what words are more commonly used than others.

Research on text reuse and language reuse has many facets. While the detection of text reuse is certainly among the more prevalent problems studied, technologies that reuse language offer exciting new challenges and a lot of room for creative solutions.

Outlook

Imagine a tool that reveals all cases of text reuse on the web. What would be the consequences? Plagiarism, the evil twin of text reuse, would become futile. But besides this immediate consequence, such a tool would induce a new kind of network among web documents, different from the hyperlink network, which links a text and its descendants. Such a network could be used, for example, as an additional feature in web search, but also as a means to pass reputation and reward back to the original source of a text. While the latter may not be easily accomplished on the whole web, certain genres of writing could be organized this way, for example, scientific writing.

Bibliography

[1] Navot Akiva. Using Clustering to Identify Outlier Chunks of Text: Notebook for PAN at CLEF 2011. In Petras and Clough [156]. ISBN 978-88-904810-1-7. 91, 92, 95, 100

[2] James Allen. Submission to the 1st International Competition on Plagiarism Detection. http://www.webis.de/research/events/pan-09, 2009. ISSN 1613-0073. URL http://ceur-ws.org/Vol-502. From the Southern Methodist University in Dallas, USA. 95

[3] Omar Alonso and Stefano Mizzaro. Can we get Rid of TREC Assessors? Using Mechanical Turk for Relevance Assessment. In Shlomo Geva, Jaap Kamps, Carol Peters, Tetsuya Sakai, Andrew Trotman, and Ellen Voorhees, editors, *Proceedings of the First SIGIR Workshop on the Future of IR Evaluation*, pages 15–16, Boston, Massachusetts, July 2009. IR Publications. 76, 159

[4] Salha Alzahrani and Naomie Salim. Fuzzy Semantic-Based String Similarity for Extrinsic Plagiarism Detection: Lab Report for PAN at CLEF 2010. In Braschler and Harman [28]. ISBN 978-88-904810-0-0. 95, 98, 99

[5] Vamshi Ambati, Stephan Vogel, and Jaime Carbonell. Active Learning and Crowd-Sourcing for Machine Translation. In

Nicoletta Calzolari, Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, Mike Rosner, and Daniel Tapias, editors, *Proceedings of the Seventh Conference on International Language Resources and Evaluation (LREC'10)*, pages 2169–2174, Valletta, Malta, May 2010. European Language Resources Association. 76

[6] Maik Anderka, Benno Stein, and Martin Potthast. Cross-language High Similarity Search: Why no Sub-linear Time Bound can be Expected. In Cathal Gurrin, Yulan He, Gabriella Kazai, Udo Kruschwitz, Suzanne Little, Thomas Roelleke, Stefan M. Rüger, and Keith van Rijsbergen, editors, *Advances in Information Retrieval. 32nd European Conference on Information Retrieval (ECIR 10)*, volume 5993 of *Lecture Notes in Computer Science*, pages 640–644, Berlin Heidelberg New York, 2010. Springer. ISBN 978-3-642-12274-3. doi:10.1007/978-3-642-12275-0_66. 13, 43, 64

[7] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, 1999. ISBN 0-201-39829-X. 15

[8] Lisa Ann Ballesteros. *Resolving Ambiguity for Cross-Language Information Retrieval: A Dictionary Approach*. PhD thesis, University of Massachusetts Amherst, USA, 2001. Bruce Croft. 49

[9] Jeff Barr and Luis Felipe Cabrera. AI gets a Brain. *Queue*, 4(4): 24–29, May 2006. 76

[10] Alberto Barrón-Cedeño, Paolo Rosso, David Pinto, and Alfons Juan. On Cross-Lingual Plagiarism Analysis Using a Statistical Model. In Benno Stein, Efstathios Stamatatos, and Moshe Koppel, editors, *ECAI 2008 Workshop on Uncovering Plagiarism, Authorship, and Social Software Misuse (PAN 08)*,

pages 9–13, Patras (Greece), July 2008. ISBN 978-960-6843-08-2. 43, 44, 48, 51

[11] Regina Barzilay and Lillian Lee. Learning to Paraphrase: An Unsupervised Approach Using Multiple-Sequence Alignment. In Eduard Hovy, Marti Hearst, and Mari Ostendorf, editors, *Proceedings of the Third Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 16–23, Edmonton, Canada, May 2003. Association for Computational Linguistics. 80

[12] Chiara Basile, Dario Benedetto, Emanuele Caglioti, Giampaolo Cristadoro, and Mirko Degli Esposti. A Plagiarism Detection Procedure in Three Steps: Selection, Matches and "Squares". In Stein et al. [213], pages 19–23. URL http://ceur-ws.org/Vol-502. 95

[13] Holger Bast, Debapriyo Majumdar, Ralf Schenkel, Martin Theobald, and Gerhard Weikum. IO-Top-k: Index-Access Optimized Top-k Query Processing. In *VLDB'06: Proceedings of the 32nd international conference on Very large data bases*, pages 475–486. VLDB Endowment, 2006. 175

[14] Leonard E. Baum. An Inequality and Associated Maximization Technique in Statistical Estimation of Probabilistic Functions of a Markov Process. *Inequalities*, 3: 1–8, 1972. 52

[15] Mayank Bawa, Tyson Condie, and Prasanna Ganesan. LSH Forest: Self-Tuning Indexes for Similarity Search. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 651–660, New York, NY, USA, 2005. ACM Press. ISBN 1-59593-046-9. 35

[16] Philip Beineke, Trevor Hastie, Christopher Manning, and Shivakumar Vaithyanathan. An Exploration of Sentiment Summarization. In *Proceedings of AAAI 2003*, pages 12–15, 2003. 112

[17] D. Belazzougui, F.C. Botelho, and M. Dietzfelbinger. Hash, Displace, and Compress. In *ESA'09: Proceedings of the 17th European Symposium on Algorithms*, pages 682–693, Springer Berlin / Heidelberg, 2009. Springer. ISBN 978-3-642-04127-3. doi:10.1007/978-3-642-04128-0. 173

[18] N. J. Belkin, R. N. Oddy, and H. M. Brooks. ASK for Information Retrieval. *Journal of Documentation*, 33(2):61–71, 1982. 108

[19] Michael Bendersky and W. Bruce Croft. Finding Text Reuse on the Web. In Ricardo A. Baeza-Yates, Paolo Boldi, Berthier A. Ribeiro-Neto, and Berkant Barla Cambazoglu, editors, *Proceedings of the Second International Conference on Web Search and Web Data Mining, WSDM 2009, Barcelona, Spain, February 9-11, 2009*, pages 262–271. ACM, 2009. ISBN 978-1-60558-390-7. doi:10.1145/1498759.1498835. 5

[20] Michael Bendersky, W. Bruce Croft, and David A. Smith. Structural Annotation of Search Queries Using Pseudo-Relevance Feedback. In Jimmy Huang, Nick Koudas, Gareth J. F. Jones, Xindong Wu, Kevyn Collins-Thompson, and Aijun An, editors, *Proceedings of the 19th ACM Conference on Information and Knowledge Management, CIKM 2010, Toronto, Ontario, Canada, October 26-30, 2010*, pages 1537–1540. ACM, 2010. ISBN 978-1-4503-0099-5. 140

[21] Micheal Bendersky, W. Bruce Croft, and David Smith. Two-Stage Query Segmentation for Information Retrieval. In James Allan, Javed A. Aslam, Mark Sanderson, ChengXiang

Zhai, and Justin Zobel, editors, *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2009, Boston, USA, July 20-24, 2009*, pages 810–811. ACM, 2009. ISBN 978-1-60558-483-6. 140

[22] Adam Berger and John Lafferty. Information Retrieval as Statistical Translation. In *SIGIR'99: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 222–229, Berkeley, California, United States, 1999. ACM. 52

[23] Shane Bergsma and Qin Iris Wang. Learning Noun Phrase Query Segmentation. In *Proceedings of Conference on Empirical Methods in Natural Language Processing and Conference on Computational Natural Language Learning, EMNLP-CoNLL 2007, June 28-30, 2007, Prague, Czech Republic*, pages 819–826. Association for Computational Linguistics, 2007. 139, 151, 152, 155, 156, 157, 158, 161

[24] Y. Bernstein and J. Zobel. A Scalable System for Identifying Co-derivative Documents. In A. Apostolico and M. Melucci, editors, *Proceedings of the String Processing and Information Retrieval Symposium (SPIRE)*, pages 55–67, Padova, Italy, September 2004. Springer. Published as LNCS 3246. 32

[25] Joshua E. Blumenstock. Size Matters: Word Count as a Measure of Quality on Wikipedia. In *WWW'08: Proceeding of the 17th international conference on World Wide Web*, pages 1095–1096. ACM, 2008. 118, 121

[26] Thorsten Brants and Alex Franz. Web 1T 5-gram Version 1. Linguistic Data Consortium LDC2006T13, Philadelphia, 2006. 8, 142, 172

[27] Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. Large Language Models in Machine Translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 858–867, Prague, Czech Republic, June 2007. Association for Computational Linguistics. 165

[28] Martin Braschler and Donna Harman, editors. *Notebook Papers of CLEF 2010 Labs and Workshops, 22-23 September, Padua, Italy*, 2010. ISBN 978-88-904810-0-0. URL http://www.webis.de/research/events/pan-10. 188, 196, 200, 201, 205, 210, 211, 212, 213, 220, 222, 225, 229

[29] David J. Brenes, Daniel Gayo-Avello, and Rodrigo Garcia. On the Fly Query Entity Decomposition Using Snippets. In *1st Spanish Conference on Information Retrieval (CERI 2010), Proceedings, June 15-16, 2010, Madrid, Spain*, 2010. 139, 140, 155, 156, 157, 158, 161

[30] Sergey Brin, James Davis, and Hector Garcia-Molina. Copy Detection Mechanisms for Digital Documents. In *SIGMOD'95*, pages 398–409, New York, NY, USA, 1995. ACM Press. ISBN 0-89791-731-6. 32

[31] Chris Brockett, William B. Dolan, and Michael Gamon. Correcting ESL Errors Using Phrasal SMT Techniques. In *ACL'06: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 249–256, Morristown, NJ, USA, 2006. Association for Computational Linguistics. doi:10.3115/1220175.1220207. 171

[32] Andrei Z. Broder. Identifying and Filtering Near-Duplicate Documents. In *COM'00: Proceedings of the 11th Annual*

Symposium on Combinatorial Pattern Matching, pages 1–10, London, UK, 2000. Springer-Verlag. ISBN 3-540-67633-3. 30, 32, 85, 90

[33] Andrei Z. Broder, Nadav Eiron, Marcus Fontoura, Michael Herscovici, Ronny Lempel, John McPherson, Runping Qi, and Eugene J. Shekita. Indexing Shared Content in Information Retrieval Systems. In *EDBT '06*, pages 313–330, 2006. 25

[34] Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, 19(2):263–311, 1993. 52

[35] Steven Burrows, Martin Potthast, and Benno Stein. Paraphrase Acquisition via Crowdsourcing and Machine Learning. *Transactions on Intelligent Systems and Technology (ACM TIST) (to appear)*, 2012. doi:http://dx.doi.org/. 13, 107

[36] Jaime Carbonell and Jade Goldstein. The Use of MMR, Diversity-based Reranking for Reordering Documents and Producing Summaries. In *SIGIR'98: Proceedings of the 21st annual international ACM SIGIR conference on research and development in information retrieval*, pages 335–336, New York, NY, USA, 1998. ACM. ISBN 1-58113-015-5. doi:10.1145/290941.291025. 122

[37] Manuel Cebrian, Manuel Alfonseca, and Alfonso Ortega. Towards the Validation of Plagiarism Detection Tools by Means of Grammar Evolution. *IEEE Transactions on Evolutionary Computation*, 13(3):477–485, June 2009. ISSN 1089-778X. 76

[38] Zdenek Ceska, Michal Toman, and Karel Jezek. Multilingual Plagiarism Detection. In *AIMSA'08: Proceedings of the 13th*

international conference on Artificial Intelligence, pages 83–92, Berlin, Heidelberg, 2008. Springer-Verlag. ISBN 978-3-540-85775-4. doi:10.1007/978-3-540-85776-1_8. 43

[39] J.S. Chall and E. Dale. *Readability Revisited: The new Dale-Chall Readability Formula*. Brookline Books, 1995. 118

[40] Moses S. Charikar. Similarity Estimation Techniques from Rounding Algorithms. In *STOC '02: Proceedings of the thirty-fourth annual ACM symposium on Theory of computing*, pages 380–388, New York, NY, USA, 2002. ACM Press. ISBN 1-58113-495-9. 35

[41] Abdur Chowdhury, Ophir Frieder, David Grossman, and Mary Catherine McCabe. Collection Statistics for Fast Duplicate Document Detection. *ACM Trans. Inf. Syst.*, 20(2): 171–191, 2002. ISSN 1046-8188. 32

[42] Paul Clough. Plagiarism in Natural and Programming Languages: An Overview of Current Tools and Technologies. Internal Report CS-00-05, University of Sheffield, 2000. 70

[43] Paul Clough. *Measuring Text Reuse*. PhD thesis, University of Sheffield, 2003. 5

[44] Paul Clough. Old and New Challenges in Automatic Plagiarism Detection. National UK Plagiarism Advisory Service, `http://ir.shef.ac.uk/cloughie/papers/pas_plagiarism.pdf`, 2003. 70

[45] Paul Clough and Mark Stevenson. Creating a Corpus of Plagiarised Academic Texts. In *Proceedings of Corpus Linguistics Conference, CL'09 (to appear)*, 2009. 71

[46] Paul Clough and Mark Stevenson. Developing a Corpus of Plagiarised Short Answers. *Lang. Resour. Eval.*, 45:5–24, March 2011. ISSN 1574-020X. doi:10.1007/s10579-009-9112-1. 71

[47] Paul Clough, Robert Gaizauskas, and S. L. Piao. Building and Annotating a Corpus for the Study of Journalistic Text Reuse. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC-02)*, pages 1678–1691, 2002. 71, 76

[48] Paul Clough, Robert Gaizauskas, Scott S. L. Piao, and Yorick Wilks. METER: MEasuring TExt Reuse. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 152–159, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics. doi:10.3115/1073083.1073110. 5

[49] Jack G. Conrad and Cindy P. Schriber. Constructing a Text Corpus for Inexact Duplicate Detection. In *SIGIR'04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 582–583, New York, NY, USA, 2004. ACM Press. ISBN 1-58113-881-4. 37

[50] Jack G. Conrad, Xi S. Guo, and Cindy P. Schriber. Online Duplicate Document Detection: Signature Reliability in a Dynamic Retrieval Environment. In *Proceedings of the 2003 ACM CIKM International Conference on Information and Knowledge Management (CIKM)*, pages 443–452. ACM, 2003. 32

[51] Neil Cooke, Lee Gillam, Henry Cooke Peter Wrobel, and Fahad Al-Obaidli. A High-performance Plagiarism Detection System: Notebook for PAN at CLEF 2011. In Petras and Clough [156]. ISBN 978-88-904810-1-7. 95, 100

[52] Marta R. Costa-jussá, Rafael Banchs, Jens Grivolla, and Joan Codina. Plagiarism Detection Using Information Retrieval and Similarity Measures based on Image Processing techniques: Lab Report for PAN at CLEF 2010. In Braschler and Harman [28]. ISBN 978-88-904810-0-0. 95, 98, 99

[53] Bruce Croft, Donald Metzler, and Trevor Strohman. *Search Engines: Information Retrieval in Practice*. Addison-Wesley, USA, 2009. 15

[54] W. Bruce Croft, Michael Bendersky, Hang Li, and Gu Xu. Query Representation and Understanding Workshop. *SIGIR Forum*, 44(2):48–53, 2010. 138

[55] Sally Jo Cunningham and David M. Nichols. How People Find Videos. In *JCDL'08: Proceedings of the 8th ACM/IEEE-CS joint conference on Digital libraries*, pages 201–210, New York, NY, USA, 2008. ACM. ISBN 978-1-59593-998-2. doi:10.1145/1378889.1378924. 114

[56] E. Dale and J.S. Chall. A Formula for Predicting Readability. *Educational Research Bulletin*, 27, 1948. 118

[57] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni. Locality-Sensitive Hashing Scheme Based on p-Stable Distributions. In *SCG '04: Proceedings of the twentieth annual symposium on Computational geometry*, pages 253–262, New York, NY, USA, 2004. ACM Press. ISBN 1-58113-885-7. 35

[58] Jean-Yves Delort. Identifying Commented Passages of Documents Using Implicit Hyperlinks. In *HYPERTEXT'06: Proceedings of the seventeenth conference on Hypertext and hypermedia*, pages 89–98, New York, NY, USA, 2006. ACM. ISBN 1-59593-417-0. doi:10.1145/1149941.1149960. 114

[59] Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977. 52

[60] Koen Deschacht and Marie-Francine Moens. Finding the Best Picture: Cross-Media Retrieval of Content. In Craig Macdonald, Iadh Ounis, Vassilis Plachouras, Ian Ruthven, and Ryen W. White, editors, *30th European Conference on IR Research, ECIR 2008, Glasgow*, volume 4956 LNCS of *Lecture Notes in Computer Science*, pages 539–546, Berlin Heidelberg New York, 2008. Springer. ISBN 978-3-540-78645-0. URL 10.1007/978-3-540-78646-7_51. 129

[61] B. Dit, D. Poshyvanyk, and A. Marcus. Measuring the Semantic Similarity of Comments in Bug Reports. In *Proceedings of 1st International ICPC2008 Workshop on Semantic Technologies in System Maintenance (STSM2008)*, 2008. 113

[62] Susan T. Dumais, Todd A. Letsche, Michael L. Littman, and Thomas K. Landauer. Automatic Cross-Language Retrieval Using Latent Semantic Indexing. In D. Hull and D. Oard, editors, *AAAI'97 Spring Symposium Series: Cross-Language Text and Speech Retrieval*, pages 18–24, Stanford University, March 1997. American Association for Artificial Intelligence. 48

[63] Dennis Fetterly, Mark Manasse, and Marc Najork. On the Evolution of Clusters of Near-Duplicate Web Pages. In *Proceedings of the 1st Latin American Web Congress, LA-WEB 2003*. IEEE, 2003. ISBN 0-7695-2058-8/03. 25, 32

[64] R. Flesch. A New Readability Yardstick. *Journal of Applied Psychology*, 32:221–233, 1948. 118

[65] Atsushi Fujii and Tetsuya Ishikawa. A System for Summarizing and Visualizing Arguments in Subjective Documents: Toward Supporting Decision Making. In *Proceedings of the Workshop on Sentiment and Subjectivity in Text*, pages 15–22, Sydney, Australia, July 2006. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W06/W06-0303. 112

[66] E. Gabrilovich. *Feature Generation for Textual Information Retrieval Using World Knowledge*. Phd thesis, Israel Institute of Technology, 2006. 123

[67] Evgeniy Gabrilovich and Shaul Markovitch. Computing Semantic Relatedness of Words and Texts in Wikipedia-derived Semantic Space. Technical report cis-2006-04, Computer Science Department, Technion, Haifa, Israel, 2006. 123

[68] Evgeniy Gabrilovich and Shaul Markovitch. Computing Semantic Relatedness Using Wikipedia-based Explicit Semantic Analysis. In Manuela M. Veloso, editor, *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, pages 1606–1611, 2007. 50, 123, 129

[69] Anindya Ghose and Panagiotis G. Ipeirotis. Designing Ranking Systems for Consumer Reviews: The Impact of Review Subjectivity on Product Sales and Review Quality. In *Proceedings of the 2007 9th International Conference on Decision Support Systems (ICDSS 2007)*, Kolkata, India, January 2007. 112

[70] Aniruddha Ghosh, Pinaki Bhaskar, Santanu Pal, and Sivaji Bandyopadhyay. Rule Based Plagiarism Detection using Information Retrieval: Notebook for PAN at CLEF 2011. In Petras and Clough [156]. ISBN 978-88-904810-1-7. 95, 100

[71] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Similarity Search in High Dimensions via Hashing. In *Proceedings of the 25th VLDB Conference Edinburgh, Scotland*, pages 518–529, 1999. 36

[72] Thomas Gottron. External Plagiarism Detection Based on Standard IR Technology: Lab Report for PAN at CLEF 2010.

In Braschler and Harman [28]. ISBN 978-88-904810-0-0. 95, 98, 99

[73] Ján Grman and Rudolf Ravas. Improved Implementation for Finding Text Similarities in Large Collections of Data: Notebook for PAN at CLEF 2011. In Petras and Clough [156]. ISBN 978-88-904810-1-7. 95, 96, 100, 101

[74] Cristian Grozea and Marius Popescu. Encoplot—Performance in the Second International Plagiarism Detection Challenge: Lab Report for PAN at CLEF 2010. In Braschler and Harman [28]. ISBN 978-88-904810-0-0. 95, 96, 98, 99, 101, 102

[75] Cristian Grozea and Marius Popescu. The Encoplot Similarity Measure for Automatic Detection of Plagiarism: Notebook for PAN at CLEF 2011. In Petras and Clough [156]. ISBN 978-88-904810-1-7. 95, 96, 100

[76] Cristian Grozea, Christian Gehl, and Marius Popescu. ENCOPLOT: Pairwise Sequence Matching in Linear Time Applied to Plagiarism Detection. In Stein et al. [213], pages 10–18. URL http://ceur-ws.org/Vol-502. 94, 95

[77] Robert Gunning. The Fog Index After Twenty Years. *Journal of Business Communication*, 6(2):3–13, 1969. 118

[78] Jiafeng Guo, Gu Xu, Hang Li, and Xueqi Cheng. A Unified and Discriminative Model for Query Refinement. In Sung-Hyon Myaeng, Douglas W. Oard, Fabrizio Sebastiani, Tat-Seng Chua, and Mun-Kew Leong, editors, *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2008, Singapore, July 20-24, 2008*, pages 379–386. ACM, 2008. ISBN 978-1-60558-164-4. 138

[79] Parth Gupta and Sameer Rao. External Plagiarism Detection: N-Gram Approach using Named Entity Recognizer: Lab Report for PAN at CLEF 2010. In Braschler and Harman [28]. ISBN 978-88-904810-0-0. 95, 98, 99

[80] Barak Hagbi and Moshe Koppel. Submission to the 1st International Competition on Plagiarism Detection. http://www.webis.de/research/events/pan-09, 2009. ISSN 1613-0073. URL http://ceur-ws.org/Vol-502. From the Bar Ilan University, Israel. 94, 95

[81] Matthias Hagen, Martin Potthast, Benno Stein, and Christof Bräutigam. The Power of Naïve Query Segmentation. In Fabio Crestani, Stéphane Marchand-Maillet, Hsin-Hsi Chen, Efthimis N. Efthimiadis, and Jacques Savoy, editors, *33rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 10)*, pages 797–798. ACM, July 2010. ISBN 978-1-4503-0153-4. doi:10.1145/1835449.1835621. 13, 138, 139, 140, 141, 144, 155, 156, 157, 161, 162

[82] Matthias Hagen, Martin Potthast, Benno Stein, and Christof Bräutigam. Query Segmentation Revisited. In Sadagopan Srinivasan, Krithi Ramamritham, Arun Kumar, M. P. Ravindra, Elisa Bertino, and Ravi Kumar, editors, *20th International Conference on World Wide Web (WWW 11)*, pages 97–106. ACM, March 2011. doi:10.1145/1963405.1963423. 13, 138, 141, 156, 157, 161

[83] Nevin Heintze. Scalable Document Fingerprinting. In *Proceedings of the Second USENIX Electronic Commerce Workshop*, pages 191–200, 1996. 32

[84] Monika Henzinger. Finding Near-Duplicate Web Pages: A Large-Scale Evaluation of Algorithms. In *SIGIR'06:*

Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval, pages 284–291, New York, NY, USA, 2006. ACM Press. ISBN 1-59593-369-7. doi:10.1145/1148170.1148222. 37

[85] Timothy C. Hoad and Justin Zobel. Methods for Identifying Versioned and Plagiarised Documents. *American Society for Information Science and Technology*, 54(3):203–215, 2003. 37

[86] W. T. B. Hordijk, M. L. Ponisio, and R. J. Wieringa. Structured Review of Code Clone Literature. Technical Report TR-CTIT-08-33, Centre for Telematics and Information Technology, University of Twente, Enschede, 2008. 70

[87] C.-F. Hsu, E. Khabiri, and J. Caverlee. Ranking Comments on the Social Web. In *IEEE International Conference on Social Computing (SocialCom)*, 2009. 112

[88] Meishan Hu, Aixin Sun, and Ee-Peng Lim. Comments-Oriented Blog Summarization by Sentence Extraction. In *CIKM'07: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 901–904, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-803-9. doi:10.1145/1321440.1321571. 114

[89] Meishan Hu, Aixin Sun, and Ee-Peng Lim. Comments-Oriented Document Summarization: Understanding Documents with Readers' Feedback. In *SIGIR'08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 291–298, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-164-4. doi:10.1145/1390334.1390385. 114

[90] Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *KDD'04: Proceedings of the tenth ACM*

SIGKDD international conference on Knowledge discovery and data mining, pages 168–177, New York, NY, USA, 2004. ACM. ISBN 1-58113-888-1. doi:10.1145/1014052.1014073. 112

[91] Jian Huang, Jianfeng Gao, Jiangbo Miao, Xiaolong Li, Kuansan Wang, and Fritz Behr. Exploring Web Scale Language Models for Search Query Processing. In Rappa et al. [180], pages 451–460. ISBN 978-1-60558-799-8. 139, 154

[92] Shen Huang, Dan Shen, Wei Feng, Yongzheng Zhang, and Catherine Baudin. Discovering Clues for Review Quality from Author's Behaviors on E-Commerce sites. In *ICEC'09: Proceedings of the 11th International Conference on Electronic Commerce*, pages 133–141, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-586-4. doi:10.1145/1593254.1593274. 112

[93] Adrian Iftene. Submission to the 1st International Competition on Wikipedia Vandalism Detection, 2010. URL `http://www.webis.de/research/events/pan-10`. From the Universtiy of Iasi, Romania. 95, 98, 99

[94] Ihab F. Ilyas, George Beskales, and Mohamed A. Soliman. A Survey of Top-k Query Processing Techniques in Relational Database Systems. *ACM Comput. Surv.*, 40(4):1–58, 2008. ISSN 0360-0300. doi:10.1145/1391729.1391730. 175

[95] P. Indyk. Stable Distributions, Pseudorandom Generators, Embeddings and data Stream Computation. In *FOCS'00: Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, page 189, Washington, DC, USA, 2000. IEEE Computer Society. ISBN 0-7695-0850-2. 36

[96] Piotr Indyk and Rajeev Motwani. Approximate Nearest Neighbor—Towards Removing the Curse of Dimensionality. In *Proceedings of the 30th Symposium on Theory of Computing*, pages 604–613, 1998. 32, 35

[97] Peter Ingwersen and Kalervo Järvelin. *The Turn: Integration of Information Seeking and Retrieval in Context (The Information Retrieval Series)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005. 14

[98] Masashi Inoue. On the Need for Annotation-based Image Retrieval. In *Proceedings of the SIGIR'04 Workshop Information Retrieval in Context*, pages 44–46, 2004. 129

[99] Salman Jamali and Huzefa Rangwala. Digging Digg: Comment Mining, Popularity Prediction, and Social Network Analysis. Technical report gmu-cs-tr-2009-7, George Mason University, USA, 2009. 115

[100] Nitin Jindal and Bing Liu. Analyzing and Detecting Review Spam. In *Proceedings of the 7th IEEE International Conference on Data Mining (ICDM 2007), October 28-31, 2007*, pages 547–552. IEEE, 2007. doi:10.1109/ICDM.2007.68. 112

[101] Nitin Jindal and Bing Liu. Opinion spam and analysis. In *WSDM '08: Proceedings of the international conference on Web search and web data mining*, pages 219–230, New York, NY, USA, 2008. ACM. doi:10.1145/1341531.1341560. 112

[102] Rosie Jones, Benjamin Rey, Omid Madani, and Wiley Greiner. Generating Query Substitutions. In Les Carr, David De Roure, Arun Iyengar, Carole A. Goble, and Michael Dahlin, editors, *Proceedings of the 15th international conference on World Wide Web, WWW 2006, Edinburgh, Scotland, UK, May 23-26, 2006*, pages 387–396. ACM, 2006. ISBN 1-59593-323-9. 139

[103] Andreas Kaltenbrunner, Vicenç Gómez, and Vicente López. Description and Prediction of Slashdot Activity. In *Latin American Web Conference (LA-WEB 2007)*, pages 57–66, Los Alamitos, CA, USA, 2007. IEEE Computer Society. ISBN 0-7695-3008-7. doi:10.1109/LA-Web.2007.21. 115

[104] Jan Kasprzak and Michal Brandejs. Improving the Reliability of the Plagiarism Detection System: Lab Report for PAN at CLEF 2010. In Braschler and Harman [28]. ISBN 978-88-904810-0-0. 92, 94, 95, 96, 98, 99

[105] Jan Kasprzak, Michal Brandejs, and Miroslav Křipač. Finding Plagiarism by Evaluating Document Similarities. In Stein et al. [213], pages 24–28. URL http://ceur-ws.org/Vol-502. 95

[106] Mike Kestemont, Kim Luyckx, and Walter Daelemans. Intrinsic Plagiarism Detection Using Character Trigram Distance Scores: Notebook for PAN at CLEF 2011. In Petras and Clough [156]. ISBN 978-88-904810-1-7. 91, 92, 95, 96, 100

[107] Elham Khabiri, Chiao-Fang Hsu, and James Caverlee. Analyzing and Predicting Community Preference of Socially Generated Metadata: A Case Study on Comments in the Digg Community. In *International AAAI Conference on Weblogs and Social Media*, 2009. URL http://aaai.org/ocs/index.php/ICWSM/09/paper/view/177/497. 112

[108] S.-M. Kim, P. Pantel, T. Chklovski, and M. Penneacchiotti. Automatically Assessing Review Helpfulness. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 423–430, Sydney, Australia, July 2006. 112

[109] J. Kincaid, R. P. Fishburne, R. L. Rogers, and B. S. Chissom. Derivation of New Readability Formulas (Automated Readability Index, Fog Count and Flesch Reading Ease Formula) for Navy Enlisted Personnel. Research Branch Report 8-75 Millington TN: Naval Technical Training US Naval Air Station, 1975. 118

[110] Julia Kiseleva, Qi Guo, Eugene Agichtein, Daniel Billsus, and Wei Chai. Unsupervised Query Segmentation Using Click Data: Preliminary Results. In Rappa et al. [180], pages 1131–1132. ISBN 978-1-60558-799-8. 138

[111] Aleksander Kołcz, Abdur Chowdhury, and Joshua Alspector. Improved Robustness of Signature-based Near-Replica Detection via Lexicon Randomization. In *KDD'04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 605–610, New York, NY, USA, 2004. ACM Press. ISBN 1-58113-888-1. 32

[112] Namhee Kwon, Stuart W. Shulman, and Eduard Hovy. Multidimensional Text Analysis for eRulemaking. In *dg.o'06: Proceedings of the 2006 international conference on Digital government research*, pages 157–166, New York, NY, USA, 2006. ACM. doi:http://doi.acm.org/10.1145/1146598.1146649. 112

[113] Namhee Kwon, Liang Zhou, Eduard Hovy, and Stuart W. Shulman. Identifying and Classifying Subjective Claims. In *dg.o'07: Proceedings of the 8th annual international conference on Digital government research*, pages 76–81. Digital Government Society of North America, 2007. ISBN 1-59593-599-1. 112

[114] Andrew Lacey. A Simple Probabilistic Approach to Ranking Documents by Sentiment. In *Proceedings of the Class of 2005 Senior Conference on Natural Language Processing*, pages 1–7, Swarthmore, Pennsylvania, USA, April 2005. Computer Science Department, Swarthmore College. 112

[115] Kevin Lerman and Ryan McDonald. Contrastive Summarization: An Experiment with Consumer Reviews. In *NAACL'09: Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume:*

Short Papers, pages 113–116, Morristown, NJ, USA, 2009. Association for Computational Linguistics. 112

[116] Kevin Lerman, Sasha Blair-Goldensohn, and Ryan McDonald. Sentiment Summarization: Evaluating and Learning User Preferences. In *EACL'09: Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 514–522, Morristown, NJ, USA, 2009. Association for Computational Linguistics. 112

[117] Gina-Anne Levow, Douglas W. Oard, and Philip Resnik. Dictionary-based Techniques for Cross-Language Information Retrieval. *Inf. Process. Manage.*, 41(3):523–547, 2005. ISSN 0306-4573. doi:10.1016/j.ipm.2004.06.012. 48

[118] Michael S. Lew, Nicu Sebe, Chabane Djeraba, and Ramesh Jain. Content-based Multimedia Information Retrieval: State of the Art and Challenges. *ACM Trans. Multimedia Comput. Commun. Appl.*, 2(1):1–19, February 2006. ISSN 1551-6857. doi:10.1145/1126004.1126005. 129

[119] Michael Littman, Susan T. Dumais, and Thomas K. Landauer. Automatic Cross-Language Information Retrieval using Latent Semantic Indexing. In *Cross-Language Information Retrieval, chapter 5*, pages 51–62. Kluwer Academic Publishers, 1998. 48

[120] Bing Liu, Minqing Hu, and Junsheng Cheng. Opinion Observer: analyzing and Comparing Opinions on the Web. In *WWW'05: Proceedings of the 14th international conference on World Wide Web*, pages 342–351, New York, NY, USA, 2005. ACM. ISBN 1-59593-046-9. doi:10.1145/1060745.1060797. 112

[121] Jingjing Liu, Yunbo Cao, Chin-Yew Lin, Yalou Huang, and Ming Zhou. Low-Quality Product Review Detection in

Opinion Summarization. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 334–342, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/D/D07/D07-1035. 112

[122] Yang Liu, Xiangji Huang, Aijun An, and Xiaohui Yu. Reviews Are Not Equally Important: Predicting the Helpfulness of Online Reviews. Technical report cse-2008-05, York University, 2008. 112

[123] Fabian Loose, Steffen Becker, Martin Potthast, and Benno Stein. Retrieval-Technologien für die Plagiaterkennung in Programmen. In Joachim Baumeister and Martin Atzmüller, editors, *Information Retrieval Workshop at LWA 08*, Technical Report 448, pages 5–12. University of Würzburg, Germany, October 2008. 26

[124] Yue Lu, ChengXiang Zhai, and Neel Sundaresan. Rated Aspect Summarization of Short Comments. In *18th International World Wide Web Conference*, pages 131–131, April 2009. URL http://www2009.eprints.org/14/. 112

[125] James A. Malcolm and Peter C. R. Lane. Tackling the PAN'09 External Plagiarism Detection Corpus with a Desktop Plagiarism Detector. In Stein et al. [213], pages 29–33. URL http://ceur-ws.org/Vol-502. 95

[126] U. Manber. Finding Similar Files in a Large File System. In *Proceedings of the USENIX Winter 1994 Technical Conference*, pages 1–10, San Fransisco, CA, USA, 1994. 32

[127] Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts, 1999. 171

[128] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schtüze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008. ISBN 0521865719, 9780521865715. 15

[129] Winter Mason and Duncan J. Watts. Financial Incentives and the "Performance of Crowds". In Paul Bennett, Raman Chandrasekar, Max Chickering, Panos Ipeirotis, Edith Law, Anton Mityagin, Foster Provost, and Luis von Ahn, editors, *Proceedings of the First ACM SIGKDD Workshop on Human Computation*, pages 77–85, Paris, France, June 2009. ACM. 78

[130] Hermann Maurer, Frank Kappe, and Bilal Zaka. Plagiarism – A Survey. *Journal of Universal Computer Science*, 12(8): 1050–1084, August 2006. 70

[131] Anthony M. McEnery and Richard Z. Xiao. Parallel and Comparable Corpora: What are they up to? In Gunilla M. Anderman and Margaret Rogers, editors, *Translating Europe. Incorporating Corpora: The Linguist and the Translator*, pages 18–31. Multilingual Matters, Clevedon, UK, 2007. ISBN 978-1-85359-986-6. 48

[132] Paul Mcnamee and James Mayfield. Character N-Gram Tokenization for European Language Text Retrieval. *Inf. Retr.*, 7(1-2):73–97, 2004. ISSN 1386-4564. doi:10.1023/B:INRT.0000009441.78971.be. 44, 48, 49

[133] Olena Medelyan, David Milne, Catherine Legg, and Ian H. Witten. Mining Meaning from Wikipedia. *Int. J. Hum.-Comput. Stud.*, 67:716–754, September 2009. ISSN 1071-5819. doi:10.1016/j.ijhcs.2009.05.004. 7

[134] Sven Meyer zu Eißen and Benno Stein. Intrinsic Plagiarism Detection. In Mounia Lalmas, Andy MacFarlane, Stefan

Rüger, Anastasios Tombros, Theodora Tsikrika, and Alexei Yavlinsky, editors, *Advances in Information Retrieval. 28th European Conference on IR Research (ECIR 06)*, volume 3936 LNCS of *Lecture Notes in Computer Science*, pages 565–569, Berlin Heidelberg New York, 2006. Springer. ISBN 3-540-33347-9. doi:10.1007/11735106_66. 91, 92

[135] Jean-Baptiste Michel, Yuan K. Shen, Aviva P. Aiden, Adrian Veres, Matthew K. Gray, The Google Books Team, Joseph P. Pickett, Dale Hoiberg, Dan Clancy, Peter Norvig, Jon Orwant, Steven Pinker, Martin A. Nowak, and Erez L. Aiden. Quantitative Analysis of Culture Using Millions of Digitized Books. *Science*, 331(6014):176–182, January 14 2011. doi:10.1126/science.1199644. URL `http://www.isrl.uiuc.edu/~amag/langev/paper/michel2011googleBooksSCIENCE.html`. 7

[136] Daniel Micol, Óscar Ferrández, and Rafael Muñoz. A Textual-Based Similarity Approach for Efficient and Scalable External Plagiarism Analysis: Lab Report for PAN at CLEF 2010. In Braschler and Harman [28]. ISBN 978-88-904810-0-0. 95, 98, 99

[137] Rada Mihalcea and Stephen Pulman. Characterizing Humour: An Exploration of Features in Humorous Texts. In *Proceedings of the Conference on Computational Linguistics and Intelligent Text Processing (CICLing), Springer, Mexico City, February 2007*, 2007. (2nd) best paper award! 121

[138] G. Mishne and N. Glance. Leave a Reply: An Analysis of Weblog Comments. In *Third Annual Workshop on the Weblogging Ecosystem: Aggregation, Analysis and Dynamics (WWW'06)*, May 2006. 107, 110, 114, 115

[139] Gilad Mishne, David Carmel, and Ronny Lempel. Blocking

Blog Spam with Language Model Disagreement. In *AIRWeb*, pages 1–6, 2005. 112

[140] Nikita Mishra, Rishiraj Saha Roy, Niloy Ganguly, Srivatsan Laxman, and Monojit Choudhury. Unsupervised Query Segmentation Using Only Query Logs. In *Proceedings of the 20th International Conference on World Wide Web, WWW 2011, Hyderabad, India, March 28 to April 01, 2011*, 2011. 141, 157, 161

[141] Markus Muhr, Roman Kern, Mario Zechner, and Michael Granitzer. External and Intrinsic Plagiarism Detection using a Cross-Lingual Retrieval and Segmentation System: Lab Report for PAN at CLEF 2010. In Braschler and Harman [28]. ISBN 978-88-904810-0-0. 92, 94, 95, 96, 98, 99, 101, 102

[142] Rao Muhammad Adeel Nawab, Mark Stevenson, and Paul Clough. University of Sheffield: Lab Report for PAN at CLEF 2010. In Braschler and Harman [28]. ISBN 978-88-904810-0-0. 95, 98, 99, 101

[143] Rao Muhammad Adeel Nawab, Mark Stevenson, and Paul Clough. External Plagiarism Detection using Information Retrieval and Sequence Alignment: Notebook for PAN at CLEF 2011. In Petras and Clough [156]. ISBN 978-88-904810-1-7. 95, 100

[144] John P. Nolan. Stable Distributions—Models for Heavy Tailed Data. `http://academic2.american.edu/~jpnolan/stable/stable.html`, 2005. 36

[145] Gabriel Oberreuter, Gaston L'Huillier, Sebastian Rios, and Juan D. Velásquez. FASTDOCODE: Finding Approximated Segments of N-Grams for Document Copy Detection: Lab Report for PAN at CLEF 2010. In Braschler and Harman [28]. ISBN 978-88-904810-0-0. 95, 98, 99, 101

[146] Gabriel Oberreuter, Gaston L'Huillier, Sebastián A. Ríos, and Juan D. Velásquez. Approaches for Intrinsic and External Plagiarism Detection: Notebook for PAN at CLEF 2011. In Petras and Clough [156]. ISBN 978-88-904810-1-7. 91, 92, 93, 95, 96, 100

[147] Franz J. Och and Hermann Ney. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29(1):19–51, 2003. 52

[148] I. Ounis, C. Macdonald, and I. Soboroff. On the TREC Blog Track. In *Proceedings of International Conference on Weblogs and Social Media*, 2008. 114

[149] Yurii Palkovskii, Alexei Belov, and Irina Muzika. Exploring Fingerprinting as External Plagiarism Detection Method: Lab Report for PAN at CLEF 2010. In Braschler and Harman [28]. ISBN 978-88-904810-0-0. 95, 96, 98, 99

[150] Yurii Palkovskii, Alexei Belov, and Iryna Muzyka. Using WordNet-based Semantic Similarity Measurement in External Plagiarism Detection: Notebook for PAN at CLEF 2011. In Petras and Clough [156]. ISBN 978-88-904810-1-7. 95, 100

[151] Yurii Anatol'yevich Palkovskii, Alexei Vitalievich Belov, and Irina Alexandrovna Muzika. Submission to the 1st International Competition on Plagiarism Detection. http://www.webis.de/research/events/pan-09, 2009. ISSN 1613-0073. URL `http://ceur-ws.org/Vol-502`. From the Zhytomyr State University, Ukraine. 95

[152] Jaehui Park, Tomohiro Fukuhara, Ikki Ohmukai, Hideaki Takeda, and Sang-goo Lee. Web Content Summarization Using Social Bookmarks: A New Approach for Social Summarization. In *WIDM'08: Proceeding of the 10th ACM*

workshop on Web information and data management, pages 103–110, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-260-3. doi:10.1145/1458502.1458519. 114

[153] Greg Pass, Abdur Chowdhury, and Cayley Torgeson. A Picture of Search. In Xiaohua Jia, editor, *Proceedings of the 1st International Conference on Scalable Information Systems, Infoscale 2006, Hong Kong, May 30-June 1, 2006*, volume 152 of *ACM International Conference Proceeding Series*, page 1. ACM, 2006. ISBN 1-59593-428-6. 154

[154] Rafael C. Pereira, V. P. Moreira, and R. Galante. Submission to the 1st International Competition on Plagiarism Detection. http://www.webis.de/research/events/pan-09, 2009. ISSN 1613-0073. URL http://ceur-ws.org/Vol-502. From the Universidade Federal do Rio Grande do Sul, Brazil. 95

[155] Rafael C. Pereira, Viviane P. Moreira, and Renata Galante. UFRGSPAN2010: Detecting External Plagiarism: Lab Report for Pan at CLEF 2010. In Braschler and Harman [28]. ISBN 978-88-904810-0-0. 95, 96, 98, 99

[156] Vivien Petras and Paul Clough, editors. *Notebook Papers of CLEF 2011 Labs and Workshops, 19-22 September, Amsterdam, The Netherlands*, 2011. ISBN 978-88-904810-1-7. URL http://www.webis.de/research/events/pan-11. 188, 196, 199, 200, 205, 211, 212, 218, 220

[157] Phil Wolff. Comment on the blog post "A vision for the next generation of blogging tools?" by David Winer. http://web.archive.org/web/20040312054524/http://blogs.law.harvard.edu/bloggerCon/2004/02/24, 2004. A copy of Wolff's comment was preserved in another blog post that summarizes these comments (found at http://www.cadence90.com/wp/?p=2515); the

comment says: *Permalinks in the commentsphere. Cross-posting of comments I post to my side-blog, preserving my blog as the central place to read what I write throughout the web. Notify me when someone comments in my blog. Be specific, better yet: Notify via my choice of email and IM.* 107

[158] David Pinto, Alfons Juan, and Paolo Rosso. Using Query-Relevant Documents Pairs for Cross-Lingual Information Retrieval. In V. Matousek and P. Mautner, editors, *Proceedings of the TSD-2006: Text, Speech and Dialogue*, volume 4629 of *Lecture Notes in Artificial Intelligence*, pages 630–637, Pilsen, Czech Republic, 2007. 52

[159] David Pinto, Jorge Civera, Alberto Barrón-Cedeño, Alfons Juan, and Paolo Rosso. A Statistical Approach to Crosslingual Natural Language Tasks. *J. Algorithms*, 64(1):51–60, 2009. ISSN 0196-6774. doi:10.1016/j.jalgor.2009.02.005. 43, 52

[160] Ana-Maria Popescu and Oren Etzioni. Extracting Product Features and Opinions from Reviews. In *HLT'05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 339–346, Morristown, NJ, USA, 2005. Association for Computational Linguistics. doi:10.3115/1220575.1220618. 112

[161] Martin Potthast. Wikipedia in the Pocket - Indexing Technology for Near-duplicate Detection and High Similarity Search. In Charles Clarke, Norbert Fuhr, Noriko Kando, Wessel Kraaij, and Arjen de Vries, editors, *30th Annual International ACM SIGIR Conference*, pages 909–909. ACM, July 2007. ISBN 978-1-59593-597-7. 26

[162] Martin Potthast. Measuring the Descriptiveness of Web Comments. In Mark Sanderson, ChengXiang Zhai, Justin Zobel, James Allan, and Javed A. Aslam, editors, *32th Annual*

International ACM SIGIR Conference, pages 724–725. ACM, July 2009. ISBN 978-1-60558-483-6. doi:10.1145/1571941.1572097. 13, 107

[163] Martin Potthast. Crowdsourcing a Wikipedia Vandalism Corpus. In Hsin-Hsi Chen, Efthimis N. Efthimiadis, Jaques Savoy, Fabio Crestani, and Stéphane Marchand-Maillet, editors, *33rd Annual International ACM SIGIR Conference*, pages 789–790. ACM, July 2010. ISBN 978-1-4503-0153-4. doi:10.1145/1835449.1835617. 159

[164] Martin Potthast and Steffen Becker. Opinion Summarization of Web Comments. In C. Gurrin et al., editor, *Advances in Information Retrieval, Proceedings of the 32nd European Conference on Information Retrieval, ECIR 2010*, volume 5993 of *Lecture Notes in Computer Science*, pages 668–669, Heidelberg, 2010. Springer. ISBN 978-3-642-12274-3. doi:10.1007/978-3-642-12275-0_73. 107

[165] Martin Potthast and Benno Stein. New Issues in Near-duplicate Detection. In Christine Preisach, Hans Burkhardt, Lars Schmidt-Thieme, and Reinhold Decker, editors, *Data Analysis, Machine Learning and Applications. Selected papers from the 31th Annual Conference of the German Classification Society (GfKl 07)*, Studies in Classification, Data Analysis, and Knowledge Organization, pages 601–609, Berlin Heidelberg New York, 2008. Springer. ISBN 978-3-540-78239-1. doi:10.1007/978-3-540-78246-9_71. 13, 26

[166] Martin Potthast, Benno Stein, and Maik Anderka. A Wikipedia-Based Multilingual Retrieval Model. In Craig Macdonald, Iadh Ounis, Vassilis Plachouras, Ian Ruthven, and Ryen W. White, editors, *Advances in Information Retrieval. 30th European Conference on IR Research (ECIR 08)*, volume 4956 of *Lecture Notes in Computer Science*, pages 522–530,

Berlin Heidelberg New York, 2008. Springer. ISBN 978-3-540-78645-0. doi:10.1007/978-3-540-78646-7_51. 13, 43, 44, 48, 50, 54

[167] Martin Potthast, Benno Stein, and Robert Gerling. Automatic Vandalism Detection in Wikipedia. In Craig Macdonald, Iadh Ounis, Vassilis Plachouras, Ian Ruthven, and Ryen W. White, editors, *Advances in Information Retrieval. 30th European Conference on IR Research (ECIR 08)*, volume 4956 of *Lecture Notes in Computer Science*, pages 663–668, Berlin Heidelberg New York, 2008. Springer. ISBN 978-3-540-78645-0. doi:10.1007/978-3-540-78646-7_75. 118

[168] Martin Potthast, Andreas Eiselt, Benno Stein, Alberto Barrón-Cedeño, and Paolo Rosso. PAN Plagiarism Corpus PAN-PC-09. http://www.webis.de/research/corpora, 2009. 68

[169] Martin Potthast, Benno Stein, Andreas Eiselt, Alberto Barrón-Cedeño, and Paolo Rosso. Overview of the 1st International Competition on Plagiarism Detection. In Benno Stein, Paolo Rosso, Efstathios Stamatatos, Moshe Koppel, and Eneko Agirre, editors, *SEPLN 09 Workshop on Uncovering Plagiarism, Authorship, and Social Software Misuse (PAN 09)*, pages 1–9. CEUR-WS.org, September 2009. URL http://ceur-ws.org/Vol-502. 13, 68, 70, 76, 103

[170] Martin Potthast, Alberto Barrón-Cedeño, Andreas Eiselt, Benno Stein, and Paolo Rosso. Overview of the 2nd International Competition on Plagiarism Detection. In Martin Braschler and Donna Harman, editors, *Notebook Papers of CLEF 10 Labs and Workshops*, September 2010. ISBN 978-88-904810-0-0. 13, 68, 103

[171] Martin Potthast, Benno Stein, Alberto Barrón-Cedeño, and Paolo Rosso. An Evaluation Framework for Plagiarism

Detection. In Chu-Ren Huang and Dan Jurafsky, editors, *23rd International Conference on Computational Linguistics (COLING 10)*, pages 997–1005, Stroudsburg, PA, USA, August 2010. Association for Computational Linguistics. 13, 68, 159

[172] Martin Potthast, Benno Stein, and Steffen Becker. Towards Comment-based Cross-Media Retrieval. In Michael Rappa, Paul Jones, Juliana Freire, and Soumen Chakrabarti, editors, *19th International Conference on World Wide Web (WWW 10)*, pages 1169–1170. ACM, April 2010. ISBN 978-1-60558-799-8. doi:10.1145/1772690.1772858. 13, 107

[173] Martin Potthast, Martin Trenkmann, and Benno Stein. Netspeak: Assisting Writers in Choosing Words. In Cathal Gurrin, Yulan He, Gabriella Kazai, Udo Kruschwitz, Suzanne Little, Thomas Roelleke, Stefan M. Rüger, and Keith van Rijsbergen, editors, *Advances in Information Retrieval. 32nd European Conference on Information Retrieval (ECIR 10)*, volume 5993 of *Lecture Notes in Computer Science*, page 672, Berlin Heidelberg New York, 2010. Springer. ISBN 978-3-642-12274-3. doi:10.1007/978-3-642-12275-0_75. 167

[174] Martin Potthast, Alberto Barrón-Cedeño, Benno Stein, and Paolo Rosso. Cross-Language Plagiarism Detection. *Language Resources and Evaluation (LRE)*, 45:45–62, 2011. ISSN 1574-020X. doi:10.1007/s10579-009-9114-z. 13, 43

[175] Martin Potthast, Andreas Eiselt, Alberto Barrón-Cedeño, Benno Stein, and Paolo Rosso. Overview of the 3rd International Competition on Plagiarism Detection. In Vivien Petras and Paul Clough, editors, *Notebook Papers of CLEF 11 Labs and Workshops*, September 2011. ISBN 978-88-904810-1-7. 13, 68, 103

[176] Bruno Pouliquen, Ralf Steinberger, and Camelia Ignat. Automatic Annotation of Multilingual Text Collections with a

Conceptual Thesaurus. In *Proceedings of the Workshop 'Ontologies and Information Extraction' at the Summer School 'The Semantic Web and Language Technology - Its Potential and Practicalities' (EUROLAN'2003)*, pages 9–28, Bucharest, Romania, August 2003. 48

[177] Bruno Pouliquen, Ralf Steinberger, and Camelia Ignat. Automatic Identification of Document Translations in Large Multilingual Document Collections. In *Proceedings of the International Conference Recent Advances in Natural Language Processing (RANLP'2003)*, pages 401–408, Borovets, Bulgaria, September 2003. 43, 53

[178] Dragomir R. Radev. *Generating Natural Language Summaries from Multiple On-Line Sources: Language Reuse and Regeneration*. PhD thesis, Columbia University, 1999. 7

[179] Sameer Rao, Parth Gupta, Khushboo Singhal, and Prasenjit Majumder. External & Intrinsic Plagiarism Detection: VSM & Discourse Markers based Approach: Notebook for PAN at CLEF 2011. In Petras and Clough [156]. ISBN 978-88-904810-1-7. 91, 92, 95, 100

[180] Michael Rappa, Paul Jones, Juliana Freire, and Soumen Chakrabarti, editors. *Proceedings of the 19th International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, April 26-30, 2010*, 2010. ACM. ISBN 978-1-60558-799-8. 203, 206

[181] Philip Resnik and Aaron Elkiss. The Linguist's Search Engine: An Overview. In *ACL'05: Proceedings of the ACL 2005 on Interactive poster and demonstration sessions*, pages 33–36, Morristown, NJ, USA, 2005. Association for Computational Linguistics. doi:10.3115/1225753.1225762. 171

[182] Antonio Reyes, Martin Potthast, Paolo Rosso, and Benno Stein. Evaluating Humor Features on Web Comments. In Nicoletta Calzolari, Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, Mike Rosner, and Daniel Tapias, editors, *7th Conference on International Language Resources and Evaluation (LREC 10)*. European Language Resources Association (ELRA), May 2010. ISBN 2-9517408-6-7. 107, 121

[183] Patrick Riehmann, Henning Gruendl, Bernd Froehlich, Martin Potthast, Martin Trenkmann, and Benno Stein. The Netspeak WordGraph: Visualizing Keywords in Context. In Giuseppe Di Battista, Jean-Daniel Fekete, and Huamin Qu, editors, *4th IEEE Pacific Visualization Symposium (PacificVis 11)*, pages 123–130. IEEE, March 2011. doi:10.1109/PACIFICVIS.2011.5742381. 13, 167, 178, 179, 180, 181, 182

[184] C. J. van Rijsbergen. *Information Retrieval*. Buttersworth, London, 1979. 15

[185] Knut Magne Risvik, Tomasz Mikolajewski, and Peter Boros. Query Segmentation for Web Search. In *WWW (Posters)*, 2003. 139

[186] Stephen Robertson. Salton Award Lecture on Theoretical Argument in Information Retrieval. *SIGIR Forum*, 34:1–10, April 2000. ISSN 0163-5840. doi:10.1145/373593.373597. 22

[187] Stephen Robertson. On the History of Evaluation in IR. *Journal of Information Science*, 34:439–456, August 2008. ISSN 0165-5515. doi:10.1177/0165551507086989. 18

[188] Diego Antonio Rodríguez Torrejón and José Manuel Martín Ramos. CoReMo System (Contextual Reference Monotony) A

Fast, Low Cost and High Performance Plagiarism Analyzer System: Lab Report for PAN at CLEF 2010. In Braschler and Harman [28]. ISBN 978-88-904810-0-0. 95, 98, 99

[189] Diego Antonio Rodríguez Torrejón and José Manuel Martín Ramos. Crosslingual CoReMo System: Notebook for PAN at CLEF 2011. In Petras and Clough [156]. ISBN 978-88-904810-1-7. 95, 100

[190] T. G. Rose, M. Stevenson, and M. Whitehead. The Reuters Corpus Volume 1 - From Yesterday's News to Tomorrow's Language Resources. In *Proceedings of the Third International Conference on Language Resources and Evaluation*, 2002. 36

[191] Chanchal K. Roy and James R. Cordy. Scenario-Based Comparison of Clone Detection Techniques. In *ICPC '08: Proceedings of the 2008 The 16th IEEE International Conference on Program Comprehension*, pages 153–162, Washington, DC, USA, 2008. IEEE Computer Society. ISBN 978-0-7695-3176-2. 70

[192] Chanchal K. Roy and James R. Cordy. Towards a Mutation-based Automatic Framework for Evaluating Code Clone Detection Tools. In *C3S2E '08: Proceedings of the 2008 C3S2E conference*, pages 137–140, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-101-9. 76

[193] Chanchal K. Roy, James R. Cordy, and Rainer Koschke. Comparison and Evaluation of Code Clone Detection Techniques and Tools: A Qualitative Approach. *Sci. Comput. Program.*, 74(7):470–495, 2009. ISSN 0167-6423. 70

[194] Chanchal Kumar Roy and James R. Cordy. A Survey on Software Clone Detection Research. Technical Report 2007-541, School of Computing, Queen's University at Kingston, Ontario, Canada, 2007. 70

[195] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, New York, 1983. 15

[196] Vladislav Scherbinin and Sergey Butakov. Using Microsoft SQL Server Platform for Plagiarism Detection. In Stein et al. [213], pages 36–37. URL http://ceur-ws.org/Vol-502. 95

[197] Saul Schleimer, Daniel S. Wilkerson, and Alex Aiken. Winnowing: Local Algorithms for Document Fingerprinting. In *SIGMOD'03: Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 76–85, New York, NY, USA, 2003. ACM Press. ISBN 1-58113-634-X. 32, 90

[198] Anne Schuth, Maarten Marx, and Maarten de Rijke. Extracting the Discussion Structure in Comments on News-Articles. In *WIDM'07: Proceedings of the 9th annual ACM international workshop on Web information and data management*, pages 97–104, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-829-9. doi:10.1145/1316902.1316919. 107, 113

[199] D. Sculley. On Free Speech and Civil Discourse: Filtering Abuse in Blog Comments. In *CEAS 2008 - The Fifth Conference on Email and Anti-Spam, 21-22 August 2008, Mountain View, California, USA*, 2008. 112

[200] D. Sculley. *Advances in Online Learning-based Spam Filtering*. PhD thesis, Tufts University, USA, 2008. Carla E. Brodley. 112

[201] Leanne Seaward and Stan Matwin. Intrinsic Plagiarism Detection Using Complexity Analysis. In Stein et al. [213], pages 56–61. URL http://ceur-ws.org/Vol-502. 95

[202] Christin Seifert, Barbara Kump, Wolfgang Kienreich, Gisela Granitzer, and Michael Granitzer. On the Beauty and

Usability of Tag Clouds. *IV*, 0:17–25, 2008. ISSN 1550-6037. doi:10.1109/IV.2008.89. 127

[203] Sobha L., Pattabhi R. K Rao, Vijay Sundar Ram, and Akilandeswari A. External Plagiarism Detection: Lab Report for PAN at CLEF 2010. In Braschler and Harman [28]. ISBN 978-88-904810-0-0. 95, 98, 99

[204] Ian Soboroff and Donna Harman. Novelty Detection: The TREC Experience. In *HLT'05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 105–112, Morristown, NJ, USA, 2005. Association for Computational Linguistics. doi:10.3115/1220575.1220589. 122

[205] Efstathios Stamatatos. Intrinsic Plagiarism Detection Using Character *n*-gram Profiles. In Stein et al. [213], pages 38–46. URL `http://ceur-ws.org/Vol-502`. 91, 93, 94, 95, 96, 100

[206] Benno Stein. Fuzzy-Fingerprints for Text-Based Information Retrieval. In Klaus Tochtermann and Hermann Maurer, editors, *5th International Conference on Knowledge Management (I-KNOW 05)*, Journal of Universal Computer Science, pages 572–579, Graz, Austria, July 2005. Know-Center. 32, 34

[207] Benno Stein. Principles of Hash-based Text Retrieval. In Charles Clarke, Norbert Fuhr, Noriko Kando, Wessel Kraaij, and Arjen P. de Vries, editors, *30th Annual International ACM SIGIR Conference (SIGIR 07)*, pages 527–534. ACM, July 2007. ISBN 987-1-59593-597-7. 31

[208] Benno Stein and Maik Anderka. Collection-Relative Representations: A Unifying View to Retrieval Models. In A Min Tjoa and Roland Wagner, editors, *6th International*

Workshop on Text-Based Information Retrieval (TIR 09) at DEXA, pages 383–387. IEEE, September 2009. ISBN 978-0-7695-3763-4. doi:10.1109/DEXA.2009.50. 50, 129

[209] Benno Stein and Martin Potthast. Hashing-basierte Indizierung: Anwendungsszenarien, Theorie und Methoden. In Norbert Fuhr, Sebastian Goeser, and Thomas Mandl, editors, *Workshop Special Interest Group Information Retrieval (FGIR 06)*, Hildesheimer Informatikberichte, pages 159–166. University of Hildesheim, Germany, October 2006. URL http://opus.bsz-bw.de/ubhi/volltexte/2011/67/. 26

[210] Benno Stein and Martin Potthast. Applying Hash-based Indexing in Text-Based Information Retrieval. In Marie-Francine Moens, Tinne Tuytelaars, and Arjen P. de Vries, editors, *7th Dutch-Belgian Information Retrieval Workshop (DIR 07)*, pages 29–35, Leuven, Belgium, March 2007. Faculty of Engineering, Universiteit Leuven. ISBN 978-90-5682-771-7. 13, 26

[211] Benno Stein and Martin Potthast. Construction of Compact Retrieval Models. In Sándor Dominich and Ferenc Kiss, editors, *Studies in Theory of Information Retrieval. 1st International Conference on the Theory of Information Retrieval (ICTIR 07)*, pages 85–93, Budapest, October 2007. Foundation for Information Society. ISBN 978-963-06-3237-9. 13, 26

[212] Benno Stein, Sven Meyer zu Eißen, and Martin Potthast. Strategies for Retrieving Plagiarized Documents. In Charles Clarke, Norbert Fuhr, Noriko Kando, Wessel Kraaij, and Arjen P. de Vries, editors, *30th Annual International ACM SIGIR Conference (SIGIR 07)*, pages 825–826, New York, July 2007. ACM. ISBN 987-1-59593-597-7. 4

[213] Benno Stein, Paolo Rosso, Efstathios Stamatatos, Moshe Koppel, and Eneko Agirre, editors. *SEPLN 2009 Workshop on Uncovering Plagiarism, Authorship, and Social Software Misuse (PAN 09)*, 2009. Universidad Politécnica de Valencia and CEUR-WS.org. URL http://ceur-ws.org/Vol-502. 190, 200, 205, 208, 221, 222, 225, 228

[214] Benno Stein, Martin Potthast, and Martin Trenkmann. Retrieving Customary Web Language to Assist Writers. In Cathal Gurrin, Yulan He, Gabriella Kazai, Udo Kruschwitz, Suzanne Little, Thomas Roelleke, Stefan M. Rüger, and Keith van Rijsbergen, editors, *Advances in Information Retrieval. 32nd European Conference on Information Retrieval (ECIR 10)*, volume 5993 of *Lecture Notes in Computer Science*, pages 631–635, Berlin Heidelberg New York, 2010. Springer. ISBN 978-3-642-12274-3. doi:10.1007/978-3-642-12275-0_64. 13, 167

[215] Ralf Steinberger, Bruno Pouliquen, and Camelia Ignat. Exploiting Multilingual Nomenclatures and Language-Independent Text Features as an Interlingua for Cross-Lingual Text Analysis Applications. In *Proceedings of the 4th Slovenian Language Technology Conference. Information Society 2004 (IS'2004)*, 2004. ISBN 961-6303-64-3. 48, 49

[216] Ralf Steinberger, Bruno Pouliquen, Anna Widiger, Camelia Ignat, Tomaz Erjavec, Dan Tufis, and Daniel Varga. The JRC-Acquis: A multilingual aligned parallel corpus with 20+ languages. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC'2006)*, May 2006. 54

[217] Philip J. Stone. *The General Inquirer: A Computer Approach to Content Analysis*. The MIT Press, 1966. 126

[218] Pablo Suárez, Jose Carlos González, and Julio Villena. A Plagiarism Detector for Intrinsic, External and Internet

Plagiarism: Lab Report for PAN at CLEF 2010. In Braschler and Harman [28]. ISBN 978-88-904810-0-0. 95, 98, 99, 102

[219] Gábor Szabó and Bernardo A. Huberman. Predicting the Popularity of Online Content. *CoRR*, 2008. 115

[220] Bin Tan and Fuchun Peng. Unsupervised Query Segmentation Using Generative Language Models and Wikipedia. In Jinpeng Huai, Robin Chen, Hsiao-Wuen Hon, Yunhao Liu, Wei-Ying Ma, Andrew Tomkins, and Xiaodong Zhang, editors, *Proceedings of the 17th International Conference on World Wide Web, WWW 2008, Beijing, China, April 21-25, 2008*, pages 347–356. ACM, 2008. ISBN 978-1-60558-085-2. 139, 140, 142, 147, 151, 155, 157, 158, 161

[221] E. Tsagkias, M. de Rijke, and W Weerkamp. Predicting the Volume of Comments on Online News Stories. In *ACM 18th Conference on Information and Knowledge Managment (CIKM 2009)*, Hong Kong, November 2009. ACM, ACM. 115

[222] Peter D. Turney and Michael L. Littman. Measuring Praise and Criticism: Inference of Semantic Orientation from Association. *ACM Trans. Inf. Syst.*, 21(4):315–346, 2003. ISSN 1046-8188. doi:10.1145/944012.944013. 126

[223] Enrique Vallés Balaguer. Putting Ourselves in SME's Shoes: Automatic Detection of Plagiarism by the WCopyFind tool. In Stein et al. [213], pages 34–35. URL `http://ceur-ws.org/Vol-502`. 95

[224] Clara Vania and Mirna Adriani. External Plagiarism Detection Using Passage Similarities: Lab Report for PAN at CLEF 2010. In Braschler and Harman [28]. ISBN 978-88-904810-0-0. 95, 98, 99

[225] Adriano Veloso, Wagner Meira, Tiago Macambira, Dorgival Guedes, and Hélio Almeida. Automatic Moderation of Comments in a Large Online Journalistic Environment. In *Proceedings of the 2007 International Conference on Weblogs and Social Media (ICWSM 2007)*, Boulder, Colorado, U.S.A., March 2007. 112, 118, 121

[226] Alexei Vinokourov, John Shawe-Taylor, and Nello Cristianini. Inferring a Semantic Representation of Text via Cross-Language Correlation Analysis. In Suzanna Becker, Sebastian Thrun, and Klaus Obermayer, editors, *NIPS-02: Advances in Neural Information Processing Systems*, pages 1473–1480. MIT Press, 2003. ISBN 0-262-02550-7. 48

[227] Ellen M. Vorhees and Donna K. Harman. *TREC—Experiment and Evaluation in Information Retrieval*. MIT Press, 2005. ISBN 978-0-262-22073-6. 18

[228] Roger Weber, Hans-J. Schek, and Stephen Blott. A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces. In *Proceedings of the 24th VLDB Conference New York, USA*, pages 194–205, 1998. 27

[229] Geoffrey R. Whale. Identification of Program Similarity in Large Populations. *The Computer Journal*, 33(2):140–146, 1990. doi:10.1093/comjnl/33.2.140. 70

[230] Ian H. Witten, Alistair Moffat, and Timothy C. Bell. *Managing Gigabytes (2nd Ed.): Compressing and Indexing Documents and Images*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999. ISBN 1-55860-570-3. 15

[231] Daisuke Yamamoto, Tomoki Masuda, Shigeki Ohira, and Katashi Nagao. Collaborative Video Scene Annotation Based

on Tag Cloud. In *PCM'08: Proceedings of the 9th Pacific Rim Conference on Multimedia*, pages 397–406, Berlin, Heidelberg, 2008. Springer-Verlag. ISBN 978-3-540-89795-8. doi:10.1007/978-3-540-89796-5_41. 114

[232] Jung-Yeon Yang, Jaeseok Myung, and Sang-goo Lee. The Method for a Summarization of Product Reviews Using the User's Opinion. *International Conference on Information, Process, and Knowledge Management*, 0:84–89, 2009. doi:10.1109/eKNOW.2009.15. 112

[233] Yiming Yang, Jaime G. Carbonell, Ralf D. Brown, and Robert E. Frederking. Translingual Information Retrieval: Learning from Bilingual Corpora. *Artif. Intell.*, 103(1-2):323–345, 1998. ISSN 0004-3702. doi:10.1016/S0004-3702(98)00063-0. 48, 50

[234] Tae Yano, William W. Cohen, and Noah A. Smith. Predicting Response to Political Blog Posts with Topic Models. In *NAACL'09: Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics on Computational Linguistics*, pages 477–485, Morristown, NJ, USA, 2009. Association for Computational Linguistics. ISBN 978-1-932432-41-1. 115

[235] Shaozhi Ye, Ji-Rong Wen, and Wei-Ying Ma. A Systematic Study of Parameter Correlations in Large Scale Duplicate Document Detection. In *Proceedings of the 10th Pacific-Asia Conferenc on Advances in Knowledge Discovery and Data Mining (PAKDD)*, volume 3918 of *Lecture Notes in Computer Science*, pages 275–284. Springer, 2006. ISBN 3-540-33206-5. 37

[236] Wai Gen Yee, Andrew Yates, Shizhu Liu, and Ophir Frieder. Are Web User Comments Useful for Search? In Claudio Lucchese, Gleb Skobeltsyn, and Wai Gen Yee, editors,

Proceedings of the 7th Workshop on Large-Scale Distributed Systems for Information Retrieval, co-located with ACM SIGIR 2009, pages 61–68. CEUR-WS, July 2009. 114

[237] Xiaohui Yu and Huxia Shi. Query Segmentation Using Conditional Random Fields. In M. Tamer Özsu, Yi Chen, and Lei Chen 0002, editors, *Proceedings of the First International Workshop on Keyword Search on Structured Data, KEYS 2009, Providence, Rhode Island, USA, June 28, 2009*, pages 21–26. ACM, 2009. ISBN 978-1-60558-570-3. 138

[238] Mario Zechner, Markus Muhr, Roman Kern, and Michael Granitzer. External and Intrinsic Plagiarism Detection Using Vector Space Models. In Stein et al. [213], pages 47–55. URL http://ceur-ws.org/Vol-502. 95

[239] Chao Zhang, Nan Sun, Xia Hu, Tingzhu Huang, and Tat-Seng Chua. Query Segmentation Based on Eigenspace Similarity. In *Proceedings of the Joint conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP 2009). August, 2-7, 2009, Singapore. Short papers*, pages 185–188. Association for Computational Linguistics, 2009. URL http://www.aclweb.org/anthology/P/P09/P09-2047.pdf. 139, 140, 155, 156, 157, 158, 161

[240] Zhu Zhang and Balaji Varadarajan. Utility Scoring of Product Reviews. In *CIKM'06: Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 51–57, New York, NY, USA, 2006. ACM. ISBN 1-59593-433-2. doi:10.1145/1183614.1183626. 112

[241] Ying Zhao, George Karypis, and Usama Fayyad. Hierarchical Clustering Algorithms for Document Datasets. *Data Min.*

Knowl. Discov., 10(2):141–168, 2005. ISSN 1384-5810. doi:10.1007/s10618-005-0361-3. 83

[242] Li Zhuang, Feng Jing, and Xiao-Yan Zhu. Movie Review Mining and Summarization. In *CIKM'06: Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 43–50, New York, NY, USA, 2006. ACM. ISBN 1-59593-433-2. doi:10.1145/1183614.1183625. 112

[243] Justing Zobel and Yaniv Bernstein. The Case of the Duplicate Documents: Measurement, Search, and Science. In X. Zhao, J. Li, H.T. Shen, M. Kitsuregawa, and Y. Zhang, editors, *Proceedings of the APWeb Asia Pacific Web Conference*, pages 26–39, Harbin, China, January 2006. LNCS 3841. 25

[244] Du Zou, Wei-Jiang Long, and Ling Zhang. A Cluster-Based Plagiarism Detection Method: Lab Report for PAN at CLEF 2010. In Braschler and Harman [28]. ISBN 978-88-904810-0-0. 95, 98, 99

About the Author

Martin Potthast was born in Steinheim on the 24th of April 1981. He completed his secondary education at Gymnasium St. Xaver in Bad Driburg in 2000. After one year of civil service, he enrolled in computer science at the University of Paderborn in 2001. He received his Bachelor degree in early 2005, and finished his diploma degree mid 2006. Since then, he has been doctoral student at the Bauhaus-Universität Weimar.