Linking Documents by Distinctive Phrases

Heike Johannsen
H_Johannsen@web.de

Eberhard-Karls Universität Tübingen
Seminar für Sprachwissenschaft

Thesis submitted for the academic degree
Bachelor of Arts
March 2007

Supervisor:
Dr. Dale Gerdemann

Abstract

This essay will explore a very simplistic way of linking a collection of documents by a set of statistically calculated key phrases. Deviating from standard linking strategies in the field of automatic hypertext generation, the present study will challenge the monopoly of semantic similarity as the exclusive indicator of document relatedness and examine an alternative criterion provisionally named document *intersection*. Documents will be assumed to intersect if, at least at some point, they refer to the same entity. The suitability and the utility of *intersection* as a connection condition will be tested in an experiment, where, on the premise that entities are represented by keywords, documents are trivially linked if they share at least one key term.

Acknowledgements

Thanks a lot to Dale Gerdeman for supervising this thesis and helping with implementation problems and scientific research! Thanks also to Oda Limbach, Sabine Kupper, Isabelle Lehn and Peter Johannsen for spending hours with link evaluation, thus making this study possible in the first place!

Thank you!!!

Contents

Appendices

Figures

Tables

1. Introduction

1.1. Topic

This essay will explore a very simplistic way of linking a collection of documents by a set of statistically calculated key phrases. Deviating from standard linking strategies in the field of automatic hypertext generation, the present study will challenge the monopoly of *semantic similarity* as the exclusive indicator of document relatedness and examine an alternative criterion provisionally named document *intersection*. Documents will be assumed to *intersect* if, at least at some point, they refer to the same entity – represented by a key term. The suitability and the utility of *intersection* as connection condition will be tested in an experiment where documents (sampled from the German Wikipedia) are trivially linked if they share at least one keyword.

For introductory purposes, this essay will also provide a summary of standard term-weighting methods for keyword-extraction and give an overview of established techniques in the field of automatic hyperlink generation.

1.2. Outline

Section 1.3 will elaborate the motivation for the regression to a very basic linking tactic as an alternative device for the detection of subtle relations between documents. Chapter 2 will introduce a sample of prominent term-weighting methods for keyword-extraction, including *IDF*, *TF.IDF* and *RIDF*, and it will give an overview of established *similarity*-based techniques in the field of automatic hyperlink generation. Chapter 3 will present the design, the evaluation and the results of an experiment in which documents were linked entirely on the basis of common key terms selected according to a high *ridf*-value. Finally, chapter 4 will summarize the conclusions drawn from the previous experimental results. Additional information concerning notation and citation conventions, terminology, experimental results and the concrete implementation source-code is included in a set of appendices, attached at the end of this paper.

1.3. Motivation

> "Here's where things get interesting. We're at an inflection point in the evolution of findability. We're creating all sorts of new interfaces and devices to access information, and we're simultaneously importing tremendous volumes about people, places, products, and possessions into our ubiquitous digital networks" (Morville, 2005:2).

This survey departs from the hypothesis that, entirely on the basis of *semantic similarity*[1], certain relations among documents are just not findable.

Imagine you're an enthusiastic student of Latin-American literature and you plan to write an essay on Alejo Carpentier's famous novel *Los pasos perdidos* ('The Lost Steps'). From your lectures you already know that the protagonist is obsessed with the creation of an opus called *el treno*[2], that the novel implicitly but steadily alludes to *The Myth of Sisyphus* by Albert Camus and that one of the most significant passages is the following:

> "Ella no Penélope. Mujer joven, fuerte, hermosa, necesita marido. Ella no Penélope. Naturaleza mujer aquí necesita varón"[3] (Carpentier 1956:329).

In search of established literature on the issue you intuitively consult Google (http://www.google.com/) with the following query: [`"Ella no Penélope" "Naturaleza mujer aquí necesita varón" treno camus`]. But, as a result, you merely obtain the shocking message that there has been no match. Of course, you're smart and you guess that the search string might be too long. So you decompose it into the smaller units [`"Ella no Penélope"`], [`"Naturaleza mujer aquí necesita varón"`], [`treno`] and [`camus`] and google each one separately. This gives you two results for the first string, one for the second (actually one of the previous), about 7.480.000 for the third and roughly 9.780.000 for the fourth. Here you are! Since one of the first two results actually points to a discussion of the novel (Pezzella 2006), you can go ahead and write your paper on the poetics of failure.

Now assume you are not a student of literature but a linguist interested in the findability of documents. Then the most important observation about the given

---

[1] *Semantic similarity* is informally defined as the extent to which two documents treat the same topic, usually measured in terms of the amount of (key-)words they have in common. For a more detailed treatise see section 2.2.

[2] Archaic Greek lyrical composition (Wikipedia.org 2006:http://es.wikipedia.org/wiki/Treno).

[3] 'She not Penelope. Young woman, strong, beautiful, needs a husband. She not Penelope. Nature woman here needs man'.

example is that the very first query failed to return a result because there was no sufficiently *similar* document (containing all query terms) available. Consequently, Google's requirement of a high *semantic similarity* prevented the delivery of a very relevant link. The second interesting phenomenon is that the search string [`"Ella no Penélope"`] delivered two documents but that the search string [`"Naturaleza mujer aquí necesita varón"`] delivered just one. Looking at the results more closely reveals that the link missed with the latter query points to a document containing the original novel text itself. This peculiar omission is probably brought about by the Fact that Google uses *term frequency* (see section 2.1.2.1) to score the relevance of page content[4]. If so, the string [`"Ella no Penélope"`] succeeds to retrieve the novel because it appears twice therein, whereas the string [`"Naturaleza mujer aquí necesita varón"`] is inapplicable because it appears just once[5]. The example thus clarifies an important corollary: The most important fragments of a text need not be the most frequent ones and accordingly, *semantic similarity* (usually relying on frequent terms) may occasionally fail to retrieve the most relevant documents[6].

But is there an alternative to high *semantic similarity* as relation detector? Obviously, the alternative is to extend connectivity to documents with low (but nonzero) *similarity*. Of course, in the light of a permanently increasing information overload in the world wide web, this is clearly a counterintuitive objective, and for commonplace browsing situations, imposing high *similarity* thresholds is doubtless the right policy. Nevertheless, there are domains in which the least evident relations can be the most interesting ones. The set of inclined fields includes literary criticism, history, comparative religion studies, journalism and virtually every kind of interdisciplinary research. Hence, it seems worthwhile to offer mechanisms that optionally provide additional links, thus bridging superficially discrete but implicitly related documents.

---

[4] The Google *content score* is the product over the summed frequency variants of document-contained query words (Langville/Meyer 2006:22).

[5] Possibly the decisive frequencies are not those of the exact search strings but those of key phrases contained therein, f.e. *Penélope* and the ungrammatical *Naturaleza mujer*, of which the former appears three times in the novel, whereas the latter occurs only once, both in the novel and in the review. Presumably, if Google normalizes scores by text length, the low frequency of the second term does not have such a severe impact in retrieval of the review, because the review is a lot shorter than the novel.

[6] Note that there can also be the opposite effect. Given that THIS document contains six ocurrences of the string *Naturaleza mujer aquí necesita varón*, it is highly inclined to be ranked as the most relevant document for a conforming query, according to *similarity*.

The most naïve approach to detect such low but nonzero *similarity* is by means of a common key term, here designated *intersection*. As Zeng and Bloniarz (2004:pdf:1) point out, "keywords are succinct descriptions of important topics and characterize document content", which makes them also a central element in *similarity*-based approaches (see section 2). The distinction between *intersection* and *similarity* thus reduces to the requirement that an *intersection* must be large in order to substantiate high *similarity*[7]. From a more philosophical point of view, the difference is essential: The larger an *intersection* between two or more documents, the larger the amount of information they have in common. And vice versa: The smaller the *intersection*, the more complementary information. Escalated to the extreme, a minimal document *intersection* should come along with a maximum of new information, whereas high document *similarity* – ideally – retrieves information that is already known[8].

Still, the most obvious disadvantage of *intersection*-based linking is that it is very susceptible to *noise*. So, a major question is whether there is a way to compensate for an inevitable contamination with irrelevant material[9]. In this respect, a very promising conception is realized in Amazon's *Statistically Improbable Phrases* (*SIP*s). *SIP*s are characteristic key phrases that reflect book content and distinguish among books from the same domain:

> "SIPs are not necessarily improbable within a particular book, but they are improbable relative to all books in Search Inside!. For example, most SIPs for a book on taxes are tax related. But because we display SIPs in order of their improbability score, the first SIPs will be on tax topics that this book mentions more often than other tax books. For works of fiction, SIPs tend to be distinctive word combinations that often hint at important plot elements" (Amazon.com 2007:http://www.amazon.com/gp/search-inside/sipshelp.html).

Figure 1 shows the *SIP*s associated with the book *Dogs for Dummies* (Spadafori 2001).

---

[7] This characterization neglects approaches involving synonyms or *LSA* (see section 2.2).

[8] Paradoxically, *similarity*-based information retrieval works so well, because, in practice, the ideal is hardly ever reached. To put it in Golovchinsky's (1997) diction: Queries? Documents? Is there a difference? Obviously, the difference is substantial and relates to length: A query is typically much shorter than the retrieved document, thus leaving room for a large complement. For documents of equal size, however, maximal similarity implies identity.

[9] The notion of non-relevance is subjective. As frequently stressed in the discourse of automatic hypertext generation, different users prefer different links (Tebutt 1998, Golovchinsky 1998, El-Beltagy et al. 2001).

Figure 1: Amazon's *SIP*s for the book *Dogs for Dummies* (Spadafori 2001)

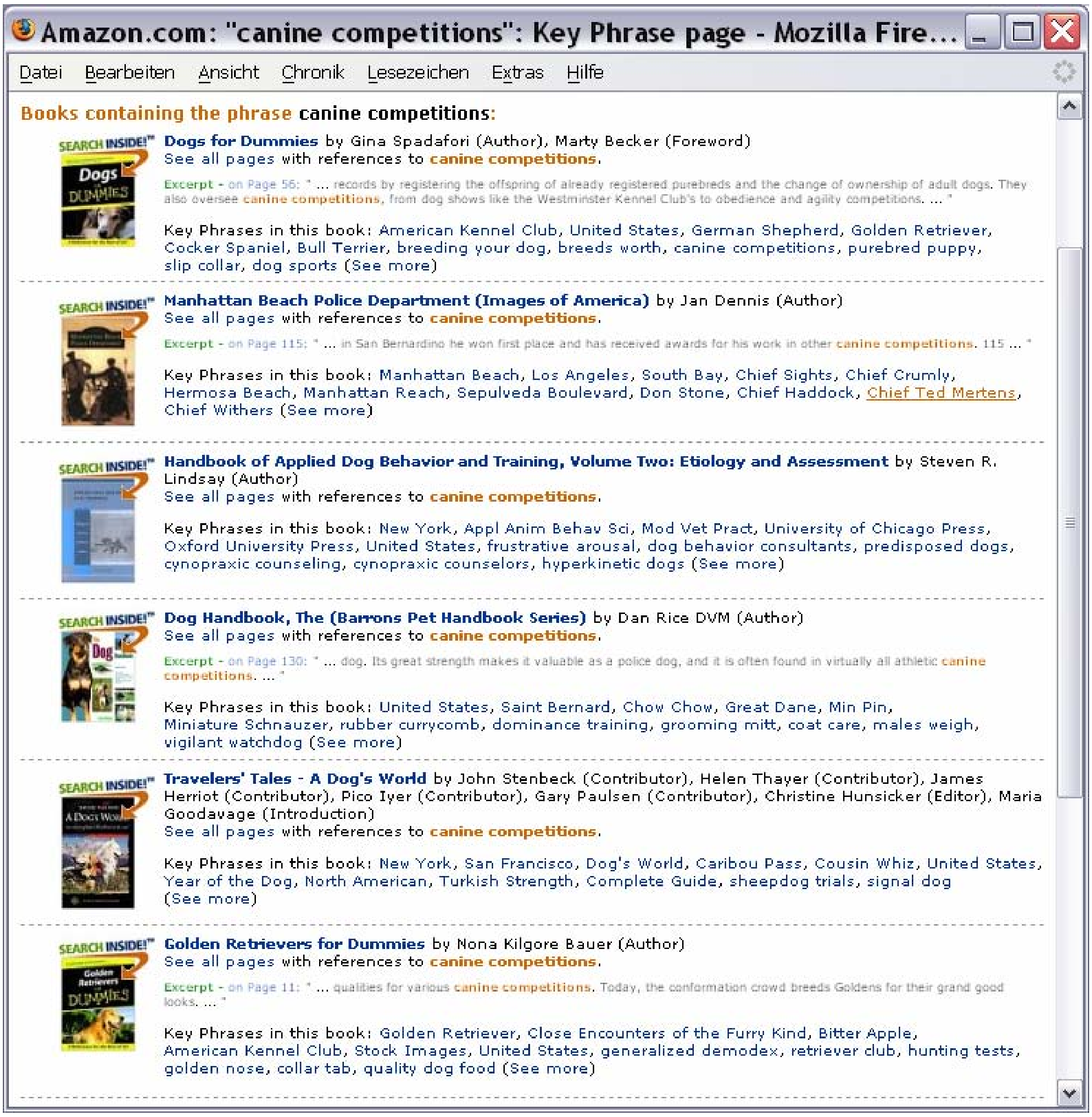

Figure 2: Amazon books linked by the *SIP canine competitions*

As expected, all of the *SIP*s exhibit a strong relation to the domain 'dog husbandry'. Moreover, all of them constitute anchors of hyperlinks that guide the user to a selection of other containing books, as those in Figure 2. Not surprisingly, these books also stem from the domain of dogs.

In sum, *SIP*s perform multiple functions. First of all, they serve to characterize their source book. Secondly, they (re)direct the user to other presumably attractive books. Thirdly, and most importantly here, they characterize the relation that holds between the connected books, namely the joint involvement of a common subtopic expressed by the *SIP* itself. In this sense, Amazon provides a highly efficient solution for relevance assessment: Let the individual user decide whether the concept of interest is *reputable breeder* or *flat collar*.

Inspired by Amazon's *SIP*s, this survey will describe a highly resembling method for document linking via key phrases, in which distinctive n-grams are identified by means of a high *ridf*-value (see section 2.1.2.4) and in which documents receive a connection in case they have at least one common key term.

2. Theoretical Foundations and Related Work

This chapter will provide an overview of the theoretical foundations of corpus-based keyword-extraction via term-weighting methods such as *IDF*, *TF.IDF* and *RIDF* and it will give a summary of established approaches to automatic hyperlink generation based on vector-space models. Moreover, it will address some structural properties of hyperlink networks and introduce a standard evaluation methodology for retrieval systems.

2.1. Keyword-Extraction

There are two primary ways to associate a document with a set of keywords (Montejo-Ráez/Steinberger 2004:htm). The first is to resort to a controlled keyword vocabulary and to assign a selection thereof to the target document – possibly manually. The second is to extract a register of key terms from the documents themselves – preferentially automatically. This latter approach presupposes the availability of a sufficiently large text corpus which constitutes "a representative sample of the population of interest" (Manning/Schütze 1999:119). Given such a corpus, good keywords can be identified by their distributional properties.

2.1.1. Standard Conception of Distinctive Phrases

A 'good keyword' for information retrieval typically has to meet two requirements: It should be representative of the containing document's content and it should be pertinent to discriminate or associate documents within a collection (Montejo-Ráez/Steinberger 2994:htm). The former constraint implies that key n-grams should comprise lexical words and has motivated numerous preprocessing procedures in various extraction systems, such as the employment of *stopword-lists*[10], *stemming* or *lemmatization*[11], *POS tagging*[12] or *chunking*[13]. The latter constraint requires that a key term should be stereotypical for a (sub-)domain of discourse. Notably, there is no entailment relation between these two prerequisites. To give an example, an n-gram such as *let x be* is perfectly applicable to distinguish a scientific article from fiction but it doesn't tell much about the article's specific topic. Conversely, a word like *penis* gives a significant clue about the containing document's content. Nevertheless, especially in the ambience of the world wide web, penises are discussed from an enormous variety of perspectives, including eroticism, humor, medical science and yellow press articles[14], which renders the term highly inappropriate for the differentiation of genres[15].

As stated earlier, there are various statistical techniques to identify phrases that, though possibly to different extents, comply with both criteria. The next section will introduce a subset thereof.

---

[10] Stopwords are usually irrelevant function words. A *stopword-list* is a list of items to be ignored.

[11] "Stemming is the process for reducing inflected (or sometimes derived) words to their stem, base or root form — generally a written word form" (Wikipedia.org 2006:http://en.wikipedia.org/wiki/Stemming). "[L]emmatisation is the process of determining the [base-form] for a given word" (wikipedia.org 2006:http://en.wikipedia.org/wiki/Lemmatization). In the context of term extraction, both *lemmatization* and *stemming* aim at collapsing several morphological realizations of the same paradigm into a single representative form.

[12] *Part-of-Speech* (*POS*) *tagging* "is the process of marking up the words in a text as corresponding to a particular [word class], based on both its definition, as well as its context" (Wikipedia.org 2006: http://en.wikipedia.org/wiki/Part_of_speech_tagging). *POS* information is essential in frameworks where certain grammatical categories are considered more accurate than others. For instance, Heyer et al. (2006:223) claim that the best keywords are always nouns.

[13] "Chunking, also called shallow or partial parsing, applies shallow processing techniques (typically regular expressions and finite automata) to group together words to larger syntactic and meaning-bearing constituents" (Cimiano 2006:39).

[14] For an example see http://www.krone.at/index.php?http://wcm.krone.at/krone/C12/S22/A7/object_id__10152/hxcms/.

[15] Of course, both terms used in the example could be beneficial if interacting in the surroundings of an entire indexing vocabulary. For a brief discussion of indexing norms see Soergel (1999).

2.1.2. Term-Weighting

Term-weighting is a technique to assess the significance of a term within a document or collection, taking advantage of the fact that eminent keyword candidates tend to exhibit a non-random distribution within or across documents.

2.1.2.1. Term Frequency (TF), Collection Frequency (CF) and Document Frequency (DF)

Each of the statistical term-weighting measures to follow exploits at least one of three relevant quantities: *term frequency*, *collection frequency* and *document frequency*, which are (informally) defined as follows (Manning/Schütze 1999:541-544) [16]:

(1) The *term frequency* tf_{ij} of a term w_i in a document d_j is the number of occurrences of w_i in d_j.

(2) The *collection frequency* cf_i of a term w_i is the total number of occurrences of a term w_i in a collection of documents.

(3) The *document frequency* df_i of a term w_i is the number of documents in the collection in which w_i occurs at least once.

According to Manning and Schütze (1999:542), *term frequency* roughly reflects the prominence of a term in a document: the more frequent, the more salient. *Document frequency*, on the other hand, expresses the degree of informativeness of a term within a collection, where semantically unfocussed terms spread out evenly, while focussed words tend to agglomerate in a potentially very small subset of documents. Still, none of these frequency counts is by itself a good indicator of the extent to which a term is representative or distinctive of document content, especially since the most frequent words in natural language are function words, and therefore bad keyword candidates (Heyer et al. 2006:87).

2.1.2.2. IDF

The measure known as *inverse document frequency* (*IDF*) was first proposed by Karen Spärck Jones in 1972 (Spärck Jones 2004:pdf:6) and is based on the intuition that terms occurring in few documents are more distinctive than terms that

[16] Note that there is a difference in terminology between Manning/Schütze (1999) and Yamamoto/Church (1998). The quantity referred to as *collection frequency* by Manning and Schütze is called *term frequency* by Yamamoto and Church, who relinquish the concept of *term frequency* as defined in (11) altogether.

widely disperse across documents (Robertson 2004:pdf:1). A formal definition is given in (4) (Manning/Schütze 1999:553, Robertson 2004:pdf:2):

(4) $$idf_i = IDF(w_i) = \log\left(\frac{D}{df_i}\right)$$

where *D* is the total number of documents in the collection.

Hence $0 \leq idf_i \leq \log(D)$, such that *idf* is minimized for those terms that occur in all documents and maximized for terms that occur in a single document. Consequently, *IDF* assigns more weight to specialized elitist expressions than to prevalent common terms and function words[17]. Up to the present, *IDF* has been an enormous success in the field of information retrieval, incorporating into almost every weighting scheme (Robertson 2004:pdf:1).

2.1.2.3. TF.IDF

The term *TF.IDF* designates a whole family[18] of weighting schemes that combine a *tf* factor with an *idf* factor. The simplest form of a *TF.IDF* weighting scheme is the following (Salton 1989:280):

(5) $$tf.idf_{ij} = TF.IDF(w_{ij}) = tf_{ij} \cdot \log\frac{D}{df_i} = tf_{ij} \cdot idf_{ij}$$

However, as stated by Salton and Buckley (1988:pdf:5), the formula in (5) unwarrantedly penalizes short documents and should therefore be normalized – for example by the total document vocabulary weight:

(6) $$tf.idf'_{ij} = \frac{tf.idf_{ij}}{\sum_k tf.idf_{kj}}$$

Baeza-Yates and Ribeiro-Neto (1999:29) suggest an alternative normalization strategy using the *term frequency* of the most frequent term in the document as a normalization factor:

(7) $$tf.idf'_{ij} = \frac{tf_{ij}}{\arg\max(tf_{kj})} \cdot idf_{ij}$$

[17] According to Robertson (2004:pdf:1-2), there have been many approaches to replace the heuristic motivation of *IDF* by a theoretically more profound one. Manning and Schütze (1999:551-553) present a probabilistic derivation of *IDF* which is summarized in Appendix B.

[18] Examples of various *TF.IDF* weighting schemes are listed in Manning/Schütze (1999:544).

Normalization aside, *TF.IDF* as defined in (5)-(7) reinstates the idea originally associated with *tf* that lexical words with a high document-internal frequency are good representatives of the document's topic. The *idf* factor, on the other hand, correctively suppresses non-distinctive items, especially function words. Consequently, only those terms that are both prominent and distinctive receive a high *tf.idf* value[19]. Until today, *TF.IDF* has turned out to be extremely robust and hard to beat in the field of document-related keyword-identification[20] (Salton/Buckley 1988, Baeza-Yates/Ribeiro-Neto 1999).

2.1.2.4. RIDF

The statistical measure finally chosen to extract the linking key phrases in the experiment conducted here (for reasons discussed in section 3.2.3) is the so-called *residual inverse document frequency* (*RIDF*). The *ridf* value of a word w_i is defined as the difference between the logs[21] of the actual sample *idf* and the *idf* expected if w_i was Poisson distributed (Manning/Schütze 1999:545,553-554):

(8) $$ridf_i = RIDF(w_i) = \log_2 \frac{D}{df_i} - \log_2(1 - p(k = 0; \lambda_i = \frac{cf_i}{D}))$$

where $p(k;\lambda_i)$ is the Poisson distribution of k with parameter λ_i:

$$p(k;\lambda_i) = e^{-\lambda_i} \frac{\lambda_i^{\,k}}{k!}$$

In (8), the random variable k models the probability of a document having exactly k occurrences of w_i, given that the average number of occurrences of w_i per document is λ_i. The term $1 - p(0;cf_i/D)$ is the Poisson probability of a document having at least 1 occurrence of the term w_i. With $k = 0$ and $\lambda_i = cf_i/D$ the equation in (8) reduces to (9):

---

19 Since *IDF* can be derived from a probabilistic model (see Appendix B), Salton and Buckley (1988:pdf:5) assume that the same is true for *TF.IDF*. However, Robertson (2004:pdf:11) stresses that this conjecture is problematic, because the *tf* factor alters the event space on which the probabilistic model used to justify *IDF* operates.

20 Though the selection criteria for Amazon's *SIP*s are essentially a black box, the following statement by Amazon.com strongly suggests the involvement of a *TF.IDF* weighting scheme:

"Amazon.com's Statistically Improbable Phrases, or "SIPs", show you the interesting, distinctive, or unlikely phrases that occur in the text of books in the Search Inside!™ program. Our computers scan the text of all books in the Search Inside! program. If they find a phrase that occurs a large number of times in a particular book relative to how many times it occurs across all Search Inside! books, that phrase is a SIP in that book" (Amazon.com 2007:http://www.amazon.com/gp/search-inside/sipshelp.html).

21 Note that the base of the log used in *IDF* is not vitally important (Robertson 2004:pdf:4). Therefore, the different bases used in (4) and in (8)-(9) do not manifest a contradiction.

(9) $ridf_i = RIDF(w_i) = \log_2 \frac{D}{df_i} - \log_2(1 - e^{-\frac{cf_i}{D}}) = idf_i - \log_2(1 - e^{-\frac{cf_i}{D}})$

Crucially, assuming a Poisson distribution for a word implies that occurrences of that word are independent (Manning/Schütze 1999:546). Since this condition is approximately true for functional categories but not for content words, high deviation from Poisson is a good indicator of lexical vocabulary (Manning/Schütze 1999: 547,554). Yamamoto and Church (1998:pdf:1) confirm that

> "[...] RIDF tends to highlight technical terminology, names, and good keywords for information retrieval (which tend to exhibit nonrandom distributions over documents)".

2.1.2.5. MI

Mutual information (*MI*) is a measure from information theory frequently used in the identification of *collocations*[22] (Church/Hanks 1990, Yamamoto/Church 1998). In its general form, *MI* is defined as follows (Manning/Schütze 1999:67):

(10) $I(X,Y) = \sum_{x,y} p(x,y) \log_2 \frac{p(x,y)}{p(x)p(y)}$ where *X* and *Y* are random variables.

MI can be thought of as a measure of the degree of (in-)dependence between the two variables *X* and *Y*. *MI* is minimal (*mi* = 0) exactly when the variables are independent. It raises with increasing dependency, according to the *entropy*[23] of the variables. In this sense, *MI* measures the amount of shared information in the random variables (Manning/Schütze 1999:67).

Yamamoto and Church (1998:pdf:18) redefine *MI* as a function of the probability of certain substrings (with length ≥ 2) in a corpus:

(11) $i_i = I(w_i = xYz) = \log \frac{p(xYz)}{p(xY)p(z \mid Y)} = \log \frac{tf(xYz)tf(Y)}{tf(xY)tf(Yz)}$

where *x* and *z* are tokens, *Y* is a sequence of tokens and *tf*(*Y*) = *C* iff *Y* is empty, and where *C* is the size of the corpus.

---

[22] "A COLLOCATION is an expression consisting of two or more words that correspond to some conventional way of saying things. [...] Collocations are characterized by limited *compositionality*. We call a natural language expression compositional if the meaning of the expression can be predicted from the meaning of its parts. Collocations are not fully compositional in that there is usually an element of meaning added to the combination" (Manning/Schütze 1999:151).

[23] While *MI* is a measure of the common information of two random variables, *entropy* can be thought of as the amount of information contained in a single random variable. The *entropy H* of a random variable *X* is defined below (Manning/Schütze 1999:61, Shannon 1948:pdf:11):

$H(X) = -\sum_{x \in X} p(x) \log_2 p(x)$

Yamamoto and Church (1998:pdf:26) then propose that *MI* as defined in (11)[24] "looks for *n*-grams whose internal structure cannot be attributed to compositionality". However, this claim has not been confirmed in the course of the conducted experiment, as emphasized in section 3.2.2.

2.1.2.6. Other term-weighting approaches

The term-weighting schemes discussed above are by no means the only techniques for keyword-extraction. A laconic summary of other popular term-weighting methods from information retrieval, information filtering and text classification, such as *information gain* (*IG*), χ^2 *chi square* (*CHI*) and *relative document frequency* (*RelDF*) is given in Nanas et al. (2003). Yet another approach using *pointwise Kullback-Leibler divergence* comes from Tomokiyo and Hurst (2003). Tomokiyo and Hurst construct four language models, viz. a unigram model and an n-gram model on a foreground and a background corpus[25], in order to determine the 'phrasiness' and the informativeness of an n-gram as a means of *KL-divergence* between the respective models. In the present context, the interesting point about this approach is that it compels the extracted n-grams to be genuine cohesive phrases in the syntactic sense. Though the experiment conducted here will not enforce syntactic phrasiness, phrasiness might be a highly desirable property, as pointed out in section 3.3.

2.2. Methods in the field of automatic hyperlink generation

The classical approach to content-based hyperlink generation[26] is to employ a vector-space model for *similarity* calculation (Salton/Allan 1993, Salton et al. 1994, Goffinet/Noirhomme-Fraiture 1995, Allan 1996, Allan 1997, Green 1998, Zeng/Bloniarz 2004). Documents and queries can be modelled as feature-vectors, where features correspond to the collection vocabulary (or alternatively to the collection inventory of key terms) and values correspond to some type of term-related frequency data – in the simplest case, the binary information on presence

---

[24] Note that definition (11) does not preserve certain properties of *MI* as defined in (10). In particular, if defined as in (11), *MI* can be negative.

[25] The foreground corpus is the (domain-specific) source corpus for keyword-extraction. The background corpus is a (general) control corpus.

[26] As opposed to structure-based link generation that utilizes internal document organization into coherent subunits such as headlines and paragraphs as the connection criterion.

or absence of a term within the document or query, in more complex models, *term frequency* or term weights. In this sense, a document or query can be pictured as a point in a multidimensional space[27] (Jurafsky/Martin 2000:647-651). Since the most popular weighting scheme in automatic hyperlink generation appears to be *TF.IDF* (Salton/Allan 1993, Salton et al. 1994, Goffinet/Noirhomme-Fraiture 1995, Allan 1997, Green 1998), most linking systems use document vectors as in (12):

(12) $\vec{d}_j = (tf.idf_{1j}, tf.idf_{2j} ... tf.idf_{Dj})$

Once the document vectors have been compiled, the next step is to compute *similarities* between pairs of documents and to link those documents whose *similarity* value exceeds a given threshold. A well-established measure of document *similarity* is the *cosine* of the angle between the vectors[28] (Goffinet/Noirhomme-Fraiture 1995, Allan 1997), defined in (13)[29] (Jurafsky/Martin 2000:651):

(13) $$\cos(\vec{u},\vec{v}) = \frac{\sum_{i=1}^{D} u_i v_i}{\sqrt{\sum_{i=1}^{D} u_i^2}\sqrt{\sum_{i=1}^{D} v_i^2}}$$

The *cosine* is minimal for orthogonal non-*intersecting* documents and maximal for identical documents (Jurafsky/Martin 2000:650). In this sense, semantic proximity is reproduced by spatial proximity (Manning/Schütze 1999:539).

A common problem for the determination of document similarity via vectors arises from keyword synonymy and polysemy[30]. As a solution for polysemy, Salton et al. suggest the use of local and global vectors to resolve ambiguities: Only documents that exhibit a high global similarity but also contain some locally matching passages are connected (Salton/Allan 1993:ps:3, Salton et al. 1994:pdf:3). Local similarity thus improves *precision* (see section 2.4). Tebutt (1998) describes a slightly different strategy, which first identifies significant passages in a document and subsequently takes these passages as query for retrieval of similar passages or documents. Green (1998) proposes the use of

---

[27] Note that the underlying conception of a document, also called a *bag-of-words* model, completely disregards linear ordering of constituents and syntactic constraints (Jurafsky/Martin 2000:647).

[28] Other similarity measures include the simple scalar product and the *Euclidean distance* (Heyer/Quasthoff/Wittig 2006:206). Green (1998:pdf:4) uses the *z-score*.

[29] In the denominator, vectors are normalized by vocabulary size (Jurafsky/Martin 2000:650).

[30] Polysemy "is the capacity for a sign (e.g. a word, phrase, etc.) or signs to have multiple meanings" (Wikipedia.org 2007:http://en.wikipedia.org/wiki/Polysemy).

lexical chains[31] rather than key phrases as vector features, where a lexical chain is identified by the affiliation of its members to a WordNet[32] synset[33]. Macedo et al. (2002), on the other hand, promote the integration of *latent semantic analysis* (*LSA*)[34] to overcome the complications due to synonymy and polysemy.

Another topic in the field of automatic hypertext generation is hyperlink typing. Allan (1996, 1997) describes several techniques to reduce *graph* complexity by link merging. Furthermore, he introduces a taxonomy in which the remaining links are subdivided into three classes: *automatic*, *pattern-matching* and *manual*[35]. The *automatic* class is further subdivided into *revision*, *summary/expansion*, *equivalence*, *comparison*, *tangent* and *aggregate.* The most interesting of these types seem to be the *tangential* and *aggregate* links. *Aggregate* links are those that interconnect a cluster of documents, associated with the same target document. A *tangential* link can be identified by its non-connectedness to a cluster of *aggregate* documents relating to the same document or passage. Tangential links thus reflect upon unusual relations between documents (Allan 1997:pdf:9). The idea of including a type with a link reappears in Macedo et al. (2002:pdf:5) who store typed semantic relations such as *explanation* or *comment* along with the link. Apparently, the display of linking key phrases manifested in the Amazon *SIP*s also constitutes an implicit way of link typing, though a very lax one.

A slightly more recent approach to automatic link augmentation, settled within open hypermedia environments[36], is to decouple hyperlinks from documents and to store them in so-called *linkbases*. More precisely, abstract links are derived

---

[31] "[A lexical chai*n*] is a list of words that captures a portion of the cohesive structure of the text" (Wikipedia.org 2007:http://en.wikipedia.org/wiki/Lexical_chain). For example, a list of words that refer to the same entity within a given context.

[32] "WordNet is a semantic lexicon for the English language. It groups English words into sets of synonyms called *synsets*, provides short, general definitions, and records the various semantic relations between these synonym sets" (Wikipedia.org 2007:http://en.wikipedia.org/wiki/WordNet).

[33] "According to WordNet, a *synset* or synonym set is defined as a set of one or more synonyms that are interchangeable in some context without changing the truth value of the proposition in which they are embedded" (Wikipedia.org 2007:http://en.wikipedia.org/wiki/Synset).

[34] *LSA* is a technique that reduces vector dimensionality, thereby projecting semantically similar terms into the same dimensions, where semantic similarity is determined by common co-occurrence patterns. Hence, in the reduced space, two semantically similar documents (documents that have synonymous vocabulary) have a high *cosine* similarity even in case they don't share any key terms (Manning/Schütze 1999:556).

[35] The complete taxonomy is reproduced in Table 9 in Appendix C.

[36] "Hypermedia is a term [...] used as a logical extension of the term hypertext, in which graphics, audio, video, plain text and hyperlinks intertwine to create a generally non-linear medium of information" (Wikipedia.org 2007:http://en.wikipedia.org/wiki/Hypermedia).

from existing links (f.e. contained in a collection of web-pages previously visited), such that details about the source anchor selection (but not the source anchor location) and the link destination are saved in the *linkbase*. Whenever a new web-page is visited, these *generic* links are retrieved from the database and added to the new document iff there is either an exact match or sufficient *similarity* between a text fragment and the stored source anchor content (Lewis et al. 1996:html). El-Beltagy et al. (2001) further condition link augmentation on the *user-context*, which is defined as a coherent collection of relevant documents (f.e. a set of bookmarked documents) and represented as the *cluster centroid*[37] of the respective document vectors. Links are extracted from the *user-context*-documents and the *user-context*-representative is stored along with the set of extracted links. Given that there is an unlimited number of dynamically changing *user-contexts*, each time a new document is to be link-augmented, the system first determines the *user context* closest to document content and subsequently selects the appropriate set of links. After all, these systems still rely on vector similarity, though they don't exactly compare documents. For further details about adaptive link augmentation see Bailey et al. (2001) and Camacho-Guerrero et al. (2004).

In sum, the standard is to take *semantic similarity*, represented as *spatial similarity*, as the decisive criterion for document linking. Yet, there are counterexamples whereof one comes from Cleary and Bareiss (1996) who severely criticize the employment of *spatial similarity* as link induction controller. Their first objection to *similarity* is that it does not specify the nature of a relation that holds between similar documents. Their second objection is that similarity might fail to link conceptually related documents due to a lack of identical key terms (Cleary/Bareiss 1996:pdf:3). As an alternative, they suggest a linking procedure via manually annotated *concepts* (Cleary/Bareiss 1996:pdf:5-8). Though, in the present study, manual annotation is rejected as infeasible and much of the argumentation against *similarity* becomes invalid in the light of the work presented above, the distrust in the thoroughness of *similarity* reflects a major question in the present survey: Is *spatial similarity* a necessary premise for document-relatedness? As stated earlier, this study advocates for 'no'.

---

[37] In a vector-space model, the *centroid* of a cluster of points (each represented by a vector), is another vector, where each value corresponds to the average over the values of the cluster vectors in the respective dimension (Manning/Schütze 1999:499).

2.3. Structural Characteristics of Hyperlink Networks

Every hyperlink structure can be represented as a *graph*[38]. In the case of a linked text collection, documents correspond to *vertices* and links to *edges* or *arcs*.

Steyvers and Tenenbaum (2005:pdf:4-9) suggest four statistical features of *graphs*[39], to characterise three network types: *random graphs*, *scale-free* networks and *small-world* structures[40]. In a *random graph*, for each pair of *vertices*, the probability of being connected is equal. A *small-world* structure, on the other hand, is typified as a highly clustered non-random network with a short average *path* length, supposedly arising from a *scale-free* formation, which exhibits an uneven *degree distribution* and therefore manifests all shades of connectivity simultaneously[41]. Figuratively, in a *small-world graph* most *vertices* interweave around highly prominent hubs. However, connections between peripheral *vertices* pertaining to distinct hubs are sparse. As shown by Watts and Strogatz (1998), *small-world* models capture a wide range of network-related phenomena in nature, social life and technology, including properties of the world wide web and semantic networks (Steyvers/Tenenbaum 2005).

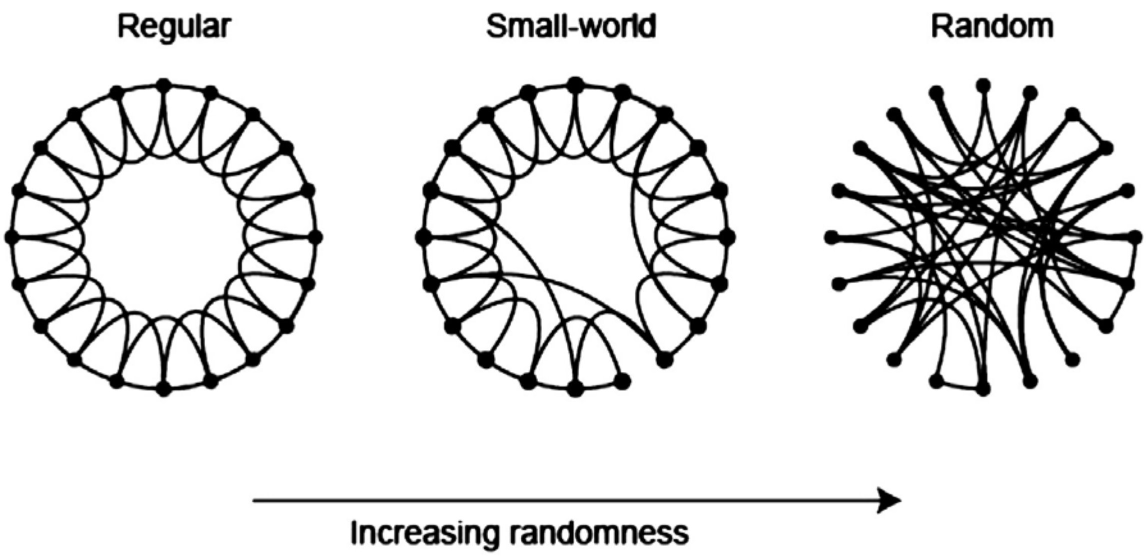

Figure 3: a *small-world graph* between the regular and the random extreme (Watts/Strogatz 1998:pdf:2)

---

[38] For an overview of the *graph*-related terminology used here, see Appendix D.

[39] Namely the *average distance*, the *diameter*, the *clustering coefficient* and the *degree distribution*.

[40] The so-called *small-world* phenomenon relates to the fact that the chain of acquaintances connecting two randomly selected persons on earth is on average very short (Watts 1999:11-12).

[41] More precisely, the *degree distribution* is determined by a *power-law* relationship. A *power-law* relationship between two quantities x and y can be formalized as $y=ax^k$, where a and k are constants (Wikipedia.org 2007:http://en.wikipedia.org/wiki/Power_law).

Interestingly, Steyvers and Tenenbaum (2005:pdf:24) claim that *small-world* properties hardly arise in semantic models based on vector-space *similarity* such as *LSA* (see chapter 2.2). As a reason, they suggest that semantic, self-organized networks subsist in a temporal dimension and grow according to the principle of *preferential attachment*, which states that the probability that a new *vertex* will connect to an existing one depends on the connectivity of the target *vertex* (Barabási et al. 2000:ps:6). In other words, nodes with a high *degree* are more likely to acquire new connections than nodes with a low *degree*. In contrast to such growing networks, the creation of links based on vector-space models is an instantaneous operation on a static set of *vertices* in which all links emerge simultaneously. This contrast indicates a fundamental difference between growing network and spatial representations of semantic knowledge (Steyvers/Tenenbaum 2005:pdf:23). After all, link structures derived from document *intersection* should exhibit *complete components* wherever key terms are shared. Average path lengths should presumably be short. Unfortunately, an experimental or empirical verification of this conjecture is beyond the scope of this study.

2.4. Established Evaluation Methodology

> "The evaluation of information retrieval or text linking operations is a major unsolved problem [...] The concept of document relevance must be settled outside the retrieval environment" (Salton et al. 1994:pdf:9).

The standard evaluation measures used in information retrieval are *precision* and *recall*, where *precision* measures how much of the extracted information is actually valid and where *recall* captures the total coverage of the system (Jurafsky/Martin 2000:578). Manning and Schütze (1999:268-269) provide the following definitions:

(14) $$precision = \frac{|\,true\ positives\,|}{|\,true\ positives\,| + |\,false\ positives\,|}$$

(15) $$recall = \frac{|\,true\ positives\,|}{|\,true\ positives\,| + |\,false\ negatives\,|}$$

Since there is generally a trade-off between *precision* and *recall*, it can be useful to combine them into a third score called *f-measure* (Manning/Schütze 1999:269):

(16) $$f-measure = \frac{1}{\alpha \frac{1}{precision} + (1-\alpha)\frac{1}{recall}}$$

where α ($0 \leq \alpha \leq 1$) is a constant factor that weights precision and recall equally if set to 0.5.

A less widely used metric is *fallout*, the proportion of items mistakenly selected (Manning/Schütze 1999:270):

(17) $$fallout = \frac{|false\ positives|}{|false\ positives| + |true\ negatives|}$$

Soergel (1999:pdf:4) suggests a further measure, *discrimination*, which is the complement of *fallout*. *Discrimination* is the fraction of irrelevant items correctly rejected. However, *discrimination* is hardly ever used in the evaluation of retrieval systems.

(18) $$discrimination = \frac{|true\ negatives|}{|false\ positives| + |true\ negatives|}$$

In the context of automatic document linking, *true* and *false positives* or *negatives* respectively correspond to correctly or incorrectly generated or non-generated hyperlinks, as schematized in Table 1:

	relevant link	irrelevant link
generated	*true positive*	*false positive*
not generated	*false negative*	*true negative*

Table 1: possible results in retrieval systems

Still, the main problem in the evaluation of retrieval systems is to determine what is relevant and what is not, especially since relevance is a notion eventually depending on the individual user. The classical strategy for relevance estimation is thus to employ human assessors or to take a human-evaluated reference corpus as the comparison standard. The current study is no exception.

3. An Experiment

This chapter will describe an experiment in which a set of key n-grams extracted from a sample of German Wikipedia articles on the basis of a high *ridf* value was used to establish hyperlinks on the same sample, such that two documents were connected if they contained at least one common key term. Link generation was performed by a program with the working title `CrossRef`. In particular, innercategorial hyperlinks have been created for the Wikipedia categories *Verschwörungstheorie* ('conspiracy theory', henceforth occasionally abbreviated *VT*), *Bibel* ('Bible', henceforth occasionally abbreviated *B*), *Handball* ('handball'), *Privatrecht* ('civil law') and *Sexualität* ('sexuality'). Intercategorial links have been generated between documents of the category *VT* and the category *B*. All created link structures have been compared to the Wikipedia gold-standard. A subset has been evaluated manually.

3.1. Aim and Expectations

The goal of the experiment was to explore whether hyperlinks created between non-similar but *intersecting* documents can be relevant. Given the linking strategy described above, the prior prospect was that – on the premise of prioritized *recall* – a minor proportion of the created links would be non-distinct from those originating from *similarity*-based linking methods or human creation. A second extensive portion of the resultant links was simply expected to be *noise*. Thus, the target was to confirm the existence of a third fraction of links that were potentially relevant but not envisioned in traditional linking ideologies.

3.2. Design Decisions and Setup

3.2.1. The Corpus

The test corpus consisted of a set of German Wikipedia articles spanning several categories, namely *Verschwörungstheorie* ('conspiracy theory'), *Bibel* ('Bible'), *Handball* ('handball'), *Privatrecht* ('civil law') and *Sexualität* ('sexuality') [42].

---

[42] The sample set of Wikipedia articles has been downloaded via the Wikipedia API developed at the TU Darmstadt (http://www.ukp.tu-darmstadt.de/software/WikipediaAPI). An alternative tool for downloading Wikipedia articles is the module WWW::Wikipedia available at the CPAN (http://search.cpan.org/~bricas/WWW-Wikipedia-1.92/lib/WWW/Wikipedia.pm).

Wikipedia (http://wikipedia.org) is a free multiauthor online encyclopedia[43]. It was chosen as reference corpus, because it constitutes a free repository of pertinent texts, and comparison link *graphs*. With respect to its contents, Wikipedia was especially suitable for the `CrossRef` experiment, because articles are grouped into categories[44], thus providing access to heterogeneous samples of different subjects. Moreover, Wikipedia hyperlinks are created manually, which allows for the inference that all of them express a relevant relation. More formally speaking, most Wikipedia links correspond to what Allan (1996) calls *pattern-matching* links, connecting a term to an article that explains the same term (see Appendix C, Table 9). As a network, Wikipedia corresponds to a *mixed graph* (see Appendix D, Table 10) presumably exhibiting *small-world* properties (see section 2.3) (Voss 2005:pdf:61).

3.2.2. Selection of Distinctive Phrases

The `CrossRef` experiment employed a very lax conception of a distinctive phrase, namely that of a simple n-gram with an *ridf* value beyond a certain threshold (*ridf* ≥ 1.5 for the link structures evaluated manually)[45]. Hence `CrossRef` neglected syntactic constituent structure and morphological variation. This relinquishment of expensive preprocessing such as *stemming* or *POS*-tagging has chiefly been due to a limitation of time and resources, and not to any theoretical considerations. Originally, it has been intended to identify coherent key phrases by means of a high *mi*-value as a second criterion besides a high *ridf*-value. However, *MI*, as defined in (11), mostly picked combinations of a lexical category with a functional one such as a noun phrase and a determiner (see Appendix E.c, Table 17 for examples). Therefore, the venture to combine *RIDF* and *MI* has been abandoned at a very early stage[46]. Since functional categories were considered harmless, as long as combined with a lexical category, they were eventually admitted to the key n-gram sequences. After all, markup chunks were removed from the corpus and

---

[43] For a detailed description of Wikipedia see Voss (2005).

[44] The category system of Wikipedia can be regarded as a kind of thesaurus, where a category is realized as a keyword, manually associated with a set of articles. Wikipedia categories form a hierarchy, though not a very strict one (Voss 2005:pdf:22,51,64).

[45] A further practical restriction was a *df* ≥ 2. Obviously, terms appearing in only one document are useless for *intersection*-based linking of the same collection.

[46] It has also been tried to use *MI* as a filter to reduce lexicon size. However, *MI* also selected for valuable items such as compounds and proper names. Therefore, the issue has been dispensed with.

particularly *noisy* lexical categories were designated as stopwords[47]. Moreover, n-gram length was restricted to values between 2 and 5. The reason to require at least bigrams consisted in the fact that a lot of bad links were induced by the ambiguous first names of persons.

3.2.3. Advantages of RIDF

The main reason to choose *RIDF* instead of the standard *TF.IDF* as the term-weighting scheme for keyword-extraction was that the term-weights calculated via *TF.IDF* are always related to a particular document. As already indicated in section 2.1.2.3, the primary purpose of the *tf* factor in *TF.IDF* is to grade terms as better keywords for particular subsets of documents than for other subsets, which is the ideal behaviour if the measurement of interest concerns *semantic similarity*. However, as pointed out earlier, this study doesn't require documents to be *semantically similar*, but rather to *intersect* at some point. Mere *intersection* does not presuppose that the common key terms are salient in the respective documents. Thus, for purposes of the `CrossRef` experiment, *RIDF* seemed conceptually superior to *TF.IDF*, because it derives keywords from the entire collection rather than from individual documents. Moreover, the parameter frequencies for *RIDF*, are easy to obtain (see also section 3.2.4).

3.2.4. Implementation Details

Key term extraction and hyperlink generation have been performed by a Perl script called `CrossRef.pm`, which implemented a suffix array as described by Yamamoto and Church (1998) to calculate the parameter frequencies for *RIDF* – *collection frequency* and *document frequency* – from corpus data. With a suffix array, all $C(C+1)/2$ substrings contained in a corpus can be grouped into a convenient set of equivalence classes, thus facilitating a calculation of frequencies in $O(C \cdot \log C)$ time (Yamamoto/Church 1998:pdf1,3). Other implementation features are trivial. The complete source-code is given in Appendix F[48].

---

[47] In fact, the only domain-neutral stopwords were *ISBN* and *Kategorie*.

[48] Besides the code presented in Appendix F, there has been a Java program to obtain the Wikipedia data (see Footnote 42) and a Visual Basic macro to compute the final scores of human evaluation.

3.2.5. Evaluation

The evaluation of experiment results took place in two phases. In a first step, the link structure created by `CrossRef` was compared to a derivate of the original Wikipedia link *graph*, in which all potentially reproducable hyperlinks had been converted into symmetric *edges*[49]. For this comparison, *true* and *false positives* and *negatives* were defined as in Table 2:

	present in Wikipedia	not present in Wikipedia
`CrossRef` generated	*true positive*	*false positive*
not `CrossRef` generated	*false negative*	*true negative*

Table 2: result specification nr. 1 for `CrossRef` evaluation

In a second phase, a subset of the generated link structures has been re-evaluated manually. Four test persons were asked to asses the links generated on the domain *VT*[50] previously classified as *false positives* and two persons evaluated the set of *false positives* generated between the domain *VT* and *B*. In particular, assessors were asked to tag the link with a *1* in case it met two requirements:

(19) At least one of the linking keywords refers to the same entity, concept or circumstance in both documents.

(20) There is a potentially relevant relation between the two documents.

Otherwise, the link was to be tagged with a *0*. All items tagged with a *1* were subsequently subtracted from the *false positives* and apportioned to the *true positives*. Since manual evaluation was considerably time-consuming, the assessors were not forced to assess all of the questionable links. Hence, the results of phase two are percental projections based on the definitions in Table 3:

	present in Wikipedia or manually assessed relevant	not present in Wikipedia or manually assessed irrelevant
`CrossRef` generated	*true positive*	*false positive*
not `CrossRef` generated	(*false negative*)	(*true negative*)

Table 3: result specification nr. 2 for `CrossRef` evaluation

---

[49] Given that the linking strategy described above produces *undirected graphs*, it seemed reasonable to compare the output to another *undirected graph*. From a theoretical point of view, this means that also incoming links are considered to express relevant relations, which is entirely consistent with the philosophy put forward here.

[50] This category was chosen because it is thematically one of the domains in which non-obvious relations are potentially interesting. Moreover, it was considered more entertaining than other candidate categories. Finally, with a total of 143 articles, it appeared to have a manageable size.

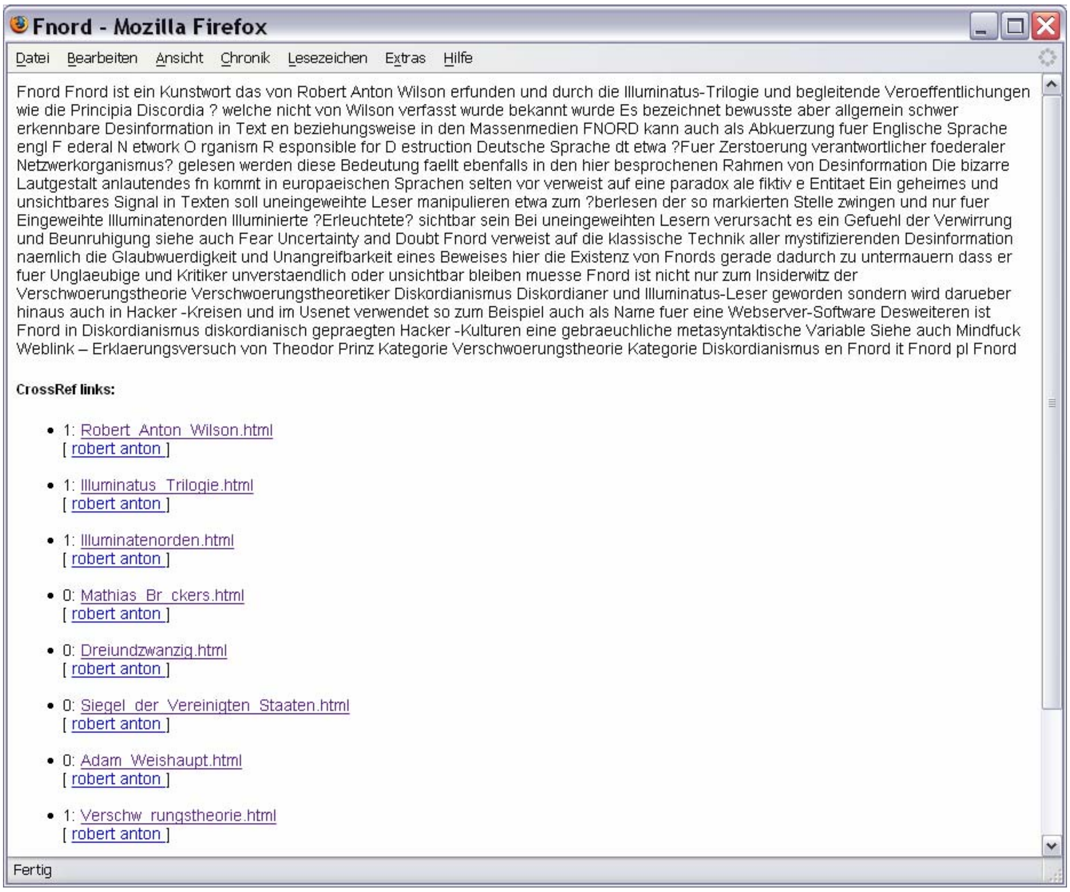

Figure 4: HTML layout of a `CrossRef`-linked article for human evaluation[51]

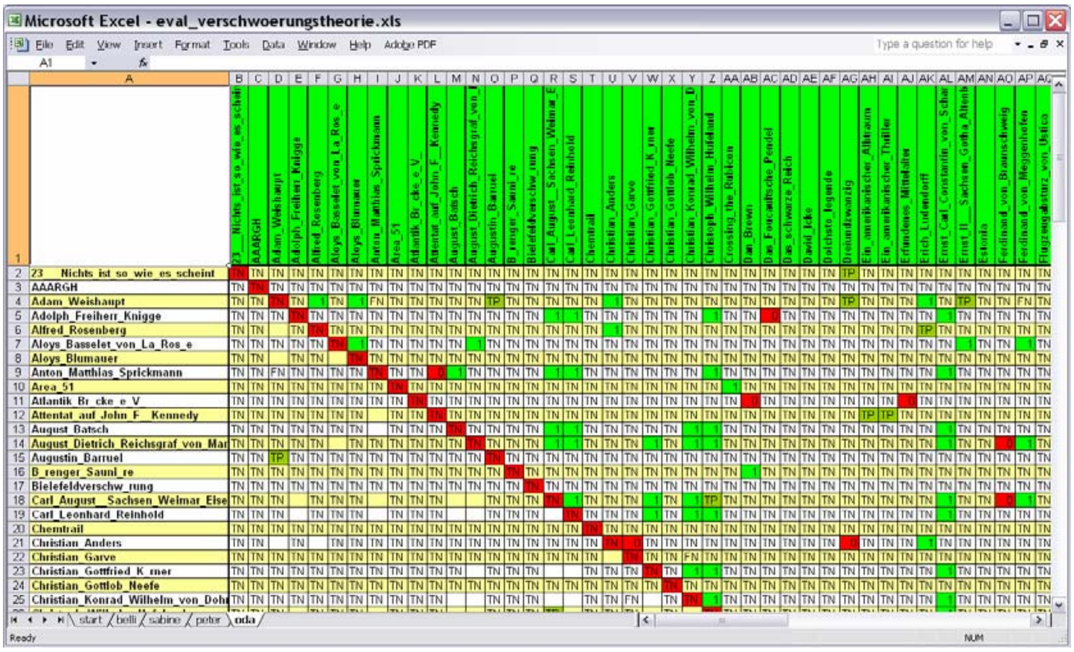

Figure 5: excerpt from an evaluation sheet[52]

[51] In Figure 4, the article text is followed by a set of links, followed by a list of linking keywords, here *Robert Anton*, wich refers to the same *Robert Anton Wilson* in all the linked articles. A *1* preceding the link indicates that the link is present in Wikipedia, a *0* indicates absence in Wikipedia.

[52] Figure 5 shows an evaluation sheet, where TP stands for *true positive*, TN for *true negative* and FN for *false negative*. The fields containing numbers correspond to the re-evaluated *false positives*.

3.3. Results and Discussion

Not surprisingly, as direct functions of *RIDF*, *precision* and *recall* turned out to be highly converse, such that low *ridf* favoured *recall* while high *ridf* preferred *precision*. In the comparison to Wikipedia, the combined *f-measure* was always very low, as exemplarily illustrated in Figure 6 and in Table 4 (see Appendix E for more result details).

	B 'Bible'	*Handball* 'handball'	*Privatrecht* 'civil law'	*Sexualität* 'sexuality'	*VT* 'conspiracy theory'	*VT + B* 'conspiracy theory' + 'Bible'
best *f-value*	0.14	0.34	0.15	0.17	0.40	0.13
ridf	1.3	0.9	1.3	1.6	2.2	1.3
number of documents *D*	965	204	1949	1752	143	1108

Table 4: best *f*-values of `CrossRef` link structures in comparison to Wikipedia link *graphs*

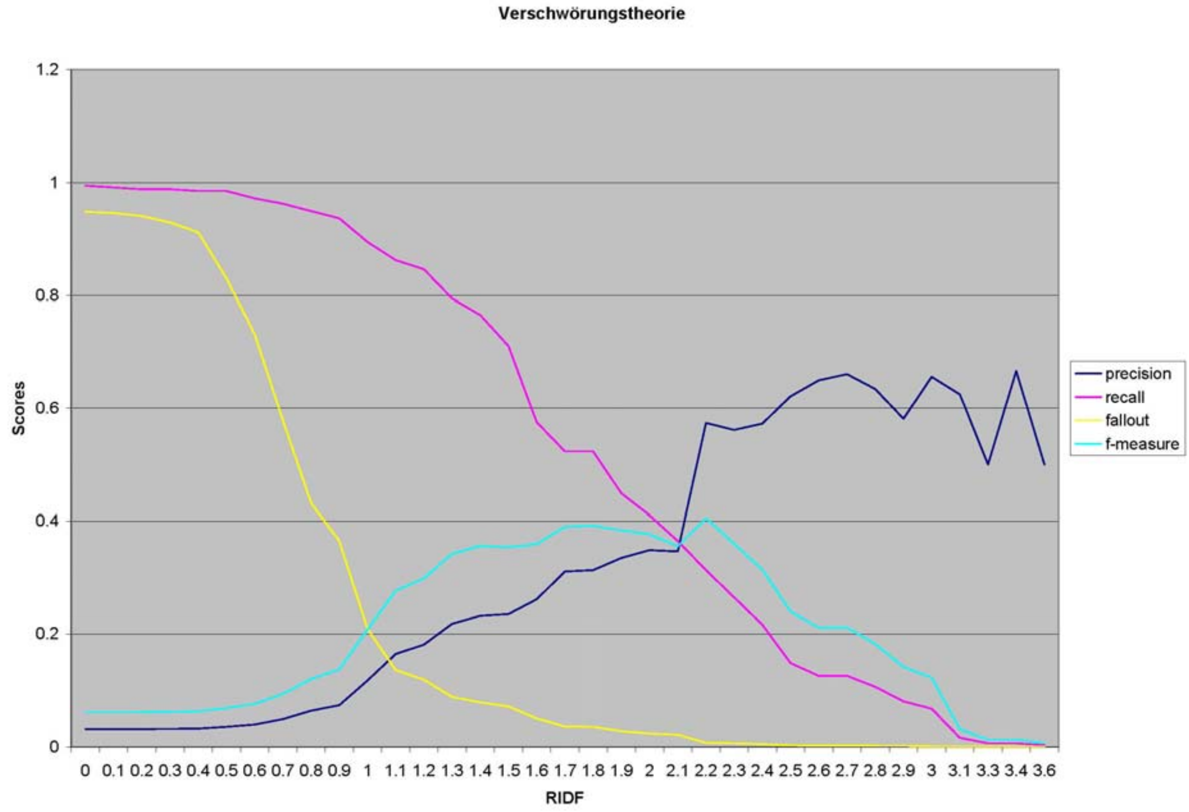

Figure 6: Wikipedia-related scores for distinct *ridf* values of (category *VT*)

If the original Wikipedia link *graphs* are taken to be the exclusive gold-standard for link quality, the values in Table 4 are extremely poor. However, as stated earlier, the main goal of the `CrossRef` experiment was not to approximate the original *graphs* but rather to discover supplementary links that might be relevant to a subgroup of users. For this reason, the links generated on the category *VT* and the links between the categories *VT* and *B* have been reassessed manually. Comprehensive trials as in Figure 6 suggested an *ridf* value of 1.5 for link creation for human assessment. With *ridf* = 1.5, the category *VT* still had a considerably high *recall*, leaving a sufficient amount of *false positives* for manual re-evaluation. At the same time, the *f-measure* was slightly below its maximum, indicating a high optimality degree for *precision* and *recall*[53]. For the sake of uniformity, this value has also been adopted for link generation bridging *VT* and *B*. The results of human re-evaluation are presented in Table 5:

	false *false positives*[54]	re-evaluated *precision*	(re-evaluated *recall*)[55]	Wikipedia-related *precision*	Wikipedia-related *recall*
VT	40.8%	54.1%	(85.0%)	23.6%	71.1%
VT – B[56]	29.9%	32.7%	(80.3%)	4.1%	33.9%

Table 5: results of human reassessment of Wikipedia related *false positives*

Though far away from vindicating the `CrossRef` linking method as qualified for real-world applications, the figures in Table 5 strongly suggest that there are latent but interesting relations besides those encoded manually in form of a hyperlink (usually involving high document *similarity*[57]). A sample of beneficial `CrossRef` links is given in Table 6:

[53] Random samples across categories indicated that choosing *ridf* according to maximal Wikipedia approximation was not a bad idea, after all. Notably, with respect to the *ridf* producing the best *f-score* (weighting precision and *recall* equally), the category *VT* diverges from the other tested categories which mostly exhibit a maximal *f-value* for *ridf* ≈ 1.5. *VT* reaches its best f-value at *ridf* = 2.2. Figure 6 indicates that this phenomenon might be due to the elimination of a few extremely productive key terms between *idf* = 2.2 and *idf* = 2.3.

[54] The term 'false *false positives*' here designates *false positives* manually classified as relevant.

[55] Since the portion of *true negatives* has not been inspected manually, the reassessed *recall* values in Table 5 do not reflect on potential 'false *true negatives*'. Hence, these values are upper bounds and not necessarily appropriate empirically.

[56] Recall that for *VT – B,* human evaluation has been restricted category-bridging links.

[57] Almost all inspected `CrossRef` links that were motivated by several common key terms conformed to links in the original. Neglecting the original link directionality, this fact allows for the conclusion that, in Wikipedia, links coming along with high document similarity roughly form a subset of the manually created links. `CrossRef` links absent from Wikipedia thus usually revealed low *similarity*.

domain	article 1	article 2	linking term
B (*ridf* = 1.3)	*Gewalt in der Bibel* ('violence in the Bible')	*Opferung Isaaks* ('sacrifice of Isaac')	*Brandopfer* ('burnt offering')
Handball (*ridf* = 0.9)	*Iñaki Urdangarin* (Basque handball player)	*Olympische Sommerspiele 1996 Handball* ('Olympic Summer Games 1996 handball')	*Olympische Sommerspiele 1996* ('Olympic Summer Games 1996')
Privatrecht (*ridf* = 1.3)	*Shimpū Tokkōtai* (Kamikaze troop in World War II)	*Selbstmordattentat* ('suicide assassination')	*Tokkōtai*
Sexualität (*ridf* = 1.6)	*Lymphopatia Venerea* (venereal disease)	*Epididymitis* (other venereal disease)	*Chlamydia Trachomatis* (bacteria species)
VT (*ridf* = 1.5)	*UFO Absturz von Roswell* ('UFO crash of Roswell')	*Reichsflugscheibe* (mythical flying saucer built by the National Socialists in the Third Reich)	*fliegende Untertasse* ('flying saucer')
VT – B (*ridf* = 1.5)	*Bibelcode* ('Bible code')	*Attentat auf John F. Kennedy* ('assassination of John F. Kennedy')	*John F. Kennedy*

Table 6: examples of interesting `CrossRef` links

But what about all the noise? Is there a prototype for good keywords and respectively one for bad keywords? The human assessors consistently reported that the valid key terms were almost exclusively noun phrases, predominantly proper names, denoting very concrete entities. A superficial survey of the key term data indicated that good key n-grams were approximately those that exhibited a high degree of *individuation*, defined by Hopper and Thompson (1980) as a parameter of cross-linguistic *transitivity*[58], namely the distinctness of objects. The parameters of *individuation* are summarized in Table 7:

individuated	non-individuated
proper	common
human, animate	inanimate
concrete	abstract
count	mass
referential, definite	non-referential

Table 7: parameters of *individuation* (Hopper/Thompson 1980:252-253)

[58] "Transitivity is traditionally understood as a global property of an entire clause, such that an activity is 'carried-over' from an agent to a patient" (Hopper/Thompson 1980:251). Notably, *transitivity* is frequently marked syntactically or morphologically across languages.

On the other hand, massive pollution was induced by ambiguous fragments of proper names such as *Theodor von* or *John F.* and by terms that did not actually belong to the article text itself but rather to bibliographical references or frame elements. Crucially, a very small fraction of terms such as *historisch-kritische* ('historic-critical') or *v. Chr* ('B.C.') caused a major portion of undesirable links.

The above observations thus imply that the `CrossRef` linking strategy, as it stands, is insufficient, but that a lot of ground could be covered by a supplementation with standard subsidiary NLP procedures, such as *named entity recognition*, (partial) *parsing*, *stopword* elimination or *lemmatization*, to make sure key phrases are unambiguous noun phrases (or at least cohesive syntactic constituents) and to obtain inflection-neutral term-weights. Another promising modification could consist in the normalization of term weights by n-gram length to corroborate less ambiguous longer strings. Alternatively, totally different extraction methods, as f.e. the one by Tomokiyo and Hurst (2003), clearly deserve a trial.

Finally, the test persons unanimously appreciated the display of the linking key phrases along with the link itself, testifying that there was a strong correlation between key term relevance and link relevance. Hence, the linking key terms qualified as an effective device to estimate the a priori link relevance.

4. Conclusions

This essay has described an experimental method of document linking based on document *intersection*, where documents were linked in case they had a common key term, determined by a high *ridf* value. The method has been evaluated both against a gold-standard and manually. The outcome has been twofold: It has been shown that a lot of potentially interesting relations can be detected, even if the connected documents do not exhibit a high degree of *spatial similarity*. On the other hand, the method employed has turned out to be premature for practical applications, due to a great amount of *noise*. With respect to the linking key terms themselves, results suggest that key n-grams dedicated to document linking by mere *intersection* should comply with distinct requirements than key phrases dedicated to similarity-based document linking: While keywords intended for the latter should primarily reflect on document content as a whole, keywords qualified for the former should designate concrete, unambiguous and highly *individuated* entities that manifest interesting and possibly document-independent subtopics themselves. Finally, it has been confirmed that the individual relevance of an *intersection*-based hyperlink is transparent to the user if the linking key term is exposed along with the link.

In sum, this study concludes that, besides document linking based on *spatial similarity*, there is room for a supplementary possibly dynamic document linking method based on document *intersection*, intended to detect non-obvious connections, relevant for any subgroup of users interested in text relations beyond the surface.

References

Bibliography

AGOSTI, Maristella / CRESTANI, Fabio / MELUCCI, Massimo (1997). "On the Use of Information Retrieval Techniques for the Automatic Construction of Hypertext". In: *Information Processing & Management*. Vol. 33, No. 2, p.133-144. http://www.cs.strath.ac.uk/~fabioc/papers/97-ipem.ps (03/2007).

ALLAN, James (1996): "Automatic Hypertext Link Typing". In: *Proceedings of the Seventh ACM Conference on Hypertext*. P. 42-52. http://typhon.perseus.tufts.edu/typhon/Flashy/Documents/Documents.current/DL%20Notes/allan.dl.96.pdf (03/2007).

ALLAN, James (1997). "Building Hypertext Using Information Retrieval". In: *Information Processing & Management*. Vol. 33, No./Issue 2, p. 145-159. http://www.sciencedirect.com/science?_ob=MImg&_imagekey=B6VC8-3SWVHFP-3-2&_cdi=5948&_user=1634476&_orig=search&_coverDate=03%2F31%2F1997&_sk=999669997&view=c&wchp=dGLbVzW-zSkzV&md5=4ddb01cf51cee830d491066498034 22c&ie=/sdarticle.pdf (03/2007).

BAEZA-YATES, Ricardo / RIBEIRO-NETO, Berthier (1999). *Modern Information Retrieval*. Harlow, England: Addison-Wesley Longman.

BAILEY, Christopher / EL-BELTAGY, Samhaa R. / HALL, Wendy (2001). "Link Augmentation: A Context-Based Approach to Support Adaptive Hypermedia". In: *Proceedings of Hypermedia: Openness, Structural Awareness, and Adaptivity*. Vol. 2266, p. 239-251. Århus, Denmark. http://wwwis.win.tue.nl/ah2001/papers/bailey.pdf (03/2007).

BARABÁSI, Albert-Lázló / ALBERT, Reka / JEONG, Hawoong (2000). "Scale-Free Characteristics of Random Networks: The Topology of the World Wide Web". In: *Physica A*. Vol. 281, p. 69-77. http://www.cogs.sussex.ac.uk/users/masters/easymsc2003/wh21/Barbarasi,%20Albert,%20Jeong%20-%20Scale%20free%20characteristics%20of%20random%20networks...ps (03/2007).

BELLOMI, Francesco / CRISTIANI, Matteo (2006). "Supervised Document Classification Based upon Domain-Specific Term Taxonomies". In: *International Journal of Metadata, Semantics and Ontologies* . Vol. 1, No.1, p. 37-46. http://www.inderscience.com/filter.php?aid=87 (03/2007).

BRODER, Andrei / KUMAR, Ravi / MAGHOUL, Farzin / RAGHAVAN, Prabhakar / RAJAGOPALAN, Sridhar / STATA, Raymie / TOMKINS, Andrew / WIENER, Janet (2000). "Graph Structure in the Web". In: *Proceedings of the 9th International World Wide Web Conference on Computer Networks: The International Journal of Computer and Telecommunication Networking*. P. 309-320. http://www9.org/w9cdrom/160/160.html (03/2007).

CAMACHO-GUERRERO, José Antonio / MACEDO, Alessandra Alaniz / PIMENTEL, Maria da Graca Campos (2004). "A Look at Some Issues During Textual Linking of Homogeneous Web Repositories". In: *Proceedings of the 2004 ACM Symposium on Document Engineering.* http://portal.acm.org/ft_gateway.cfm?id=1030413&type=pdf&coll=GUIDE&dl=GUIDE&CFID=12954610&CFTOKEN=30853434 (03/2007).

CIMIANO, Philipp (2006). *Ontology Learning and Population from Text – Algorithms, Evaluation and Applications*. Springer Science+Business Media.

CLEARY, Chip / BAREISS, Ray (1996). "Practical Methods for Automatically Generating Typed Links". In: *Proceedings of the Seventh ACM Conference on Hypertext*. P. 31-41. http://typhon.perseus.tufts.edu/typhon/Flashy/Documents/Documents.current/DL%20Notes/cleary.96.pdf (03/2007).

CHURCH, Kenneth Ward / Hanks, Patrick (1990). "Word Association Norms, Mutual Information and Lexicography". In: *Proceedings of the 27th Annual Meeting on Association for Computational Linguistics*. P. 76-83. http://acl.ldc.upenn.edu/J/J90/J90-1003.pdf (03/2007).

CROFT, W. B. / HARPER, D.J. (1979). "Using Probabilistic Models of Document Retrieval without Relevance Information". In: Ed.: SPÄRCK JONES, Karen / WILLETT, Peter (1997). *Readings in Information Retrieval*. P. 339-344. San Francisco, USA: Morgan Kaufmann.

EL-BELTAGY, Samhaa R. / HALL, Wendy / DEROURE, David / CARR, Leslie (2001). "Linking in Context". In: *Proceedings of the Twelfth ACM Conference on Hypertext and Hypermedia*. http://www.sigweb.org/papers/ElBeltagy.pdf (03/2007).

GOFFINET, Luc / NOIRHOMME-FRAITURE Monique (1995). "Automatic Hypertext Link Generation Based on Similarity Measures between Documents". *Research Paper, RP-96-034,* Namur, Belgium: Institut d'Informatique, FUNDP. http://perso.fundp.ac.be/~lgoffine/Hypertext/semantic_links.html (03/2007).

GOLOVCHINSKY, Gene (1997). "Queries? Links? Is There a Difference?" In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. P. 407-414. http://www.cindoc.csic.es/cybermetrics/pdf/159.pdf (03/2007).

GREEN, Stephen J. (1998). "Automated Link Generation: Can We Do Better Than Term Repetition?". In: *International World-Wide Web Conference*. P. 75-84. Brisbane, Australia. http://ftp.cs.toronto.edu/pub/gh/Green-98.pdf (03/2007).

HEYER, Gerhard / QUASTHOFF, Uwe / WITTIG, Thomas (2006). *Text Mining: Wissensrohstoff Text.* Bochum: W3L GmbH.

HOPPER, Paul J. / THOMPSON, Sandra (1980). “Transitivity in grammar and discourse”. In: *Language*. Vol. 56, p. 251-299.

JURAFSKY, Daniel / MARTIN, James H. (2000). *Speech and Language Processing – An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition*. New Jersey: Prentice Hall.

LANGVILLE, Amy N. / MEYER, Carl D. (2006). *Google's Page Rank and Beyond – The Science of Search Engine Rankings*. New Jersey, USA: Princeton University Press.

LEWIS, Paul H. / HUGH, C. Davis, / GRIFFITHS, Steve R. / HALL, Wendy / WILKINS, Rob J. (1996). "Media-Based Navigation with Generic Links". In: *Proceedings of the Seventh ACM Conference on Hypertext*. P. 215-223. http://eprints.ecs.soton.ac.uk/797/05/html/ (03/2007).

MACEDO, Alessandra Alaniz / CAMACHO-GUERRERO, Jose Antonio / PIMENTEL, Maria da Graca (2002). "An Infrastructure for Open Latent Semantic Linking". In: *Proceedings of the Thirteenth ACM Conference*. http://portal.acm.org/ft_gateway.cfm?id=513369&type=pdf&coll=GUIDE&dl=portal,ACM&CFID=16084462&CFTOKEN=39931354 (03/2007).

MANNING, Christopher D. / SCHÜTZE, Hinrich (1999). *Foundations of Statistical Natural Language Processing*. Cambridge, Massachusetts / London, England: MIT Press.

MCNAMEE, Paul / MAYFIELD, James (2001). "JHU/APL Experiments at CLEF: Translation Resources and Score Normalization". In: *Proceedings of the CLEF 2001 Cross-LanguageText Retrieval System Evaluation Campaign*. Springer-Verlag. http://www.ercim.org/publication/ws-proceedings/CLEF2/mcnamee.pdf (03/2007).

MONTEJO-RÁEZ, Arturo / STEINBERGER, Ralf (2004). "Why Keywording Matters". In: *HEP Libraries Webzine.* Issue 10. http://library.cern.ch/HEPLW/10/papers/2/ (03/2007).

MORVILLE, Peter (2005). *Ambient Findability – What We Find Changes Who We Become*. Sebastopol, CA: O'Reilly.

NANAS, Nikolaos / UREN, Victoria / DEROECK, Anne (2003). "A Comparative Study of Term Weighting Methods for Information Filtering". In: *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. http://kmi.open.ac.uk/publications/pdf/kmi-03-4.pdf (03/2007).

RENNIE, Jason / JAAKKOLA, Tommi (2005). "Using Term Informativeness for Named Entity Detection". In: *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. http://people.csail.mit.edu/tommi/papers/RenJaa-sigir05.pdf (03/2007).

ROBERTSON, Stephen (2004). "Understanding Inverse Document Frequency: On Theoretical Arguments for IDF". In: *Journal of Documentation*. Vol. 60, No. 5, p. 503-520. MCB University Press. http://www.soi.city.ac.uk/~ser/idfpapers/Robertson_idf_JDoc.pdf (03/2007).

SALTON, Gerard / BUCKLEY, Christopher (1988). "Term-Weighting Approaches in Automatic Text Retrieval". In: *Information Processing & Management.* Vol. 24, No. 5, p. 513-523. http://www.sciencedirect.com/science?_ob=MImg&_imagekey =B6VC8-469WV05-11&_cdi=5948&_user=1634476&_orig=search&_coverDate= 12%2F31%2F1988 &_sk=999759994&view=c&wchp=dGLbVtbzSkWW&md5 =032407b57ceb 1e55fb8f33034888bead&ie=/sdarticle.pdf (03/2007).

SALTON, Gerard (1989). *Automatic Text Processing – The Transformation, Analysis, and Retrieval of Information by Computer*. Reading, USA: Addison-Wesley.

SALTON, GERARD / ALLAN, JAMES (1993). "Selective Text Utilization and Text Traversal". In: *Proceedings of ACM Hypertext*. P. 131-144. http://www-ciir.cs.umass.edu/~allan/Papers/1993-hypertext.ps (03/2007).

SALTON, Gerard / ALLAN, James / BUCKLEY, Chris (1994). "Automatic Structuring and Retrieval of Large Text Files". In: *Communications of the ACM*. Vol. 37, No./Issue 2, p. 97-108. http://portal.acm.org/ft_gateway.cfm?id=175243&type=pdf&coll=GUIDE&dl=GUIDE&CFID=12954356&CFTOKEN=65231581 (03/2007).

SHANNON, Claude (1948). "A Mathematical Theory of Communication". In: *Bell System Technical Journal.* Vol. 27. http://cm.bell-labs.com/cm/ms/what/shannonday/shannon1948.pdf (03/2007).

SOERGEL, Dagobert (1999). "Indexing and Retrieval Performance: The Logical Evidence". In: *Journal of the American Society for Information Science*. Vol. 45, No./Issue 8, p. 589-599. http://www.dsoergel.com/cv/B46.pdf (03/2007).

SPÄRCK JONES, Karen (2004). "A Statistical Interpretation of Term Specificity and its Application in Retrieval". In: *Journal of Documentation*. Vol. 60, No. 5, p. 493-502. MCB University Press. http://www.soi.city.ac.uk/~ser/idfpapers/ksj_orig.pdf (03/2007).

STEYVERS, Mark / TENENBAUM, Joshua B. (2005). "The Large-Scale Structure of Semantic Networks: Statistical Analyses and a Model of Semantic Growth". In: *Cognitive Science*. http://web.mit.edu/cocosci/Papers/03nSteyvers.pdf (03/2007).

TEBUTT, John (1998). "Finding Links". In: *Proceedings of the Ninth ACM Conference on Hypertext*. ACM. http://www-nlpir.nist.gov/works/papers/finding_links.html (03/2007).

TOMOKIYO, Takashi / HURST, Matthew (2003). "A Language Model Approach to Keyphrase Extraction". In: *Proceedings of the ACL Workshop on Multiword Expressions: Analysis, Acquisition and Treatment*. P. 33-40. http://acl.ldc.upenn.edu/acl2003/mwexp/pdfs/Tomokiyo.pdf (03/2007).

VOSS, Jacob (2005). "Informetrische Untersuchungen an der Online-Enzyklopädie Wikipedia". http://jakobvoss.de/magisterarbeit/MagisterarbeitJakobVoss.pdf (03/2007).

WATTS, Duncan J. / Strogaz, Steven H. (1998). "Collective Dynamics of Small-World Networks". In: *Nature*. Vol. 393, p. 440-442. http://www.tam.cornell.edu/SS_nature_smallworld.pdf (03/2007).

WATTS, Duncan J. (1999). *Small Worlds – The Dynamics of Networks between Order and Randomness*. Chichester, England: Princeton University Press.

WILKINSON, Ross / SMEATON, Alan F. (1999). "Automatic Link Generation". In: *ACM Computing Surveys (CSUR)*. Vol. 31, No. 4. http://typhon.perseus.tufts.edu/typhon/Flashy/Documents/Documents.current/DL%20Notes/wilkinson.acm.1999.pdf (03/2007).

YAMAMOTO, Mikio / CHURCH, Kenneth Ward (1998). "Using Suffix Arrays to Compute Term Frequency and Document Frequency for All Substrings in a Corpus". In: *Proceedings of ACL Workshop on Very Large Corpora*. P. 28-37. Montreal. http://acl.ldc.upenn.edu/J/J01/J01-1001.pdf (03/2007).

ZENG, Jihong / BLONIARZ, Peter A. (2004). "From Keywords to Links: An Automatic Approach". In: *International Conference on Information Technology: Coding and Computing (ITCC'04)*. Vol. 1. http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1286467 (03/2007).

Examples

Texts

CAMUS, Albert (1956). *Der Mythos von Sisyphos — Ein Versuch über das Absurde.* Düsseldorf: Karl Rauch Verlag GmbH.

CARPENTIER, Alejo (1985). *Los pasos perdidos.* Madrid: Cátedra. http://www.lms.uchile.cl/PRINCIPAL/planlector/IV%BA%20Medio/Carpentier,%20Alejo%20-%20Los%20pasos%20perdidos.doc (03/2007).

PEZZELLA, Daniel (2006). "Las mujeres y su función en Los pasos perdidos, de Alejo Carpentier". http://www.cienciared.com.ar/ra/usr/10/177/hln1.pdf (03/2007).

SPADAFORI, Gina (2001). *Dogs for Dummies*. New York, USA: Hungry Minds. http://www.amazon.com/Dummies-Miniature-Editions-Running-Press/dp/076241362X/ref=pd_bbs_sr_1/102-1499838-4006546?ie=UTF8&s=books&qid=1173094595&sr=8-1 (03/2007).

Websites

http://www.amazon.com/phrase/canine-competitions/ref=sip_bod_2/102-1499838-4006546 [Amazon books linked by a *SIP*] (03/2007).

http://www.krone.at/index.php?http://wcm.krone.at/krone/C12/S22/A7/object_id__10152/hxcms/ [yellow press article] (03/2007).

Perl Textbooks

CONWAY, Damian (2005). *Perl – Best Practices*. Sebastopol, CA: O'Reilly.

COZENS, Simon (2005). *Advanced Perl Programming, Second Edition.* Sebastopol, CA: O'Reilly.

SCHWARTZ, Randal L. / PHOENIX, Tom / FOY, Brian D. (2005). *Einführung in Perl*. Köln: O'Reilly.

SCHWARTZ, Randal L. / PHOENIX, Tom (2004). *Perl – Objekte, Referenzen & Module*. Köln: O'Reilly.

Further Weblinks

Amazon

http://www.amazon.com (03/2007).
http://www.amazon.com/gp/search-inside/sipshelp.html (03/2007).

CPAN

http://www.cpan.org/ (03/2007).
http://search.cpan.org/~bricas/WWW-Wikipedia-1.92/lib/WWW/Wikipedia.pm (03/2007).

Google

http://www.google.com/ (03/2007).

The IDF page

http://www.soi.city.ac.uk/~ser/idf.html (03/2007).

Wikipedia

http://en.wikipedia.org/wiki/Graph_%28mathematics%29 (03/2007).
http://en.wikipedia.org/wiki/Hypermedia (03/2007).
http://en.wikipedia.org/wiki/Lemmatization (03/2007).
http://en.wikipedia.org/wiki/Lexical_chain (03/2007).
http://en.wikipedia.org/wiki/Part_of_speech_tagging (03/2007).
http://en.wikipedia.org/wiki/Polysemy (03/2007).
http://en.wikipedia.org/wiki/Power_law (03/2007).
http://en.wikipedia.org/wiki/Stemming (03/2007).
http://en.wikipedia.org/wiki/Synset (03/2007).
http://es.wikipedia.org/wiki/Treno (03/2007).
http://en.wikipedia.org/wiki/WordNet (03/2007).
http://wikipedia.org (03/2007).

Wikipedia API (TU Darmstadt)

http://www.ukp.tu-darmstadt.de/ (03/2007).
http://www.ukp.tu-darmstadt.de/software/WikipediaAPI (03/2007).

WordNet

http://wordnet.princeton.edu/ (03/2007).

Appendices

A. Citation Conventions Notation and Abbreviations

A.a Citation conventions

Since most of the scientific papers referenced here have been retrieved as electronic versions, citations frequently refer to electronic documents. This fact is emphasized by specifying the file type of the electronic documents in the respective citations. A citation such as 'Steyvers and Tenenbaum (2005:pdf:4-9)' refers to the PDF-version of the article. In a citation such as '(Croft/Harper 1979:340)', on the other hand, page numbers refer to the printed version.

A.b Notation and abbreviations

For the reader's convenience, a subset of abbreviations is used consistently with a constant meaning throughout the paper. These abbreviations are listed in Table 8:

notation, symbols, abbreviations	explanation
B	set containing all Wikipedia articles from the Wikipedia category *Bibel* ('Bible')
C	total number of tokens contained in a corpus
cf	*collection frequency*: total number of occurrences of a term *w* in a collection of documents (see 2.1.2.1)
D	total number of documents *d* in a collection
d	document in a collection
$\vec{d}$	document vector representing document *d* (see 2.2)
df	*document frequency*: number of documents in a collection in which a term *w* occurs at least once (see 2.1.2.1)
IDF, *idf*	*inverse document frequency*: $idf_i = IDF(w_i) = \log\left(\frac{D}{df_i}\right)$ (see2.1.2.2)
MI, I, mi, i	*mutual information*: $i_i = I(w_i = xYz) = \log\frac{p(xYz)}{p(xY)p(z\mid Y)}$ (see 2.1.2.5)
RIDF, ridf	*residual inverse document frequency:* $ridf_i = RIDF(w_i) = \log_2\frac{D}{df_i} - \log_2(1-e^{-\frac{cf_i}{D}}) = idf_i - \log_2(1-e^{-\frac{cf_i}{D}})$ (see 2.1.2.4)
tf	*term frequency*: number of occurrences of a term *w* in a document *d* (see 2.1.2.1)
TF.IDF, tf.idf	$tf.idf_{ij} = TF.IDF(w_{ij}) = tf_{ij}\cdot\log\frac{D}{df_i} = tf_{ij}\cdot idf_{ij}$ (see 2.1.2.3)
VT	set containing all Wikipedia articles from the Wikipedia category *Verschwörungstheorie* ('conspiracy theory')
w	term (word, N-gram, etc.)

Table 8: Notation and abbreviations

B. Probabilistic Motivation for IDF

> "[...] in the case where no relevance information is available, the best function for ranking documents is a combination of a simple match and a match using inverse document frequency weights" (Croft/Harper 1979:340).

Croft and Harper (1979), Manning and Schütze (1999) and also Robertson (2004: 7-10) argue that, on the basis of a few supplementary suppositions, *IDF* can be legitimated as a function of probability.

Manning and Schütze (1999: 551-553) derive *IDF* from the *odds of relevance*

$$O(d) = \frac{P(R|d)}{P(\neg R|d)}$$

where $P(R|d)$ is the probability of relevance of a given document *d* and $P(\neg R|d)$ is respectively the probability of document *d* being non-relevant.

In a first step, they construct a ranking function $g'(d)$ from $O(d)$ that ranks documents relative to a query *Q*. For this purpose, they rewrite $O(d)$ using Bayes' formula and then compute the log odds:

$$O(d) = \frac{P(d\,|\,R)P(R)}{P(d\,|\,\neg R)P(\neg R)}$$

$$\log(O(d)) = \log\left(O(d) = \frac{P(d\,|\,R)P(R)}{P(d\,|\,\neg R)P(\neg R)}\right)$$

$$\log(O(d)) = \log(P(d\,|\,R)) + \log(P(R)) - \log(P(d\,|\,\neg R)) - \log(P(\neg R))$$

In order to relate $O(d)$ to a query $Q = \{w_i\}$, they introduce a random variable X_i, with range {0,1} where 1 represents the occurrence and 0 the absence of w_i in *d*. On the Naïve Bayes assumption that word-order can be neglected and that the occurrences of words in a document are independent of each other (also called *bag-of-words* model), they formulate a ranking function $g(d)$, dropping the constant factor $\log(P(R)) - \log(P(\neg R))$:

$$g(d) = \sum_i \left(\log(P(X_i\,|\,R)) - \log(P(X_i\,|\,\neg R))\right)$$

$g(d)$ can be rewritten as follows:

$$g(d) = \sum_i X_i \log\frac{P(X_i = 1|R)}{1-P(X_i = 1|R)} + \sum_i X_i \log\frac{1-P(X_i = 1|\neg R)}{P(X_i = 1|\neg R)} + \sum_i \log\frac{1-P(X_i = 1|R)}{1-P(X_i = 1|\neg R)}$$

Since

$$\sum_i \log\frac{1-P(X_i = 1|R)}{1-P(X_i = 1|\neg R)}$$

is another constant factor, irrelevant for ranking, it can also be abandoned, finally yielding the function

$$g'(d) = \sum_i X_i \log\frac{P(X_i = 1|R)}{1-P(X_i = 1|R)} + \sum_i X_i \log\frac{1-P(X_i = 1|\neg R)}{P(X_i = 1|\neg R)}.$$

The subsequent simplifications are the most drastic ones. Since, in ad-hoc retrieval, there are generally no direct estimates for $P(X_i{=}1|R)$ and $P(X_i{=}1|\neg R)$ available, Manning and Schütze (1999) postulate that $P(X_i{=}1|R)$ is small and constant across terms and that, to approximate $P(X_i{=}1|\neg R)$, it can be presumed that none of the documents in the collection is relevant (which is an exaggeration of more realistic assumption that the vast majority of documents is irrelevant)[59]. Thus they arrive at the following approximations:

$$\sum_i X_i \log \frac{P(X_i = 1 | R)}{1 - P(X_i = 1 | R)} \approx c \sum_i X_i$$

$$\sum_i X_i \log \frac{1 - P(X_i = 1 | \neg R)}{P(X_i = 1 | \neg R)} \approx \sum_i X_i \log \frac{D}{df_i}$$

where c is a weighting factor, D is the number of documents in the collection and df_i is the number of documents that contain w_i at least once.

In other words, they reduce the former term to a simple weighted count of the number of first occurrences of the distinct query terms $\{w_i\}$ in the respective document and they further assume that the probability of a query term w_i occuring in a non-relevant document approximates the unconditional probability of a document containing w_i representable as the maximum likelihood estimate

$$P(w_i) \approx \frac{df_i}{D}$$

thereof.

Consequently $g'(d)$ rewrites as

$$g'(d) = c \sum_i X_i + \sum_i X_i idf_i$$

and represents a ranking function in which the impracticality of estimating the probability of document relevance directly is compensated for by frequency information. As Robertson (2004) points out, the derivation via a Naïve Bayes model provides a strong justification for *IDF*. *IDF* can thus be regarded as an implicit but direct measure of the probability of relevance (Robertson 2004:pdf:10).

59 Perhaps the most obvious blemish of the above approach is that the approximations, Manning and Schütze (1999: 553) suggest for $P(X_i=1|R)$ and $P(X_i=1|\neg R)$ are contradictory. If all documents in the collection were irrelevant, $P(X_i=1|R)$ should actually be 0.

C. Allan's Hyperlink Taxonomy

manual	require human authoring, "those [links] which connect documents which describe circumstances under which one document occurred, those which collect the various components of a debate or argument, and those which describe forms of logical implication (caused-by, purpose, warning, and so on)" (Allan 1996:pdf:2)	
pattern-Matching	typically definitions, e.g. links that point from a term to a description of the meaning of the term	
automatic	*revision*	derives from a revision-history, f.e. version numbers, backup copies, etc.
	summary	points from a larger section to a summary of that section
	expansion	inverse of summary, points from a digest to an elaboration
	equivalence	connection between a strongly related discussion of the same topic
	comparison	links that identify similarities and differences between texts
	tangent	subtype of equivalence links that relate topics in an unusual manner, the target document is usually disconnected from other documents that are related to the source document by an equivalence link
	aggregate	agglomerating documents that are highly interconnected by equivalence links

Table 9: Allan's link types (Allan 1996:pdf:2-3)

D. Graph-Related Terminology

Following Broder et al. (2000:html), Steyvers and Tenenbaum (2005:pdf:4-9), and Wikipedia.org (2007:http://en.wikipedia.org/wiki/Graph_%28mathematic% 29), this study uses the graph-related terminology given in Table 10:

arc	unidirectional connection between two *vertices*, formally defined as an ordered pair of two distinct *vertices*
average distance[60]	average over the shortest *path* lengths of all pairs of *vertices*[61]
complete graph	a *graph* where all pairs of vertices are joined by an *edge*
component	a set of *vertices* in a graph where every pair of *vertices* is connected by at least one *path*
cluster coefficient	the probability that two distinct *vertices* connected to a third *vertex* will be connected themselves
degree	The number of incoming and outgoing *arcs* of a *vertex* in a *directed graph* is respectively called the *in-* or *out-degree* of the *arc*, in case of an *undirected graph*, the number of *edges* connecting a *vertex* to other *vertices* is called the *degree* of the *vertex*.
degree distribution	The *degree distribution* $P(k)$ is the probability that a randomly chosen node will have *degree* κ.
diameter[62]	maximum shortest *path* length over all pairs of *vertices* in the graph
directed graph	A *directed graph G* is an ordered pair (V, A), where V is a nonempty finite set of vertices and A is a finite set of *arcs*, where an *arc* is an ordered pair of elements of V.
distance	The *distance* between two *vertices* is defined as the length of the shortest *path* that connects them.
edge	symmetric connection between two *vertices* formally defined as an unordered pair of *vertices*
graph	A *graph G* is either a *directed graph*, where *vertices* are connected by *arcs*, an *undirected graph*, where *vertices* are connected by *edges* or a *mixed graph* involving both *vertices* and *edges*.
mixed graph	A *mixed graph G* is a triple (V, E, A) involving both ordered and unordered pairs of *vertices*.
path	In an *undirected graph*, a *path* is a sequence of *edges* that connects two *vertices*. In a *directed graph* a *path* is a sequence of *arcs* that unidirectionally leads from one vertex to another.
undirected graph	An *undirected graph G* is an ordered pair (V, E), where V is a nonempty finite set of vertices and E is a finite set of *edges*, where an *edge* is an unordered pair of distinct elements of V.
vertex	A *vertex* is a node.

Table 10: Graph-related terminology

[60] Broder et al. (2000:htm) warn that inter-author terminology is inconsistent. They point out that Barabási et al. (2000) use the term *diameter* for what's referred to as *average distance* here.

[61] Broder et al. (2000:htm) point out that the use of the average distance may be problematic: Even one single pair of unconnected *vertices* will provoke an average distance that is infinitely large.

[62] See Footnote 60.

E. Experimental Results

This appendix contains additional result details from the `CrossRef` experiment.

E.a Results of comparisons to Wikipedia Link Structures

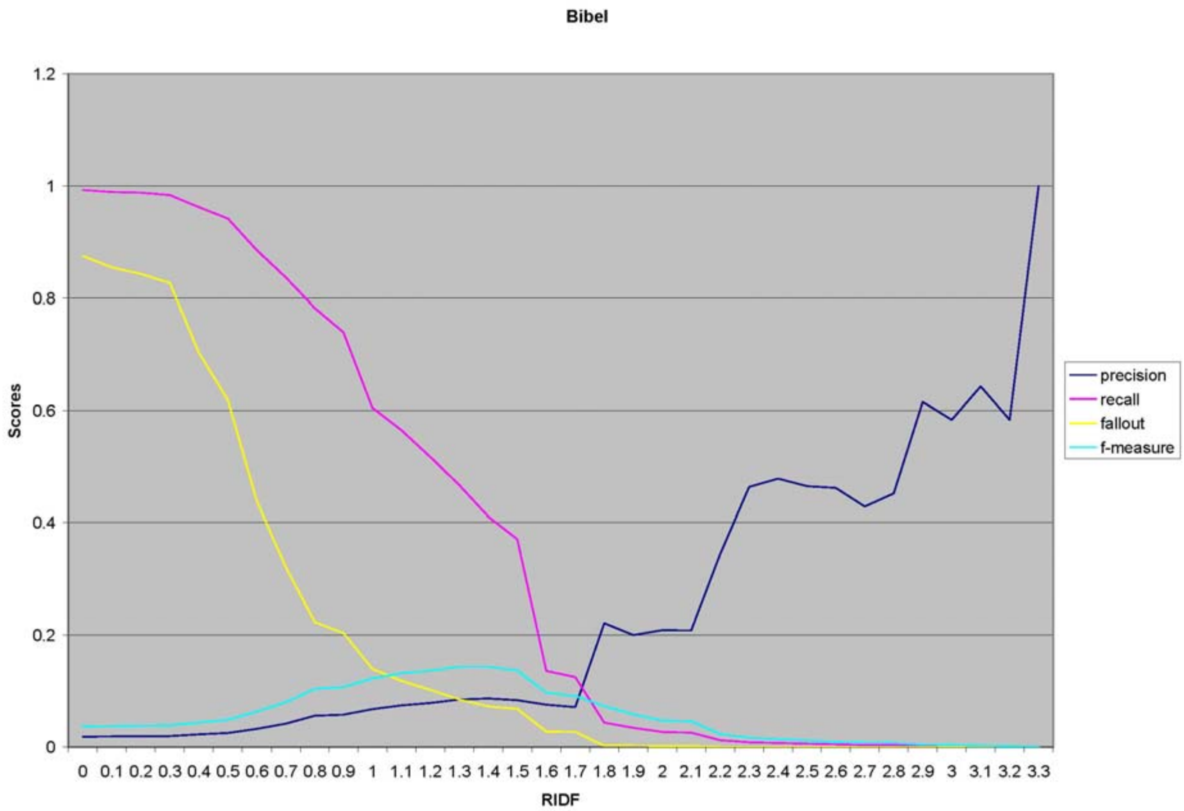

Figure 7: scores for distinct *ridf* values (category *Bibel*)

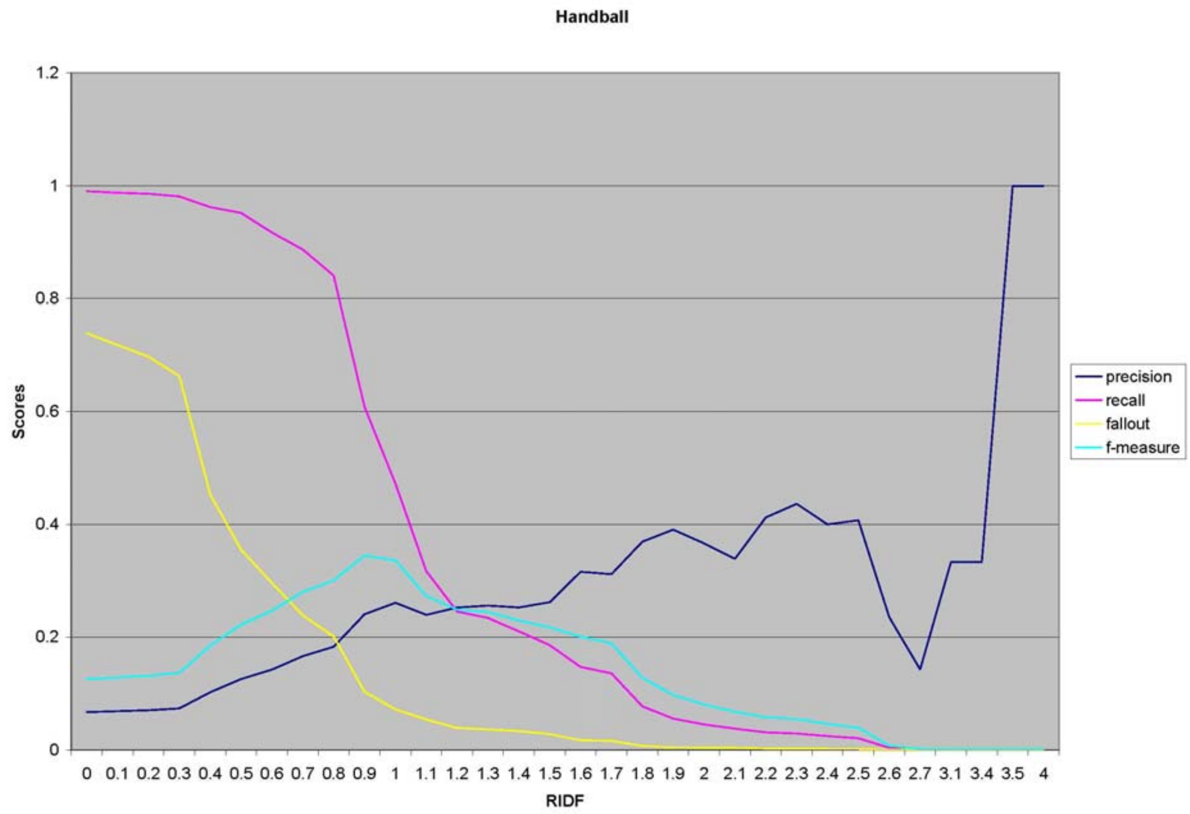

Figure 8: scores for distinct *ridf* values (category *Handball*)

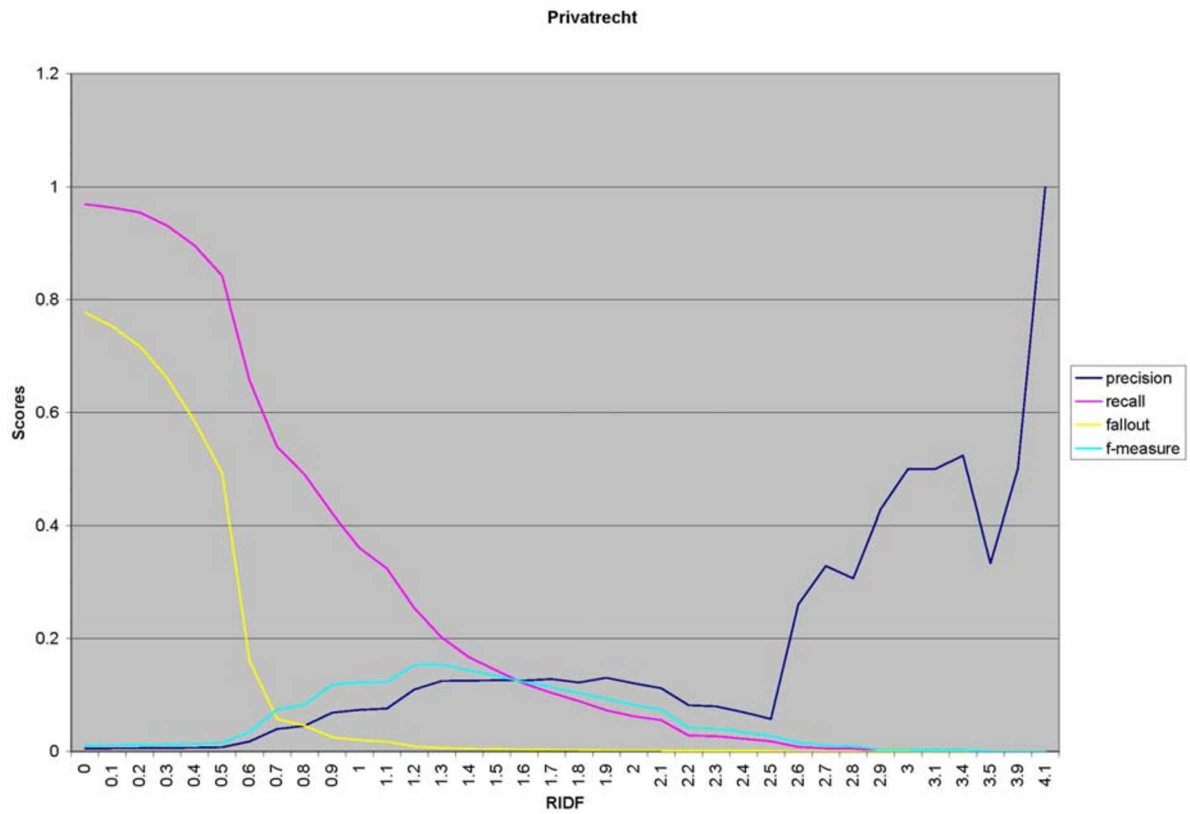

Figure 9: scores for distinct *ridf* values (category *Privatrecht*)

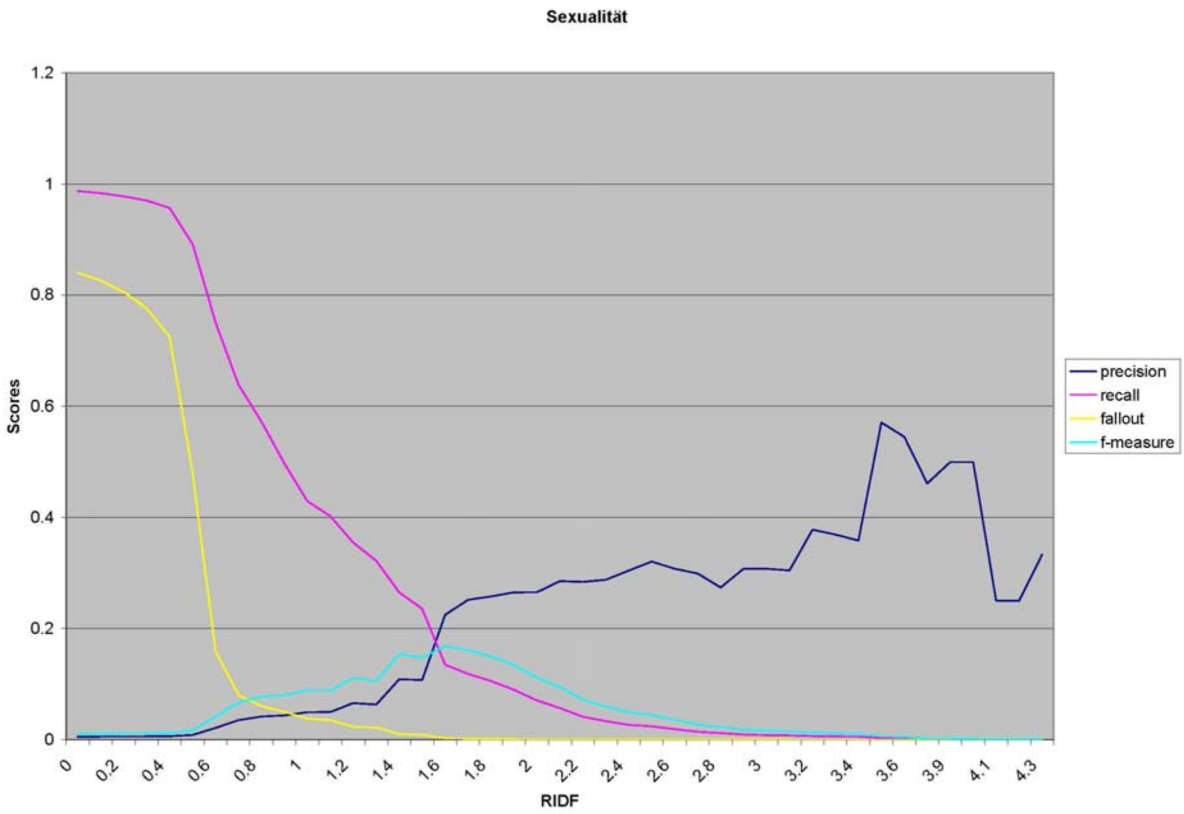

Figure 10: scores for distinct *ridf* values of (category *Sexualität*)

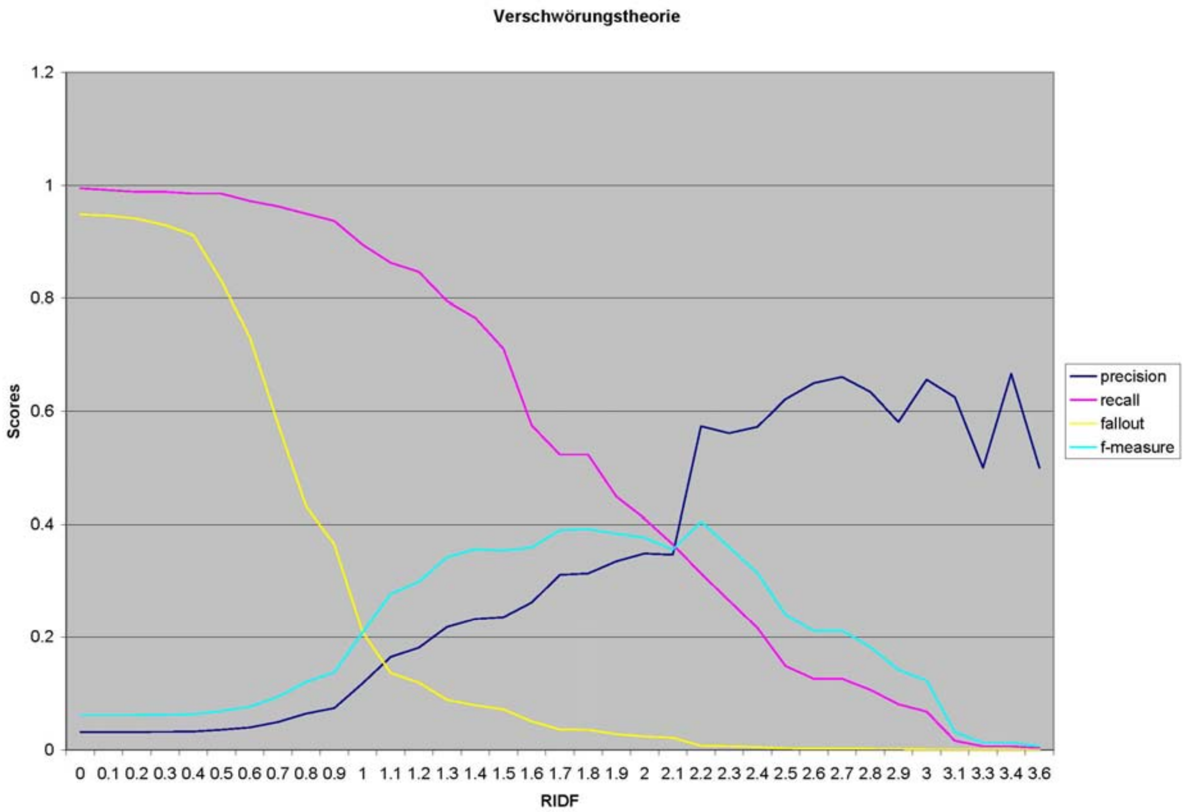

Figure 11: scores for distinct *ridf* values (category *Verschwörungstheorie*)

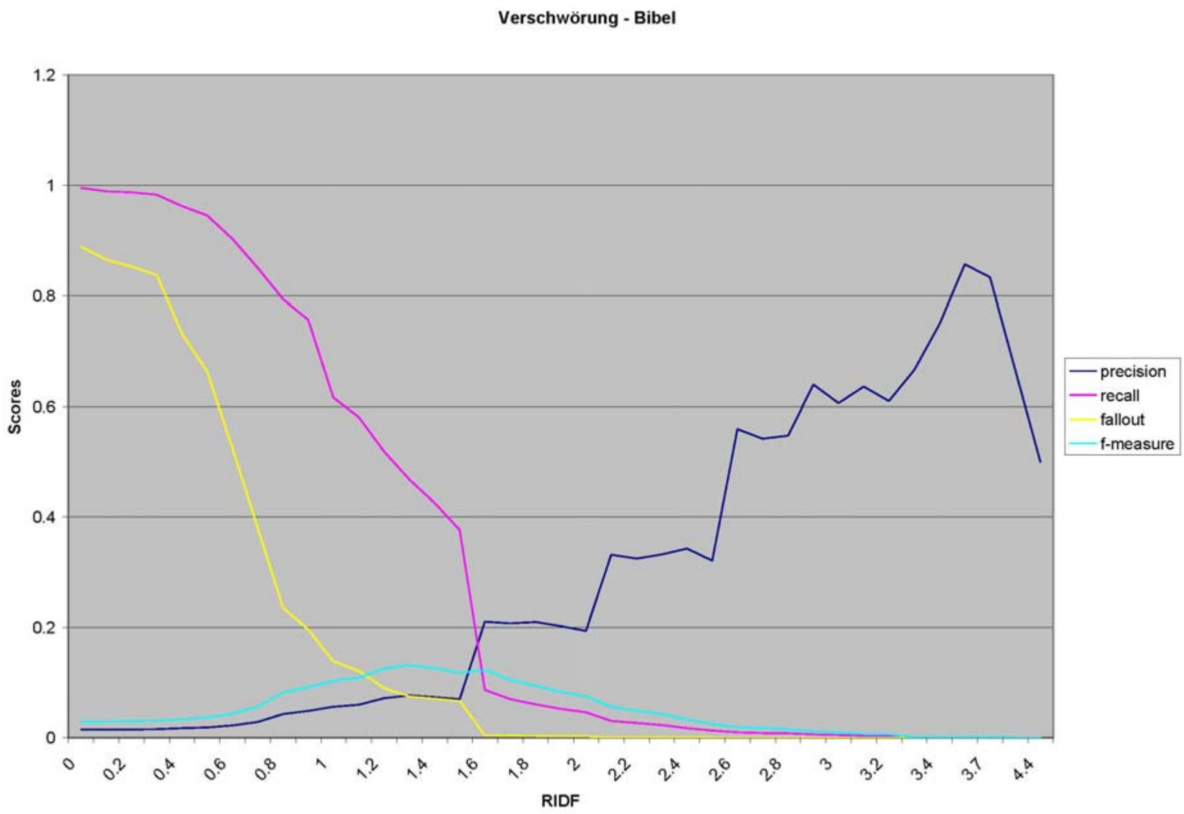

Figure 12: scores for distinct *ridf* values (category *Verschwörungstheorie* + *Bibel*)

E.b Top Key N-Grams

tf	*df* (≥2)	*ridf*	n-gram (2 ≤ n ≤ 5)				
44	3	3.841704	ecce	homo			
39	4	3.256347	des	muensters			
29	3	3.251395	biblischen	theologie			
28	3	3.201513	das	muenster			
18	2	3.156491	herz	jesu	verehrung		
18	2	3.156491	jesu	verehrung			
42	5	3.039108	ave	maria			
16	2	2.988056	becker	2005			
16	2	2.988056	der	mit	seinen		
16	2	2.988056	gospel	of	mark		
16	2	2.988056	schnelle	2005			
16	2	2.988056	soeding	1998			
15	2	2.895692	stralsund	marienkirche			
30	4	2.884523	die	historisch	kritische	methode	
30	4	2.884523	herz	jesu			
59	8	2.838765	biblische	theologie			
14	2	2.796902	die	goldene	madonna		
14	2	2.796902	marienkirche	stralsund			
14	2	2.796902	meiser	kuehneweg			
14	2	2.796902	steck	1999			

Table 11: top 20 n-grams for category *Bibel*

tf	*df* (≥2)	*ridf*	n-gram (2 ≤ n ≤ 5)				
34	2	3.968908	text	flag	of		
34	2	3.968908	text	flag			
21	2	3.318698	1	23	14	23	20
21	2	3.318698	1	23	14	23	
21	2	3.318698	1	23	14		
21	2	3.318698	1	23			
21	2	3.318698	14	23	20		
21	2	3.318698	14	23			
21	2	3.318698	23	14	23	20	
21	2	3.318698	23	14	23		
21	2	3.318698	23	14			
21	2	3.318698	23	20			
21	2	3.318698	80	1	23	14	23
21	2	3.318698	80	1	23	14	
21	2	3.318698	80	1	23		
21	2	3.318698	80	1			
36	4	3.0445	flag	of			
17	2	3.027768	spartak	kiew			
27	4	2.660468	empor	rostock			
13	2	2.654716	hypo	niederoesterreich	aut		

Table 12: top 20 n-grams for category *Handball*[63]

tf	*df* (≥2)	*ridf*	n-gram (2 ≤ n ≤ 5)				
37	2	4.195781	e	books			
34	2	4.074897	us	amerikanischer			
29	2	3.847261	nur	maenner			
34	3	3.489935	der	wechsel			
34	3	3.489935	ias	39			
34	3	3.489935	nein	ja			
33	3	3.447235	der	handelsvertreter			
43	4	3.410379	e	book			
21	2	3.384552	documentation	license			
21	2	3.384552	free	documentation	license		
21	2	3.384552	free	documentation			
21	2	3.384552	gnu	free	documentation	license	
21	2	3.384552	gnu	free	documentation		
21	2	3.384552	gnu	free			
17	2	3.081176	13	nr			
17	2	3.081176	der	sorbonne			
24	3	2.991126	1	mio			
15	2	2.901342	das	zeugnis			
111	15	2.846638	vob	b			
14	2	2.802176	bremer	hoehe			

Table 13: top 20 n-grams for category *Privatrecht*

[63] These key terms may appear strange but they reflect document content very well.

tf	*df* (≥2)	*ridf*	n-gram (2 ≤ n ≤ 5)				
60	2	4.882257	nein	nein			
46	2	4.504664	pet	shop			
38	2	4.23231	rowspan	2			
35	2	4.114897	pet	shop	boys		
35	2	4.114897	shop	boys			
34	2	4.073487	freddie	mercury			
50	3	4.038356	brokeback	mountain			
32	2	3.986845	britischen	charts			
48	3	3.980282	tennessee	williams			
30	2	3.894556	frank	n	furter		
30	2	3.894556	frank	n			
30	2	3.894556	n	furter			
27	2	3.743785	a	t			
78	6	3.668444	karl	ii			
37	3	3.609284	alan	turing			
37	3	3.609284	village	people			
23	2	3.514103	anne	rice			
23	2	3.514103	den	britischen	charts		
23	2	3.514103	in	den	britischen	charts	
34	3	3.488524	marcel	proust			

Table 14: top 20 n-grams for category *Sexualität*

tf	*df* (≥2)	*ridf*	n-gram (2 ≤ n ≤ 5)				
62	2	4.652726	der	titanic			
51	2	4.422799	skull	bones			
33	3	3.296167	men	in			
18	2	3.080078	johann	karl	august		
18	2	3.080078	johann	karl			
50	5	3.07705	da	vinci			
48	5	3.02767	area	51			
26	3	2.98631	der	prieur			
26	3	2.98631	von	mainz			
25	3	2.934621	in	black			
25	3	2.934621	men	in	black		
16	2	2.920042	baggesen	jens			
15	2	2.831886	harvey	oswald			
15	2	2.831886	jacobi	friedrich			
15	2	2.831886	lee	harvey	oswald		
15	2	2.831886	lee	harvey			
23	3	2.824133	alfred	rosenberg			
23	3	2.824133	dem	mond			
31	4	2.800644	sauni	re			
22	3	2.764915	auf	dem	mond		

Table 15: top 20 n-grams for category *Verschwörungstheorie*

tf	*df* (≥2)	*ridf*	n-gram (2 ≤ n ≤ 5)				
62	2	4.91402	der	titanic			
51	2	4.63935	skull	bones			
42	2	4.36506	die	titanic			
44	3	3.845918	ecce	homo			
50	5	3.289499	da	vinci			
39	4	3.260086	des	muensters			
29	3	3.25418	biblischen	theologie			
48	5	3.231898	area	51			
28	3	3.204202	das	muenster			
18	2	3.158222	herz	jesu	verehrung		
18	2	3.158222	jesu	verehrung			
18	2	3.158222	johann	karl	august		
18	2	3.158222	johann	karl			
26	3	3.098583	der	prieur			
34	4	3.065384	men	in			
42	5	3.043132	ave	maria			
16	2	2.989596	baggesen	jens			
16	2	2.989596	becker	2005			
16	2	2.989596	der	mit	seinen		
16	2	2.989596	gospel	of	mark		

Table 16: top 20 n-grams for categories *Verschwörungstheorie* + *Bibel*

E.c Example of n-grams with high *MI*- and high *RIDF*-values

tf	*df* (≥2)	*ridf*	*mi*	n-gram (2 ≤ n ≤ 5)	
33	3	3.296167	29.17734	men	in
7	2	1.772188	29.11185	nixon	und
38	12	1.475521	28.68726	weisen	von
11	3	1.819337	28.58326	reichsgraf	von
10	2	2.271778	28.55267	constantin	von
10	2	2.271778	28.44576	erzbischof	von
20	6	1.637253	28.39513	ignaz	von
13	3	2.050397	28.3837	xaver	von
6	2	1.554802	28.34622	konzil	von
8	3	1.374871	28.17292	untergang	der
13	4	1.635359	28.15694	wissenschaft	und
11	4	1.404299	28.04739	van	der
10	3	1.686816	27.8608	buergermeister	von
13	4	1.635359	27.82427	bischof	von
17	5	1.68063	27.80919	ferdinand	von
17	3	2.417595	27.73325	theodor	von
8	3	1.374871	27.58795	weg	der
8	3	1.374871	27.53887	andreas	von
10	2	2.271778	27.5322	offenbarung	des
12	2	2.524853	27.47083	umgang	mit

Table 17: top 20 n-grams sorted by *MI* with *ridf* ≥ 2.2 (category *Verschwörungstheorie*)

F. Perl Code: CrossRef.pm

```
#!/usr/bin/perl

package CrossRef;

#######################################################################
##
## ABOUT THIS MODULE
##
#######################################################################
##
## AUTHOR: Heike Johannsen
## NAME: CrossRef.pm
## VERSION: 0.01
## CREATED: 02/2007.
## LAST UPDATE: 27.03.2007
##
#######################################################################
##
## PURPOSE:
## This module is the implementation part of a BA-thesis
## called "Linking Documents by Distinctive Phrases".
## It is designed to:
## - calculate RIDF and MI values of n-grams given a corpus,
## - extract a set of key n-grams based on high RIDF,
## - link documents if they contain common key n-grams,
## - match the resulting link structure against a gold-standard,
## - create HTML output and other files for human assessment.
##
## IMPLEMENTATION DETAILS:
## In order to calculate the parameter frequencies for RIDF
## (residual inverse document frequency) and MI (Mutual
## information)from corpus data, this module employs a suffix
## array as described in Yamamoto/Church (1998).
##
## TERMINOLOGY:
## Stuff for lexicon generation with suffix array
## (following Yamamoto/Church 1998):
## 'Term frequency (TF)': The total number of
## occurrences of an N-gram in the entire collection
## (!TF is called 'collection frequency' (CF) in Manning/Schütze
## (1999)!).
## 'Document frequency (DF)': The number of documents in
## the corpus that contain an N-Gram at least once
## 'Longest common prefix (LCP)': The longest common
## prefix of two or more suffixes stored in the suffix array.
## 'Class': "Let <i,j> be an interval on the suffix array, [...].
## Class(<i,j>) is the set of substrings that start every suffix
## within the interval and no suffix outside the interval"
## (Yamamoto/Church 1998:7). As I understand this, a class contains
## prefixes of an LCP that is common to substrings over an interval.
## 'Bounding LCPs': The first LCP of an interval and the first LCP
## after an interval.
## 'Interior LCPs: The set of LCPs following the first LCP of an
## interval and preceding the first LCP after the interval.
## 'Longest bounding LCP (LBL)': The longer one of the 2 bounding LCPs.
## 'Shortest interior LCP (SIL)': The shortest of the interior LCPs.
## 'LCP delimited': A class is LCP-delimited iff LBL < SIL.
## 'Trivial': Intervals and classes of size 1.
##
######################################################################
```

```
######################################################################
## CURRENT STATE:
## Grown peace of code, partially suboptimal, not ready for any
## public use, but sufficient for the experiment.
## Not all features potentially available in this module have
## recently been tested. Therefore, there is no guarantee that
## this module is absolutely bug-free.
## The tested and recently used functions are:
## &test_parameters,
## &link_and_match_directly,
## &demo.
## and client functions thereof (depending on config settings).
## See USAGE for working settings.
##
######################################################################
##
## USAGE:
## Recently, this module has only been used to process
## file input. Hence, you'll need a corpus available in
## text format.
## This module uses configuration files to manage parameter
## settings. You can create such a config file by running
## demo() without arguments and refusing to use the default
## file. But ideally, you should have a template for a config
## file and a demo version.
## For a demo, just type '>perl CrossRef.pm' on the command-line.
##
######################################################################
##
## REFERENCES:
## MANNING, Christopher D. / SCHÜTZE, Hinrich (1999).
## Foundations of Statistical Natural Language Processing.
## Cambridge, Massachusetts / London, England: MIT Press.
## YAMAMOTO, Mikio / CHURCH, Kenneth Ward (1998).
## "Using Suffix Arrays to Compute Term Frequency
## and Document Frequency for All Substrings in
## a Corpus". In: Proceedings of ACL Workshop on Very
## Large Corpora. P. 28-37. Montreal.
## http://acl.ldc.upenn.edu/J/J01/J01-1001.pdf (03/2007).
##
#######################################################################

BEGIN {

  # REQUIRE...
  require WWW::Wikipedia;

  # USE...
  use 5.008008;
  use strict;
  use warnings;
  use WWW::Wikipedia;
  use constant {

    # General N-gram data keys:
    ng_start => 'ng_start', # N-gram hash key: Start index in corpus.
    ng_length => 'ng_length', # N-gram hash key: N-Gram length.
    ng_tf => 'tf', # N-gram hash key: term frequency.
    ng_df => 'df', # N-gram hash key: document frequency.
    ng_mi => 'mi', # N-gram hash key: mutual information.
    ng_ridf => 'ridf', # N-gram hash key: Residual IDF.
    ng_txt => 'ng_txt', # N-gram hash key: Text.
    ng_list => 'ng_list', # Key: N-gram list.

    # N-gram keys for MI-calculation:
    xYz => 'tf_xYz', # Key for input hash to mi_cy();
    Y => 'tf_Y', # Key for input hash to mi_cy();
    xY => 'tf_xY', # Key for input hash to mi_cy();
    Yz => 'tf_Yz', # Key for input hash to mi_cy();
```

```
        # Parameter keys (%params):
        df_min => 'df_min', # Minimum DF.
        ridf_min => 'ridf_min', # Minimum RIDF.
        mi_max => 'mi_max', # Maximum MI.
        length_min => 'length_min', # Minimum key N-gram length.
        store_lk => 'store_lk', # Store the linking keywords?
        limit => 'lim', # Maximal N-gram length.
        lex_file => 'lexfile', # Lexicon.
        link_cr => 'linkfile', # Link file.
        keyw_file => 'keyword_file', # Keyword file.
        dat_dir => 'data_dir', # Key for data directory.
        page_dir => 'page_dir', # Key for collection directory.
        html_dir => 'html', # Key: Output directory for HTML files.
        pid_o => 'pid_orig', # Key for original page-data file.
        pid_cr => 'pid_cr', # Key for CrossRef PID-file.
        link_o => 'links_orig', # Original link adjacency matrix file.
        config_file => 'config_file', # Key: config data file.
        score_file => 'score_file', # Key: score file.
        param_result_file => 'param_res', # Key: outfile parameter testing.
        stop_list => 'stop_list', # Stopword list.
        eval_sheet => 'eval_sheet', # Filename fpr evaluation sheet.
        alpha => 'alpha', # Weighting factor for F-measure.
        print_prog => 'print_prog', # Key: Print progress info?

        # Adjacency matrix - relevant keys:
        pid_lm_index => 0, # Matrix-ID column in PID file.
        pid_fn_index => 1, # Real filename column in PID file.
        pid_wid_index => 2, # Wiki-ID column in PID file.
        pid_pt_index => 3, # Page title column in PID file.
        cr_matrix => 'cr_matrix', # CrossRef link matrix.
        o_matrix => 'o_matrix', # Original link matrix.
        cr_pid_map => 'cr_pid_map', # CrossRef PID map.
        o_pid_map => 'o_pid_map', # Original PID map.
        linkey =>  'linkey', # Map of linking keywords.

        # Matches, evaluation measures - keys:
        true_pos => 'true_pos', # Key: nr of true positives.
        false_pos => 'false_pos', # Key: nr of false positives.
        true_neg => 'true_neg', # Key: nr of true negatives.
        false_neg => 'false_neg', # Key: nr of false negatives.
        prec => 'precision', # Key: precision.
        rec => 'recall', # Key: recall.
        fall => 'fallout', # Key: fallout value.
        f_score => 'f-measure', # F-measure name.

        # Other keys:
        meas => 'meas', # Threshold measure (filter_results()).
        thresh => 'thresh', # Threshold value (filter_results()).
        keywords => 'keywords', # Key: Keyword-list.

        # Default values, symbols:
        no_mi_val => -999999, # Value for undefined MI.
        no_val => undef, # No value.
        sep => ';', # Column separator in files.
        file_suff => '.txt', # Filename extension.
        comment => '#', # Comment marker in config file.
        mi_step => 0.5, # Decrement MI by this value in parameter testing.
        ridf_step => 0.1, # Increment RIDF by this val in parameter testing.
        abs_df_min => 2, # Absolute DF minimum.
        html_suffix => '.html', # HTML filename extension.
        key_html_prefix => '00_Keyword_', # Prefix for keyword html files.
        demo_config => '../config_files/config_demo.txt', # Demo config.

        # Tokenization:
        end_of_doc => '°', # Document end symbol.
        word_boundary => qr/(?:[^\w])+/, # Token delimiter.

    };
}
```

```
# GLOBAL VARIABLES:

# Delimiters:
my $word_boundary; # Word delimiter.

# Mappings:
my %ix_to_doc; # Corpus index to document map.
my %token; # Beginning of token to token end.
my %next_token; # For each token beginning of next token.
my %fn_to_doc; # Filename -> doc ID mapping.
my %linking_keywords; # Stores linking words for document pairs.

# Counts:
my $cur_ix; # Start of current word (&tokenize).
my $nr_docs = 0; # Number of documents in corpus.

# Text data:
my $corpus; # The corpus on which to calculate frequencies.

# Data arrays:
my @suff_arr; # Suffix array.
my @lcp; # LCP array.
my @link_table; # Adjacency matrix for links.
my @keywords; # Keyword list.
my @stopwords; # Stopword list.

# Other:
my $n_max; # Maximal N-gram length;
my $print_info; # Flag: Print progress info to STDOUT?


# MAIN PROGRAM:

demo($ARGV[0]); # Run a demo...
1; # End of main program.


# SUBROUTINES:

# Runs a demo on parameter testing or link creation.
# PARAMETERS:
# Path to configuration file
# (optional but recommended).
sub demo {
  my $settings = shift @_; # Configuration file.
  my $params; # Reference to parameter hash.
  unless (defined $settings){
    print "Use demo configuration file? {1,0}\n";
    my $yes = <STDIN>;
    if ($yes == 1){
      $params = load_config(demo_config());
    } else {
      $params = prompt_for_settings();
    }
  } else {
    $params = load_config($settings);
  }
  print "Test parameters (type '1') or create HTML (type '0')?\n";
  my $function = <STDIN>;
  if ($function == 1){
    # Use slightly different settings...
    my $ridf_min = $$params{ridf_min()};
    $$params{ridf_min()} = 0;
    $$params{store_lk()} = 0;
    test_parameters($params);
    $$params{ridf_min()} = $ridf_min;
    $$params{store_lk()} = 1;
  } else {
    link_and_match_directly($params);
  }
}
```

```
# This sub creates links for a range of RIDF and MI values and matches
# the outcome against a gold-standard.
# MI-testing has originally been intended to explore the possibility of
# reducing lexicon-size by MI above a certain threshold. However, that
# plan has been abandoned. Therefore, MI-testing has become somewhat
# obsolete.
# Creates an output file with test results.
# RESTRICTION:
# Don't run this function immediately before or after
# another function in this module.
# PARAMETERS:
# Ref to hash with the following keys (see constants):
# page_dir => collection directory (should not end with slash),
# links_orig => path to gold standart link map,
# pid_orig => original page identification data,
# stop_list => path to stopword list,
# param_res => output file for test-results,
# limit => maximal length of an N-gram,
# ridf_min => RIDF threshold (minimum RIDF)(optional),
# df_min => DF threshold (minimum DF) (optional),
# mi_max => MI threshold (maximum MI) (optional).
# alpha => weighting factor for precision and recall in f-measure,
# print_prog => Flag: Print progress info at runtime?
# SIDE EFFECT:
# Selects STDOUT.
sub test_parameters {
  my $params = shift @_;
  if ($$params{print_prog()}){
    $print_info = $$params{print_prog()};
  }
  append_to_file($$params{param_result_file()},
                 "---------------------------------------------\n");
  append_to_file($$params{param_result_file()},
                 "New Test: Start values:\n");
  append_to_file($$params{param_result_file()}, "Min RIDF: " .
                 $$params{ridf_min()} . "\n");
  append_to_file($$params{param_result_file()}, "Max MI: " .
                 $$params{mi_max()} . "\n");
  my $best_ridf = $$params{ridf_min()};
  my $best_mi = $$params{mi_max()};
  my $best_f = -1;
  my $best_prec = -1;
  my $best_rec = -1;
  my $best_fall = -1;
  my $lex_size;
  my $lex_size_fin;
  my @lex_copy;
  my $collection = load_collection($params);
  $$params{ng_list()} = generate_frequency_lexicon($collection,
                                                   $$params{limit()});
  $$params{ng_list()} = filter_keywords($params);
  {
    my $lex = $$params{ng_list()};
    $lex_size = @$lex;
    $lex_size++;
    @lex_copy = @$lex; # Save a copy for reuse in MI-testing.
  }
  my $pid_orig = load_pid_data($$params{pid_o()});
  my $links_orig = load_link_matrix($$params{link_o()});
  symmetrize($links_orig);
  printprog("Lexicon generated, original links loaded.\n");
  my $time = times()/60;
  printprog("Duration so far (minutes): $time\n");
```

```
my $header = ng_ridf . sep
              . ng_mi . sep
              . prec . sep
              . rec . sep
              . fall . sep
              . f_score . sep
              . true_pos . sep
              . false_pos . sep
              . true_neg . sep
              . false_neg . sep . "\n";
append_to_file($$params{param_result_file()}, $header);
# Find RIDF that gives best F-measure...
printprog("Finding best thresholds...\n");
while ($lex_size > 0){
  printprog("Raising RIDF: Remaining keywords: $lex_size\n");
  my $c_lex = filter_keywords($params);
  unless(@$c_lex < $lex_size){
    $$params{ridf_min()} += ridf_step;
    next;
  } else {
    $lex_size = @$c_lex;
    $$params{ng_list()} = $c_lex;
  }
  link_by_lex($params, 0);
  my $matches = match_links($links_orig,
                             \@link_table,
                             $pid_orig,
                             \%fn_to_doc);
  if ($$matches{prec()} > 0 && $$matches{rec()} > 0){
    my $f = f_measure($$params{alpha()},
                      $$matches{prec()},
                      $$matches{rec()});
    printprog("F: RIDF: " . $$params{ridf_min()} . ": $f\n");
    if ($f >= $best_f){
      $best_f = $f;
      $best_prec = $$matches{prec()};
      $best_rec = $$matches{rec()};
      $best_fall = $$matches{fall()};
      $best_ridf = $$params{ridf_min()};
      my $lex = $$params{ng_list()};
      $lex_size_fin = $lex_size;
    }
    my $result_line = $$params{ridf_min()} . sep()
                    . $$params{mi_max()} . sep()
                    . $$matches{prec()} . sep()
                    . $$matches{rec()} . sep()
                    . $$matches{fall()} . sep()
                    . $f . sep()
                    . $$matches{true_pos()} . sep()
                    . $$matches{false_pos()} . sep()
                    . $$matches{true_neg()} . sep()
                    . $$matches{false_neg()} . sep()
                    . "\n";
    append_to_file($$params{param_result_file()}, $result_line);
  }
  $$params{ridf_min()} += ridf_step;
}
printprog("Best RIDF calculated: $best_ridf\n");
# Now, see how much MI can be lowered without lowering the best
# F-score...
$lex_size = @lex_copy;
$lex_size++;
$$params{ng_list()} = \@lex_copy;
$$params{ridf_min()} = $best_ridf;
while ($lex_size > 0){
  printprog("Lowering MI: Remaining keywords: $lex_size\n");
  my $c_lex = filter_keywords($params);
```

```
      unless(@$c_lex < $lex_size){
        $$params{mi_max()} = $$params{mi_max()} - mi_step;
        next;
      } else {
        $lex_size = @$c_lex;
        $$params{ng_list()} = $c_lex;
      }
      link_by_lex($params, 0);
      my $matches = match_links($links_orig,
                                \@link_table,
                                $pid_orig,
                                \%fn_to_doc);
      if ($$matches{prec()} > 0 && $$matches{rec()} > 0){
        my $f = f_measure($$params{alpha()},
                          $$matches{prec()},
                          $$matches{rec()});
        printprog("F: MI: " . $$params{mi_max()} . ": $f\n");
        unless ($f < $best_f){
          $best_mi = $$params{mi_max()};
          printprog("F: " . $$params{mi_max()} . ": $f\n");
        }
        my $result_line = $$params{ridf_min()} . sep()
                    . $$params{mi_max()} . sep()
                    . $$matches{prec()} . sep()
                    . $$matches{rec()} . sep()
                    . $$matches{fall()} . sep()
                    . $f . sep()
                    . $$matches{true_pos()} . sep()
                    . $$matches{false_pos()} . sep()
                    . $$matches{true_neg()} . sep()
                    . $$matches{false_neg()} . sep()
                    . "\n";
        append_to_file($$params{param_result_file()}, $result_line);
      }
      $$params{mi_max()} = $$params{mi_max()} - mi_step;
    }
    printprog("Best MI calculated: $best_mi\n");
    # Save relevant data.
    append_to_file($$params{param_result_file()},
                   "Nr of documents: $nr_docs\n");
    append_to_file($$params{param_result_file()},
                   "Nr of tokens: " . keys(%token) . "\n");
    append_to_file($$params{param_result_file()},
                   "Nr of keywords (N-grams): $lex_size_fin\n");
    append_to_file($$params{param_result_file()},
                   "Best RIDF minimum: $best_ridf\n");
    append_to_file($$params{param_result_file()},
                   "Best MI maximum: $best_mi\n");
    append_to_file($$params{param_result_file()},
                   "Best F-score: $best_f\n");
    append_to_file($$params{param_result_file()},
                   "Weighting factor: " . $$params{alpha()} . "\n");
    append_to_file($$params{param_result_file()},
                   "Precision: $best_prec\n");
    append_to_file($$params{param_result_file()},
                   "Recall: $best_rec\n");
    append_to_file($$params{param_result_file()},
                   "Fallout: $best_fall\n");
    $time = times()/60;
    printprog("Duration so far (minutes): $time\n");
    printprog("Collection linked.\n");
    append_to_file($$params{param_result_file()},
            "Processing time (minutes): $time\n");
    append_to_file($$params{param_result_file()},
                   "---------------------------------------------\n");
  }
```

```perl
# Links a collection of documents directly after calculating
# TF, DF, RIDF and MI. Hence, thresholds must be known in advance.
# Compares the resulting link-structure to a gold-standard.
# Optionally creates some output files with the results.
# PARAMETERS:
# Ref to hash with the following keys (see constants):
# page_dir => collection directory (should not end with slash),
# lex_file => output file for frequency lexicon (optional),
# link_cr => output file for CrossRef link adjacency matrix,
# keyw_file => output file for mere keywords (optional),
# pid_cr => output file for CrossRef page identification data
(optional),
# score_file => output file for scores (optional).
# limit => maximal length of an N-gram,
# ridf_min => RIDF threshold (minimum RIDF)(optional),
# df_min => DF threshold (minimum DF) (optional),
# mi_max => MI threshold (maximum MI) (optional).
# print_prog => Flag: Print progress info at runtime?
# SIDE EFFECT:
# Selects STDOUT.
sub link_and_match_directly {
  my $params = shift @_;
  my $outdir;
  if ($$params{print_prog()}){
    $print_info = $$params{print_prog()};
  }
  my $pid_orig = load_pid_data($$params{pid_o()});
  my $links_orig = load_link_matrix($$params{link_o()});
  symmetrize($links_orig);
  if ($$params{html_dir()}){
    # Postpone HTML-creation.
    $outdir = $$params{html_dir()};
    $$params{html_dir()} = undef;
  }
  link_directly($params);
  my $matches = match_links($links_orig,
                            \@link_table,
                            $pid_orig,
                            \%fn_to_doc);
  if (defined($$params{score_file()})){
    write_hash_info($matches, $$params{score_file()});
  }
  if ($outdir){
    $$params{html_dir()} = $outdir;
    $$params{cr_matrix()} = \@link_table;
    $$params{cr_pid_map()} = \%fn_to_doc;
    $$params{o_matrix()} = $links_orig;
    $$params{o_pid_map()} = $pid_orig;
    if ($$params{store_lk()}){
      $$params{linkey()} = \%linking_keywords;
    }
    create_html($params);
  }
  if ($$params{eval_sheet()}){
    $$params{cr_matrix()} = \@link_table;
    $$params{cr_pid_map()} = \%fn_to_doc;
    $$params{o_matrix()} = $links_orig;
    $$params{o_pid_map()} = $pid_orig;
    create_eval_sheet($params);
  }
}
```

```
# Links documents by a predefined lexicon.
# PARAMETERS:
# Hash with the following key (constant):
# ng_list => Reference to N-gram lexicon.
# Boolean flag: 1 for switching off filtering,
# 0 or undef for filtering.
sub link_by_lex {
  my $params = shift @_;
  my $filter = shift @_;
  my $lex_size;
  {
    my $lex = $$params{ng_list()};
    $lex_size = @$lex;
  }
  if ($filter){
    $$params{ng_list()} = filter_keywords($params);
  }
  @link_table = @{initialize_link_array(\@link_table, $nr_docs)};
  my $ngs = $$params{ng_list()};
  foreach(@{$ngs}){
    my $corpus_index = find_string($_, \&cmp_ix_ix);
    if (defined($corpus_index)){
      link_docs ($corpus_index, $$_{ng_length()});
    }
  }
  my $time = times()/60;
  printprog("Duration so far (minutes): $time\n");
  printprog("Collection-linked.\n");
}

# Links a collection of documents directly after calculating
# TF, DF, RIDF and MI. Hence, thresholds must be known in advance.
# Optionally creates some output files with the results.
# PARAMETERS:
# Ref to hash with the following keys (see constants):
# page_dir => collection directory (should not end with slash),
# lex_file => output file for frequency lexicon (optional),
# link_cr => output file for CrossRef link adjacency matrix,
# keyw_file => output file for mere keywords (optional),
# pid_cr => output file for CrossRef page identification data
(optional),
# limit => maximal length of an N-gram,
# ridf_min => RIDF threshold (minimum RIDF)(optional),
# df_min => DF threshold (minimum DF) (optional),
# mi_max => MI threshold (maximum MI) (optional).
# print_prog => Flag: Print progress info at runtime?
# SIDE EFFECT:
# Selects STDOUT.
sub link_directly {
  my $params = shift @_;
  if ($$params{print_prog()}){
    $print_info = $$params{print_prog()};
  }
  my $collection = load_collection($params);
  printprog("Linking collection...\n");
  my $linkfile = $$params{link_cr()};
  if (defined($$params{stop_list()})){
    initialize_stops($params);
  }
  $$params{ng_list()} = generate_frequency_lexicon($collection,
                                                   $$params{limit()});
  if (defined($$params{lex_file()})){
    create_freq_lex_file($$params{lex_file()}, $$params{ng_list()});
  }
  my $time = times()/60;
  printprog("Duration so far (minutes): $time\n");
  $$params{ng_list()} = filter_keywords($params);
  @link_table = @{initialize_link_array(\@link_table, $nr_docs)};
  my $ngs = $$params{ng_list()};
```

```
  foreach(@{$ngs}){
    my $corpus_index = find_string($_, \&cmp_ix_ix);
    if (defined($corpus_index)){
      if ($$params{store_lk()}){
        link_docs ($corpus_index, $$_{ng_length()}, $_);
      } else {
        link_docs ($corpus_index, $$_{ng_length()});
      }
    }
  }
  open LINKS, ">$linkfile"  or die "Couldn't open link file";
  select LINKS;
  print_link_array(sep);
  close LINKS;
  select STDOUT;
  if ($$params{keyw_file()}){
    $$params{keywords()} = ng_to_text($params);
    write_keywords_to_file($params);
  }
  if ($$params{pid_cr()}){
    my $pidfile = $$params{pid_cr()};
    open PIDCR, ">$pidfile" or die "Couldn't write PID data";
    select PIDCR;
    print_pid_data(sep);
    close PIDCR;
    select STDOUT;
  }
  if ($$params{html_dir()}){
    $$params{cr_matrix()} = \@link_table;
    $$params{cr_pid_map} = \%fn_to_doc;
    if ($$params{store_lk()}){
      $$params{linkey()} = \%linking_keywords;
    }
    create_html{$params};
  }
  $time = times()/60;
  printprog("Duration so far (minutes): $time\n");
  printprog("Collection-linked.\n");
}

# Filters a file with N-gram frequency
# data such that it outputs a file with
# mere key-N-grams.
# PARAMETERS:
# Hash with the following keys (constant):
# lex_file => N-gram lexicon file with TF, DF, RIDF, MI,...,
# df_min => minimum DF (optional),
# mi_max => maximum MI (optional),
# ridf_min => minimum RIDF,
# keyw_file => output file for keyword list.
sub create_keywords_from_lex {
  my $params = shift @_;
  $$params{ng_list()} = load_ngram_data($params);
  $$params{ng_list()} = filter_keywords($params);
  $$params{keywords()} = ng_to_text($params);
  write_keywords_to_file($params);
}
```

```
# Generates a frequency lexicon from a collection of files.
# PARAMETERS:
# Hash with the following keys (constants):
# page_Dir => directory with input files (text only).
# lex_file => output filename.
# limit => maximal length of output N-grams.
sub gen_freq_lex_from_files {
  my $params = shift @_;
  printprog("Generating lexicon from file...\n");
  my $collection = load_collection($params);
  printprog("Generating lexicon...\n");
  my $ngrams = generate_frequency_lexicon($collection,
                                          $$params{limit()});
  create_freq_lex_file($$params{lex_file()}, $ngrams);
  my $time = times()/60;
  printprog("Duration so far (minutes): $time\n");
}

# Generates file with frequeny data using
# Wiki articles retrieved online as corpus.
# PARAMETERS:
# Filename of a file with Wiki titles.
# Language of articles.
# Maximal length of resulting N-grams-
# Output filename.
sub gen_freq_lex_from_wiki {
  my $filename = shift @_; # File with Wiki titles.
  my $lang = shift @_;
  my $limit = shift @_;
  my $lexfile = shift @_;
  my @collection; # Training corpus.
  my $wiki; # Wikipedia.
  my $id = 0;
  open TITFILE, "<$filename" or die "Couldn't open $filename";
  $wiki = WWW::Wikipedia->new(language => "$lang");
  while(<TITFILE>){
    chomp($_);
    printprog("Getting article $_\n");
    my $result = $wiki->search($_);
    if (defined($result) && $result->text()) {
      push(@collection, $id);
      $fn_to_doc{$_} = $id;
      $id++;
    } else {
      warn "No text retrieved for article $_";
    }
  }
  close TITFILE;
  printprog("Articles: " . ($#collection + 1) . "\n"); # Testing only!
  my $ngrams = generate_frequency_lexicon(\@collection, $limit);
  create_freq_lex_file($lexfile, $ngrams);
  my $time = times()/60;
  printprog("Duration so far (in minutes): $time\n");
}
```

```
# Counts term frequencies (TF) and
# document frequencies (DF) in a given collection
# of texts.
# PARAMETERS:
# Reference to array of references
# to texts (the collection).
# Optional: maximal Term length
# (n of N-gram).
sub generate_frequency_lexicon {
  # INPUT:
  # Reference to an array of document references and IDs.
  # (docref, id, docref, id, docref, ...)
  my $collection = shift @_;
  unless (defined($collection)){
    die "No collection to process handed over";
  }
  unless (defined($_[0])){
    warn "No maximal n-value specified. N-grams can be large!";
  } else {
    $n_max = shift @_;
  }
  if (@_ > 0){
    warn "Obsolete arguments are ignored by generate_lexicon()";
  }
  # TOKENIZE COLLECTION:
  # Transform the collection into a long sequence ($corpus).
  # Tokenize each document.
  # Store the start and end indices of each token (in %token).
  # For each token store the document id (in %ix_to_doc).
  $word_boundary = word_boundary;
  $cur_ix = 0;
  while (@$collection){
    my $docref = shift @$collection; # Document reference.
    my $id = shift @$collection; # Document ID.
    tokenize($docref, $id);
  }
  printprog("Collection tokenized.\n"); # Testing only!
  # INITIALIZE HASHES AND ARRAYS:
  # Create a map from token to next token (%next_token).
  my $nr_tokens = keys(%token);
  printprog("Documents: $nr_docs\n");
  printprog("Tokens: $nr_tokens\n");
  printprog("Maximal Nr of output N-grams: " . max_nr_ngrams() . "\n");
  initialize_next_token();
  printprog("Next-token map initialized.\n"); # Testing only!
  # Initialize the suffix array and sort it (@suff_arr).
  initialize_suffix_array();
  printprog("Suffix array initialized.\n"); # Testing only!
  # For each suffix in the suffix array ($suff_arr),
  # calculate the length of the longest common prefix
  # of the suffix and the previous suffix in the array and
  # store it in the LCP array ($lcp) which is parallel to
  # the suffix array (except that it has one more position).
  initialize_lcp();
  printprog("LCP array initialized.\n"); # Testing only!
  # Calculate classes,
  # for all class members (N-grams) that don't exceed
  # $n_max, get N-gram (start index suffix array + length),
  # TF and DF.
  my $n_grams = calc_class_freqs();
  printprog("TF and DF calculated.\n"); # Testing only!
  # Add RIDF and MI (if length > 1).
  calc_measures($n_grams);
  printprog("RIDF and MI calculated.\n"); # Testing only!
  # Give it back.
  return $n_grams;
}
```

```
# Calculates MI as defined in Yamamoto/Church (1998:18).
# Applicable to N-grams with n > 1;
# PARAMETERS:
# Reference to a hash with n-gram frequencies:
# xYz => TF of the N-gram xYz,
# Y => TF of the N-gram y,
# xY => TF of the N-gram xY (>0),
# Yz => TF of the N-gram Yz (>0),
# where x and z are unigrams and Y is an
# N-gram of arbitrary length.
# RETURNS:
# Mutual information.
sub mi_yc {
  my $tf = shift @_;
  return log2($$tf{xYz()}*$$tf{Y()}/$$tf{xY()}*$$tf{Yz()});
}

# Calculates RIDF of an N-gram
# as it is defined in
# Manning/Schütze (1999:553).
# PARAMETERS:
# Term frequency.
# Document frequency (>0);
# Total number of documents (>0).
# RETURNS:
# Residual inverse document frequency.
sub ridf {
  my $tf = shift @_;
  my $df = shift @_;
  my $docs = shift @_;
  my $Ddf = $docs/$df;
  my $tfD = $tf/$docs;
  my $ridf = log2($Ddf)+log2(1 -1/exp($tfD));
  return $ridf;
}

# Logarithm with base 2.
# PARAMETERS:
# Number whose log to calculate.
# RETURNS:
# Logarithm of number in base 2.
sub log2 {
  my $num = shift;
  return log($num)/log(2);
}

# Logarithm with base 10.
# PARAMETERS:
# Number whose log to calculate.
# RETURNS:
# Logarithm of number in base 10.
sub log10 {
  my $num = shift;
  return log($num)/log(10);
}

# Calculates f-measure as suggested in Manning/Schütze (1999:269).
# PARAMETERS:
# Weighting factor alpha: 0<=alpha<=1,
# alpha==0.5 for equal weighting,
# alpha>0.5 for higher weighting of precision,
# alpha<0.5 for higher weighting of recall.
# Precision (>0).
# Recall (>0).
# RETURNS:
# F-value.
# PRESUPPOSES:
# Both precision and recall > 0.
sub f_measure {
  my ($alpha, $prec, $rec) = @_;
  return 1/(($alpha/$prec)+((1-$alpha)/$rec));}
```

```
# Calculates precision as suggested in
# Manning/Schütze (1999:268).
# PARAMETERS:
# Nr of true positives.
# Nr of false positives.
# RETURNS:
# Precision value.
# PRESUPPOSES:
# Sum of parameters > 0.
sub precision {
  my ($tp, $fp) = @_;
  return $tp/($tp+$fp);
}

# Calculates recall as suggested in
# Manning/Schütze (1999:269).
# PARAMETERS:
# Nr of true positives.
# Nr of false negatives.
# RETURNS:
# Recall value.
# PRESUPPOSES:
# Sum of parameters > 0.
sub recall {
  my ($tp, $fn) = @_;
  return $tp/($tp+$fn);
}

# Calculates fallout as suggested in
# Manning/Schütze (1999:270).
# PARAMETERS:
# Nr of false positives.
# Nr of true negatives.
# RETURNS:
# Fallout value.
# PRESUPPOSES:
# Sum of parameters > 0.
sub fallout {
  my ($fp, $tn) = @_;
  return $fp/($fp+$tn);
}

# Initializes a square link adjacency matrix
# with 0-values.
# PARAMETERS:
# Reference to array to be initialized.
# Number of documents to be linked.
# PRESUPPOSES:
# tokenize();
# RETURNS:
# Reference to empty matrix.
sub initialize_link_array {
  my $lt = shift @_;
  my $docs = shift@_;
  my @arr = ();
  $lt = \@arr;
  foreach (0..($docs-1)){
    my @table_row = ();
    foreach (0..($nr_docs-1)){
      $table_row[$_] = 0;
    }
    $$lt[$_] = \@table_row;
  }
  return $lt;
}
```

```perl
# Initializes the suffix array with tokens
# (= start indices of suffixes).
# Sorts the suffix array alphabetically.
# PRECONDITION:
# &tokenize();
sub initialize_suffix_array {
  unless (defined($corpus)){
    die "No corpus defined. Probably forgot to tokenize";
  }
  foreach (sort numeric keys(%token)){ # Sort necessary?
    push(@suff_arr, $_);
  }
  @suff_arr = sort suffix_tokens_alphabetic @suff_arr; # !!!
}

# Initializes the next-token map.
# PRECONDITION:
# &tokenize();
sub initialize_next_token {
  my $prev; # Previous token.
  foreach (sort numeric keys(%token)){
    if (defined $prev){
      $next_token{$prev} = $_;
    }
    $prev = $_;
  }
}

# Initializes the LCP-array.
# PRECONDITION:
# &initialize_suffix_array().
# &initialize_next_token().
sub initialize_lcp {
  $lcp[0] = 0; # Always 0.
  foreach (0..($#suff_arr-1)){
    my $lcpl = lcp_length($suff_arr[$_], $suff_arr[$_+1]);
    $lcp[$_+1] = $lcpl;
  }
  $lcp[$#lcp+1] = 0; # Always 0.
}

# Initializes stopword list.
# PARAMETERS:
# Hash with the following key (constant):
# stop_list => filename of stopword list.
sub initialize_stops{
  my $params = shift @_;
  my $list = load_list($$params{stop_list()}, 0);
  @stopwords = @$list;
}

# Counts the number of matching and respectively non-matching
# links in the CrossRef link matrix and a comparison link matrix
# (probably the original one).
# PARAMETERS:
# Reference to comparison link matrix.
# Reference to CrossRef link matrix.
# Reference to original PID-hash.
# Reference to CrossRed PID-hash.
# RETURNS:
# Reference to a hash that contains the counts for
# true and false positives and true and false negatives.
# Also contains performance values: precision, recall, fallout.
sub match_links {
  my ($links_o, $links_cr, $pid_o, $pid_cr) = @_;
  my %matches = (
    true_pos() => 0, # True positives.
    true_neg() => 0, # True negatives.
    false_pos() => 0, # False positives.
    false_neg() => 0, # False negatives.
  );
```

```
  printprog("Matching links against gold-standard...\n");
  foreach (keys(%$pid_cr)){
    if (defined($$pid_o{$_}) && defined($$pid_cr{$_})){
      my $key_row_o = $$pid_o{$_}; # Row index in original matrix.
      my $key_row_cr = $$pid_cr{$_}; # Row index in CrossRef matrix.
      my $row_o = ${$links_o}[$key_row_o]; # Row array in original.
      my $row_cr = ${$links_cr}[$key_row_cr]; # Row array in CrossRef.
      foreach (keys(%$pid_cr)){
        if (defined($$pid_o{$_}) && defined($$pid_cr{$_})){
          my $key_col_o = $$pid_o{$_}; # Column index in original.
          my $key_col_cr = $$pid_cr{$_}; # Column index in CrossRef.
          my $val_o = ${$row_o}[$key_col_o]; # Link value in original.
          my $val_cr = ${$row_cr}[$key_col_cr]; # Link value Crossref.
          if ($val_cr > 0 && $val_o > 0){
            $matches{true_pos()}++;
          } elsif ($val_cr == 0 && $val_o == 0){
            $matches{+true_neg()}++;
          } elsif ($val_cr > 0 && $val_o == 0){
            $matches{+false_pos()}++;
          } elsif ($val_cr == 0 && $val_o > 0){
            $matches{+false_neg()}++;
          } else {
            die "Bad values!";
          }
        } else {
          warn "Cols: complementary document $_";
          next;
        }
      }
    } else {
      warn "Rows: complementary document $_";
      next;
    }
  }
  # Calculate performance measures...
  if (($matches{true_pos()} + $matches{false_pos()}) > 0){
    $matches{prec()} = precision($matches{true_pos()},
                                 $matches{false_pos()});
  }
  if (($matches{true_pos()} + $matches{false_neg()}) > 0){
    $matches{rec()} = recall($matches{true_pos()},
                             $matches{false_neg()});
  }
  if (($matches{false_pos()} + $matches{true_neg()}) > 0){
    $matches{fall()} = fallout($matches{false_pos()},
                               $matches{true_neg()});
  }
  return \%matches;
}

# Links docs that contain the same keyword N-gram.
# PARAMETERS:
# Start index of one keyword (N-gram) containing substring
# in suffix array.
# Keyword (N-gram) length;
# Optional: ref to connecting N-gram to store it
# (for testing and evaluation).
sub link_docs {
  my $doc1 = shift @_;
  my $length = shift @_;
  my $ngram;
  if (@_){
    $ngram = shift @_;
  }
  my $doc2;
  # Move to the FIRST occurrence of the N-gram in the
  # suffix array.
  while($lcp[$doc1] >= $length){
    $doc1--;
  }
```

```
    # For each doc, create symmetric links to other
    # docs containing the same key N-Gram.
    while($lcp[$doc1+1] >= $length){
      $doc2 = $doc1;
      while($lcp[++$doc2] >= $length){
        # Link (only) distinct documents.
        unless ($ix_to_doc{$suff_arr[$doc1]} ==
                $ix_to_doc{$suff_arr[$doc2]}){
          ${$link_table[$ix_to_doc{$suff_arr[$doc1]}]}
            [$ix_to_doc{$suff_arr[$doc2]}]++;
          ${$link_table[$ix_to_doc{$suff_arr[$doc2]}]}
            [$ix_to_doc{$suff_arr[$doc1]}]++;
          if (defined($ngram)){
            # Store also the linking keywords.
            my $keyw_key = $ix_to_doc{$suff_arr[$doc1]}
                   . sep . $ix_to_doc{$suff_arr[$doc2]};
            my $kw_list;
            if (exists($linking_keywords{$keyw_key})){
              $kw_list = $linking_keywords{$keyw_key};
            } else {
              my @arr = ();
              $kw_list = \@arr;
            }
            my $contained = 0;
            foreach (@$kw_list){
              if ($_ == $ngram){
                $contained++;
                last;
              }
            }
            unless ($contained){
              push(@$kw_list, $ngram);
            }
            $linking_keywords{$keyw_key} = $kw_list;
            $keyw_key = $ix_to_doc{$suff_arr[$doc2]}
                   . sep . $ix_to_doc{$suff_arr[$doc1]};
            $linking_keywords{$keyw_key} = $kw_list;
          }
        }
      }
      $doc1++;
    }
  }

  # Iterates over an N-gram array and, for
  # each N-gram, adds RIDF and MI if defined
  # to the associated data.
  # PARAMETERS:
  # Reference to N-gram array.
  # RETURNS:
  # Reference to N-gram array augmented
  # with RIDF and MI data.
  sub calc_measures {
    my $ngr = shift @_;
    my %ngrh = ();
    for (@$ngr){
      # Store for quick access...
      $ngrh{lc(token_text($$_{ng_start()}))
          . "_" . $$_{ng_length()}} = $_;
    }
    foreach (@$ngr){
      # Get RIDF...
      $$_{ng_ridf()} = ridf($$_{ng_tf()}, $$_{ng_df()}, $nr_docs);
      # Get MI...
      if ($$_{ng_length()} >= 2){
        my %tf = ();
        my $xg;
        $tf{xYz()} = $$_{ng_tf()};
```

```
# Get TF for Y...
if ($$_{ng_length()} > 2){
   $xg = $ngrh{lc(token_text($next_token{$$_{ng_start()}}))
         . "_" . ($$_{ng_length()} - 2)};
   $tf{Y()} = $$xg{ng_tf()};
} else {
   $tf{Y()} = keys %token;
}
# Get TF for xY...
$xg = $ngrh{token_text($$_{ng_start()}) . "_"
      . ($$_{ng_length()} - 1)};
$tf{xY()} = $$xg{ng_tf()};
# Get TF for Yz...
$xg = $ngrh{token_text($next_token{$$_{ng_start()}})
      . "_" . ($$_{ng_length()} - 1)};
$tf{Yz()} = $$xg{ng_tf()};
my $zeros = 0;
# TODO:
# There seems to be a minor bug.
# (But it occurred only once in manymany trials.)
# Fix it!
foreach (keys %tf){
   if ($tf{$_} == 0){
      $zeros = 1;
   }
}
# Actually, there shouldn't be any 0s.
# -> Tokenization or string matching?
# However, since MI is a feature likely to be removed
# from this code, I'll postpone the problem.
# TODO: Fix this!
unless ($zeros){
   $$_{ng_mi()} = mi_yc(\%tf);
} else {
   $$_{ng_mi()} = no_mi_val();
}
         }
      }
   }

   # Calculates term frequencies
   # and document frequencies
   # for classes.
   # RETURNS:
   # Reference to a list hashes with N-gram data.
   # Hash keys (constants):
   # ng_start => Corpus start index.
   # ng_length => Length.
   # ng_tf => Term frequency (TF).
   # ng_df => Document frequency (DF).
   # PRESUPPOSES:
   # initialize_lcp();
   sub calc_class_freqs {
      my @stack_start; # Stack of left edges.
      my @stack_rep; # Stack of representative.
      my @stack_df; # Document frequency stack.
      my @doc_link; # Most recently processed suffix at doc index.
      my $sp = 1; # Stack pointer.
      my @intervals; # Array of intervals.
      push(@stack_start, 0);
      push(@stack_rep, 0);
      push(@stack_df, 1);
      foreach (0..$#suff_arr){
         # List a trivial interval.
         # Interval data:
         # $intv_data[0]: Start index on suffix array.
         # $intv_data[1]: Last element index.
         # $intv_data[2]: LBL.
         # $intv_data[3]: SIL (undef if trivial).
         # $intv_data[4]: Term frequency.
         # $intv_data[5]: Document frequency.
```

```
      my @intv_data = ($_, $_, lbl($_, $_), undef, 1, 1);
      if ($intv_data[5] >= abs_df_min){
        push(@intervals, \@intv_data);
      }
      my $doc = $ix_to_doc{$suff_arr[$_]};
      if (defined($doc_link[$doc])){
        my $beg = 0;
        my $end = $sp;
        my $x = int($end/2);
        while ($beg != $x){
          if ($doc_link[$doc] >= $stack_start[$x]){
            $beg = $x;
          } else {
            $end = $x;
          }
          $x = int(($beg + $end)/2);
        }
        $stack_df[$x] = $stack_df[$x] - 1;
      }
      $doc_link[$doc] = $_;
      my $d_freq = 1; # Document frequency.
      while ($lcp[$_+1] < $lcp[$stack_rep[$sp-1]]){
        $d_freq = $stack_df[$sp-1] + $d_freq;
        if (is_lcp_delimited($stack_start[$sp-1], $_)){
          my @intv_data2 = ($stack_start[$sp-1], # Start of interval.
                  $_, # Last element of interval.
                  lbl($stack_start[$sp-1], $_), # Longest bound LCP.
                  sil($stack_start[$sp-1], $_), # Shortest inner LCP.
                  ($_ - $stack_start[$sp-1] + 1), # Term frequency.
                  $d_freq); # Document frequency.
          if ($intv_data2[5] >= abs_df_min){
            push(@intervals, \@intv_data2);
          }
        }
        $sp--;
      }
      $stack_start[$sp] = $stack_rep[$sp-1];
      $stack_rep[$sp] = $_ + 1;
      $stack_df[$sp] = $d_freq;
      $sp++;
    }
    # Get N-grams with TF and DF from classes.
    return class_n_grams(\@intervals);
  }

  # Turns interval data into N-gram data.
  # PARAMETERS:
  # Reference to an array with interval data.
  # RETURNS:
  # Reference to a list with N-gram data.
  # PRESUPPOSES:
  # Supposed to be called by &calc_class_freqs.
  sub class_n_grams {
    my $intervals = shift @_; # Interval data.
    my @n_grams; # Array of N-grams of the input class.
    foreach (@$intervals){
      my $intv_data = $_;
      my $m;
      if (defined($$intv_data[3])){
        if (defined($n_max) && ($$intv_data[3] > $n_max)){
          $m = $n_max;
        } else {
          $m = $$intv_data[3];
        }
      } else { # Very long, extends till the end of the document.
        if (defined $n_max){
          $m = suffix_length_max($suff_arr[$$intv_data[0]], $n_max);
        } else {
          $m = suffix_length($suff_arr[$$intv_data[0]]);
        }
      }
```

```
    # Get class members, store data...
    while ($m > $$intv_data[2]){
      my %n_gram = (
        ng_start() => $suff_arr[$$intv_data[0]], # N-gram start corpus.
        ng_length() => $m, # N-gram length;
        ng_tf() => $$intv_data[4], # Term frequency.
        ng_df() => $$intv_data[5], # Document frequency.
      );
      push(@n_grams, \%n_gram);
      $m--;
    }
  }
  return \@n_grams;
}

# Checks if an interval is LCP-delimited.
sub is_lcp_delimited {
  my $ix_first = shift @_; # Start of interval on suffix array.
  my $ix_last = shift @_; # Last element of interval on suffix array.
  return (lbl($ix_first, $ix_last) < sil($ix_first, $ix_last));
}

# Finds the longest bounding LCP (LBL)
# of the input interval.
# PARAMETERS:
# Start index of interval on suffix array.
# Index of last element of interval on suffix array.
# RETURNS:
# The larger one of the 2 bounding LCPs (its length).
# PRESUPPOSES:
# initialize_lcp();
sub lbl {
  my $ix_first = shift @_; # Start of interval on suffix array.
  my $ix_last = shift @_; # Last element of interval on suffix array.
  my $lcp_first = $lcp[$ix_first];
  my $lcp_after = $lcp[$ix_last+1];
  if ($lcp_after > $lcp_first){
    return $lcp_after;
  } else {
    return $lcp_first;
  }
}

# Finds the shortest internal LCP (SIL)
# of an interval on the suffix array.
# PARAMETERS:
# Start index of interval on suffix array.
# Index of last element of interval on suffix array.
# RETURNS:
# The shortest of the interior LCPs (its length).
# PRESUPPOSES:
# initialize_lcp();
sub sil {
  my $ix_first = shift @_; # Start of interval on suffix array.
  my $ix_last = shift @_; # Last element of interval on suffix array.
  if ($ix_first >= $ix_last){
    return undef; # Infinity.
  }
  my $min = $lcp[$ix_last];
  $ix_first++;
  while ($ix_first < $ix_last){
    if ($lcp[$ix_first] < $min){
      $min = $lcp[$ix_first];
    }
    $ix_first++;
  }
  return $min;
}
```

```
# Finds an occurrence of the input words in
# the corpus.
# PARAMETERS:
# Reference to array of words.
# Reference to comparison function:
# \&cmp_txt_ix to compare words to corpus content.
# \&cmp_ix_ix to compare corpus content to other corpus content.
# RETURNS:
# Index of found match in suffix array.
# -1 if the sought string is not contained.
# PRESUPPOSES:
# initialize_suffix_array();
sub find_string {
  my $ng = shift @_;
  my $cmp_func = shift @_;
  my $beg = 0;
  my $end = @suff_arr;
  my $mid = int($end/2);
  my $cmp_val = $cmp_func->($ng, $mid);
  while ($cmp_val != 0){
    if ($cmp_val < 0){
      $end = $mid;
    } else {
      if ($mid <= $beg){
        return no_val; # Not contained.
      }
      $beg = $mid;
    }
    $mid = $beg + int(($end - $beg)/2);
    $cmp_val = $cmp_func->($ng, $mid);
  }
  return $mid;
}

# Finds an N-gram with a specific start index and a specific length in
# an array of N-grams. Binary search.
# PARAMETERS:
# Array with references to N-grams.
# Searched start index (word).
# Searched length.
# RETURNS:
# Reference to the appropriate N-gram.
# PRECONDITION:
# The N-gram list actually contains the
# sought N-gram.
sub find_ng_ix_l {
  my $ngr = shift @_; # Reference to N-gram array.
  my $start = shift @_; # Searched start index.
  my $l = shift @_; # Searched length.
  my $beg = 0;
  my $end = @$ngr;
  my $mid = int($end/2);
  my $gram = $$ngr[$mid];
  while (!($$gram{ng_start()} == $start && $$gram{ng_length()} == $l)){
    if($$gram{ng_start()} == $start){
      if ($$gram{ng_length()} > $l){
        $end = $mid;
      } else {
        $beg = $mid;
      }
    } else {
      if ($$gram{ng_start()} > $start){
        $end = $mid;
      } else {
        $beg = $mid;
      }
    }
    $mid = int(($beg + $end)/2);
    $gram = $$ngr[$mid];
  }
  return $gram;}
```

```
# Calculates the length of a suffix
# from the start index in the corpus
# to the end of the containing document.
# PARAMETERS:
# Start index of suffix in corpus.
# RETURNS:
# Length of suffix (bounded by end of doc).
# PRESUPPOSES:
# initialize_next_token();
sub suffix_length {
  my $suffix = shift @_;
  my $l = 1;
  while (!is_doc_end($suffix)){
    if (defined($next_token{$suffix})){
      $suffix = $next_token{$suffix};
      $l++;
    } else {
      last;
    }
  }
  return $l;
}

# Calculates the length of a suffix
# where either the document end or the
# specified max value provide an upper
# bound.
# PARAMETERS:
# Start index of suffix in corpus.
# Maximal length.
# RETURNS:
# Length of suffix (bounded by end of doc
# or maximal length).
# PRESUPPOSES:
# initialize_next_token();
sub suffix_length_max {
  my $suffix = shift @_;
  my $l_max = shift @_;
  my $l = 1;
  while (!is_doc_end($suffix) && $l < $l_max){
    if (defined($next_token{$suffix})){
      $suffix = $next_token{$suffix};
      $l++;
    } else {
      last;
    }
  }
  return $l;
}

# Finds the longest common prefix of 2 sequences starting with the input
# tokens (= start indices). Stops at document boundaries! Ignores case.
# PARAMETERS:
# Start token of suffix 1.
# Start token of suffix 2.
# RETURNS:
# Length of the LCP of the input suffixes.
sub lcp_length {
  my $suff1 = shift @_; # Suffix.
  my $suff2 = shift @_; # Other suffix.
  my $l = 0; # LCP length.
  while (token_text($suff1) eq token_text($suff2)){
    if (is_doc_end($suff1) || is_doc_end($suff2)){
      last;
    }
    $l++;
    $suff1 = $next_token{$suff1};
    $suff2 = $next_token{$suff2};
  }
  return $l;
}
```

```
# Checks if the input word is contained
# in the stopword list.
# PARAMETERS:
# Word to be checked (lowercase).
# RETURNS:
# 1 if the word is a designated stopword,
# 0 otherwise.
sub is_stop {
  my $word = shift @_;
  unless (@stopwords > 0){
    return 0;
  }
  my $stops = grep {$word eq $_} @stopwords;
  if ($stops > 0){
    return 1;
  } else {
    return 0;
  }
}

# Checks if a token is the last token of
# a document (Document boundary?).
# RETURNS:
# TRUE iff so.
# PARAMETERS:
# Token to check (start ID).
# PRESUPPOSES:
# &tokenize().
sub is_doc_end {
  my $tok = shift @_;
  unless (defined($next_token{$tok})){
    return 1;
  }
  return ($ix_to_doc{$tok} != $ix_to_doc{$next_token{$tok}});
}

# Returns the text of the token starting at
# the input index.
# PARAMETERS:
# Token start index.
# PRECONDIZION:
# $tokenize();
sub token_text {
  my $tox = shift @_; # Token start index;
  return substr($corpus, $tox, ($token{$tox} - $tox));
}

# Accessor for the text units of an N-gram.
# Ignores document boundaries!
# PARAMETERS:
# Start index of N-gram in corpus.
# Length of the n-gram (nr of units).
# RETURNS:
# Reference to an array of words that constitute
# the N-gram text.
# PRESUPPOSES:
# initialize_next_token();
sub n_gram_text {
  my $start = shift @_; # Start index in corpus.
  my $n_val = shift @_; # N-gram length.
  unless (defined($start) && defined($n_val)){
    die "Bad arguments [$start] [$n_val]";
  }
  my @n_gram_text; # List of words to be returned.
  push(@n_gram_text, token_text($start));
  foreach (2..$n_val){
    if (defined($next_token{$start})){
      $start = $next_token{$start};
      push(@n_gram_text, token_text($start));
    } else {
```

```
        warn "End of corpus: $start?";
        my $d9 = <STDIN>;
        last;
      }
    }
    return \@n_gram_text;
}

# Gets a list of mere keywords
# from a list of corpus-index form
# N-grams.
# PARAMETERS:
# Hash with key:
# ng_list => list of raw N-grams.
# RETURNS:
# Reference to a list of word-arrays.
sub ng_to_text {
    my $params = shift @_;
    my $ngs = $$params{ng_list()};
    my @wordforms = ();
    foreach (@$ngs){
      unless ($$_{ng_txt()}){
        my $txt = n_gram_text($$_{ng_start()}, $$_{ng_length()});
        push(@wordforms, $txt);
      } else {
        push(@wordforms, $$_{ng_txt});
      }
    }
    return \@wordforms;
}

# This sub tokenizes the input text such that
# it appends the text to a long sequence of documents
# which is used by the suffix array,
# it stores the inices of beginnings and ends of tokens
# (relative to that long sequence) in a hash and
# it creates a mapping from tokens (that means from the
# beginning of a token) to the ID of the document from
# which the token was retrieved.
# PARAMETERS:
# $text reference to the document text
# $text_id document ID
sub tokenize {
    my $text = shift @_;
    my $text_id = shift @_;
    my $end_of_doc_length = length(end_of_doc); # Length of doc delimitor.
    $nr_docs++;
    while ($$text =~ s/((\w+?)$word_boundary)//){
      my $str = $1; # Token text + delimiter(s).
      my $toxt = $2; # Token text.
      unless (is_stop(lc($toxt))){
        $token{$cur_ix} = $cur_ix + length($toxt); # Token-end.
        $ix_to_doc{$cur_ix} = $text_id;
      }
      $corpus .= lc($str);
      $cur_ix += length($str);
      if ($$text eq ""){
        $corpus .= end_of_doc;
        $cur_ix += $end_of_doc_length;
      }
    }
}
```

```
# Compares the words in the input array
# to words in the corpus.
# PARAMETERS:
# Reference to array of words.
# Start index of comparison string in corpus.
# RETURNS:
# 0 iff equal.
# < 0 if words < corpus string.
# > 0 otherwise.
# PRESUPPOSES:
# initialize_suffix_array();
# initialize_next_token();
sub cmp_txt_ix {
  my @words = @{shift @_};
  my $ix = shift @_;
  my $length = @words;
  $ix = $suff_arr[$ix];
  my $cmp_val;
  foreach (0..$#words){
    $cmp_val = $words[$_] cmp token_text($ix);
    unless ($cmp_val == 0){
      return $cmp_val;
    }
    $ix = $next_token{$ix};
  }
  return $cmp_val;
}

# Compares the words in the corpus
# specified by index in the input array
# to other words in the corpus.
# PARAMETERS:
# Reference to N-gram.
# Start index of comparison string in corpus.
# RETURNS:
# 0 iff equal.
# < 0 if words < corpus string.
# > 0 otherwise.
# PRESUPPOSES:
# initialize_suffix_array();
# initialize_next_token();
sub cmp_ix_ix {
  my $ng = shift @_;
  my $corp_ix = shift@_;
  $corp_ix = $suff_arr[$corp_ix];
  my $ng_ix = $$ng{ng_start()};
  my $cmp_val;
  for (1..$$ng{ng_length()}){
    $cmp_val = token_text($ng_ix) cmp token_text($corp_ix);
    unless($cmp_val == 0){
      return $cmp_val;
    } else {
      $corp_ix = $next_token{$corp_ix};
      $ng_ix = $next_token{$ng_ix};
    }
  }
  return $cmp_val;
}

# Sorts links in descending order by the number
# of shared terms.
sub links_by_nr_shared_keywords {
  return $$b[0] <=> $$a[0];
}

# Sort N-grams by RIDF (descending).
sub n_grams_by_ridf {
  $$b{ng_length()} <=> $$a{ng_length()};
}
```

```perl
# Sort N-grams by length.
sub n_grams_length {
  $$a{ng_length()} <=> $$b{ng_length()};
}

# Numeric sort sequence for N-gram references.
sub n_grams_numeric {
  $$a{ng_start()} <=> $$b{ng_start()};
}

# Numeric sort sequence.
sub numeric {
  $a <=> $b;
}

# Sorts alphabetically but descending.
sub alpha_descending {
  $b cmp $a;
}

# Compares the substrings beginning at the
# indices handed over.
# Case is ignored!
# PRECONDIZION:
# tokenize();
sub suffixes_alphabetic {
  substr($corpus, $a) cmp substr($corpus, $b);
}

# Compares the token-substrings beginning at the
# indices handed over.
# Case is ignored!
# Since there are no 2 substrings
# extending to corpus end with  the
# same length in the corpus, if one substring
# ends, it is the shorter one and comes first.
# PRECONDIZION:
# tokenize();
# initialize_next_token();
sub suffix_tokens_alphabetic {
  my $first = $a;
  my $sec = $b;
  if ($first == $sec){
    return 0;
  } else {
    my $cmp_val = token_text($first) cmp token_text($sec);
    while ($cmp_val == 0){
      if (defined($next_token{$first})){
        $first = $next_token{$first};
      } else {
        return -1;
      }
      if (defined($next_token{$sec})){
        $sec = $next_token{$sec};
      } else {
        return 1;
      }
      $cmp_val = token_text($first) cmp token_text($sec);
    }
    return $cmp_val;
  }
}
```

```
# Accessor for the number of tokens per document.
# Needed for testing purposes.
# RETURNS:
# Reference to array with
# number of tokens per doc (index = doc ID).
# PRESUPPOSES:
# tokenize();
sub tok_per_doc {
  my @d = ();
  push(@d, 0);
  for (sort numeric keys(%ix_to_doc)){
    unless ($#d == $ix_to_doc{$_}){
      push(@d, 0);
    }
    $d[$#d]++;
  }
  return \@d;
}

# Calculates the maximum number of N-grams
# from token data.
# Since here, N-grams are
# supposed to terminate at document ends,
# The total number of N-Grams is the sum
# over the number of substrings for each
# document.
# RETURNS:
# Maximal number of countable substrings
# in the corpus.
# PRESUPPOSES:
# tokenize();
sub max_nr_ngrams {
  my $tok_doc = tok_per_doc();
  my $sum = 0;
  for (@$tok_doc){
    my $nr_substrings = $_*($_+1)/2;
    $sum += $nr_substrings;
  }
  return $sum;
}


# Turns a matrix representing a directed graph
# into a matrix representing an undirected graph
# such that it creates a backlink for each
# link in the matrix.
# PARAMETERS:
# Reference to adjacency matrix array.
sub symmetrize {
  my $matrix = shift;
  foreach (0..$#{$matrix}){
    my $row = $_;
    foreach(0..$#{$$matrix[$row]}){
      my $col = $_;
      if (${$$matrix[$row]}[$col] > 0){
        ${$$matrix[$col]}[$row]++;
      }
    }
  }
}

# Creates a simple keyword list from N-gram data.
# PARAMETERS:
# Hashwith the following key:
# ng_list => reference to N-gram array.
# RETURNS:
# Reference to list of key N-gram hashes.
sub filter_keywords {
  my $params = shift @_;
  my $ngrams = $$params{ng_list()};
  my @keywords = ();
```

```
  foreach (@$ngrams){
    if (defined($$params{mi_max()}) && ($$_{ng_mi()}
        > $$params{mi_max()})){
      next;
    }
    if (defined($$params{df_min()}) && ($$_{ng_df()}
        < $$params{df_min()})){
      next;
    }
    if (defined($$params{ridf_min()}) && ($$_{ng_ridf()}
        < $$params{ridf_min()})){
      next;
    }
    if (defined($$params{length_min()})){
      if (defined($$_{ng_length()}) && $$_{ng_length()}
          < $$params{length_min()}){
        next;
      } elsif (defined($$_{ng_txt()})){
        my $length = @{$$_{ng_txt()}};
        if ($length < $$params{length_min()}){
          next;
        }
      }
    }
    push(@keywords, $_);
  }
  return \@keywords;
}

# Pretty printer for the suffix
# array ($suff_arr).
# PRESUPPOSES:
# &initialize_suffix_array().
sub print_sa_pretty {
  my $l = shift @_; # Number of tokens to print.
  foreach (0..$#suff_arr){
    my $sux = $suff_arr[$_]; # Token start index.
    print "[$_][$sux]:[";
    print token_text($sux);
    foreach (2..$l){
      unless (defined($next_token{$sux})){
        next;
      }
      print " ";
      $sux = $next_token{$sux};
      print token_text($sux);
    }
    print "]\n";
  }
}

# Pretty printer for token data.
# PARAMETERS:
# Name of sort order:
# "numeric" (for order as in corpus) or
# "suffix_tokens_alphabetic".
# PRESUPPOSES:
# &tokenize().
sub print_token_data_pretty {
  my $order = shift @_; # Sort order.
  print "[TOKEN][BEGIN][END][DOC ID]\n";
  foreach (sort $order keys(%token)){
    print "[" . token_text($_) . "]"; # Token.
    print "[$_]"; # Token start.
    print "[$token{$_}]"; # Token end.
    print "[$ix_to_doc{$_}]\n"; # Document ID.
  }
}
```

```
# Prints the LCP-data pretty. Well...
# at least a bit less ugly.
# PRECONDITION:
# &initialize_lcp();
sub print_lcp_data_pretty {
  for (0..$#suff_arr){
    my $suff = $suff_arr[$_];
    my $lcpl = $lcp[$_];
    print "[$_]: $suff_arr[$_]: ";
    for (1..$lcpl){
      print token_text($suff) . " ";
      $suff = $next_token{$suff};
    }
    print "| ";
    print token_text($suff) . " ";
    if (defined($next_token{$suff})){
      print token_text($next_token{$suff}) . "... ";
    } else {
      print ". ";
    }
    print "$lcpl\n";
  }
}

# Prints LCP-delimited intervals (LDIs)
# on the suffix array pretty.
sub print_ldis_pretty {
  my $ix_first = shift @_; # Start of interval on suffix array.
  my $ix_rep = shift @_; # Index of a representative.
  my $ix_last = $ix_rep; # Index of last element.
  print_interval($ix_rep, $ix_rep);
  while (($lcp[$ix_rep] <= $lcp[$ix_last+1])&&(($ix_last+1)<@suff_arr)){
    $ix_last = print_ldis_pretty($ix_rep, ($ix_last+1));
    if (is_lcp_delimited($ix_first, $ix_last)){
      print_interval($ix_first, $ix_last);
    }
  }
  return $ix_last;
}

# Prints an interval on the suffix array, pretty.
# For each item: LCP and following word + length of LCP.
# PARAMETERS:
# When called externally:
# (0,0).
# When called recursively:
# Start index of interval on suffix array.
# Index of last element of interval on suffix array.
# PRESUPPOSES:
# initialize_lcp();
sub print_interval {
  my $ix_first = shift @_; # Start of interval on suffix array.
  my $ix_last = shift @_; # Last element of interval on suffix array.
  for ($ix_first..$ix_last){
    print "[$_]: $suff_arr[$_]: ";
    print_n_gram($suff_arr[$_], ($lcp[$_]+1));
    print " | $lcp[$_]\n";
  }
  print "------------------------\n";
}
```

```
# Prints the link array (almost pretty).
# PARAMETERS:
# Column separating string (f.e. ';' or ' ').
# PRESUPPOSES:
# initialize_link_array();
# link_docs();
sub print_link_array {
  my $sep = shift @_;
  foreach (0..$#link_table){
    my $table_row = $link_table[$_];
    foreach(@$table_row){
      print $_ . $sep;
    }
    if ($_ < $#link_table){
      print "\n";
    }
  }
}

# Prints page identification data
# (the mapping from filename to doc ID
# in %fn_to_doc).
# PARAMETERS:
# Column separating string.
# PRESUPPOSES:
# load_collection();
sub print_pid_data {
  my $sep = shift @_;
  foreach(keys(%fn_to_doc)){
    print "$fn_to_doc{$_}$sep$_$sep\n";
  }
}

# Prints an N-Gram.
# PARAMETERS:
# Start index on suffix array.
# Length.
# PRESUPPOSES:
# initialize_next_token();
sub print_n_gram {
  my $start = shift @_; # start of the n-gram.
  my $length = shift @_; # Length;
  print token_text($start);
  for (2..$length){
    if (defined($next_token{$start})){
      $start = $next_token{$start};
      print " " . token_text($start);
    } else {
      last;
    }
  }
}

# Prints progress info to STDOUT if allowed.
# PARAMETERS:
# String to be printed.
# SIDE-EFFECT:
# May select STDOUT.
sub printprog {
  my $info = shift @_;
  if ($print_info){
    select STDOUT;
    print $info;
  }
}
```

```perl
# Creates an evaluation sheet for manual evaluation
# of false positives in CSV-format.
# PARAMETERS:
# Hash with the following keys (constants):
# cr_matrix => reference to link matrix,
# cr_pid_map = reference to PID map,
# o_matrix => reference to comparison link matrix (optional),
# p_link_pid => reference to comparison PID (obligatory if o_matrix),
# eval_sheet => output file for evaluation sheet.
# SIDE-EFFECT:
# Selects STDOUT.
sub create_eval_sheet {
  my $params = shift @_;
  my $cr_links = $$params{cr_matrix()};
  my $o_links = $$params{o_matrix()};
  my $o_pid = $$params{o_pid_map()};
  my $cr_pid = $$params{cr_pid_map()};
  my $evalfile = $$params{eval_sheet()};
  my %links_by_name;
  foreach(keys(%$cr_pid)){
    my $cr_row_ix = $$cr_pid{$_};
    my $o_row_ix = $$o_pid{$_};
    my $cr_row = $$cr_links[$cr_row_ix];
    my $o_row = $$o_links[$o_row_ix];
    my %link_to = ();
    foreach(keys(%$cr_pid)){
      my $val;
      my $cr_col_ix = $$cr_pid{$_};
      my $o_col_ix = $$o_pid{$_};
      if ($$cr_row[$cr_col_ix] > 0 && $$o_row[$o_col_ix] > 0){
        $val = 'TP'; # Good link, true positive.
      } elsif ($$cr_row[$cr_col_ix] > 0 && $$o_row[$o_col_ix] == 0) {
        $val = ''; # False positive? To be evaluated manually.
      } elsif ($$cr_row[$cr_col_ix] == 0 && $$o_row[$o_col_ix] > 0){
        $val = 'FN'; # False negative :(.
      } else {
        $val = 'TN'; # True negative :).
      }
      $link_to{$_} = $val;

    }
    $links_by_name{$_} = \%link_to;
  }
  open EVAL, ">$evalfile" or die "Couldn't write evaluation sheet";
  select EVAL;
  print 'X' . sep;
  my @sorted_arts = sort keys(%links_by_name);
  foreach (@sorted_arts) {
    print $_ . sep;
  }
  print "\n";
  foreach (@sorted_arts){
    my $art = $_;
    print $art . sep;
    my $outlinks = $links_by_name{$art};
    foreach(@sorted_arts){
      print $$outlinks{$_} . sep;
    }
    print "\n";
  }
  close EVAL;
  select STDOUT;
  printprog("Evaluation sheet created.\n");
}
```

```perl
# Creates linked HTML documents.
# PARAMETERS:
# Hash with the following data:
# page_dir => collection directory,
# html_dir => output directory,
# cr_matrix => reference to link matrix,
# cr_pid_map = reference to PID map,
# o_matrix => reference to comparison link matrix (optional),
# p_link_pid => reference to comparison PID (obligatory if o_matrix),
# linkey => reference to map of linking keywords (optional).
# SIDE-EFFECTS:
# May select STDOUT;
sub create_html {
  my $params = shift @_;
  my $indir = $$params{page_dir()}; # Collection directory.
  my $outdir = $$params{html_dir()}; # Output directory.
  my $links = $$params{cr_matrix()}; # CrossRef links.
  my $pid = $$params{cr_pid_map()}; # CrossRef PID-data.
  my $control_links = $$params{o_matrix()}; # Comparison links.
  my $control_pid = $$params{o_pid_map()}; # Comparison PID.
  my $lk = $$params{linkey()}; # Linking keywords.
  my %lk_docs = ();
  printprog("Creating HTML output...\n");
  unless ($outdir =~ m/.+\/\z/){
    $outdir .= '/';
  }
  unless ($indir =~ m/.+\/\z/){
    $indir .= '/';
  }
  foreach (keys(%$pid)){
    my @outgoing = ();
    my @outgoing_orig = ();
    my %linking_kw = ();
    my %orig_match = ();
    my $c_page = $_; # Current page.
    my $c_doc_id = $$pid{$c_page}; # Current document ID.
    my $out_links = $$links[$c_doc_id]; # List of outgoing links.
    my $c_orig_id; # Comparison document ID.
    my $orig_out_links; # List of outgoing links in comparison matrix.
    if (defined($control_links)){
      my $c_orig_id = $$control_pid{$c_page};
      $orig_out_links = $$control_links[$c_orig_id];
    }
    foreach (keys(%$pid)){
      my $tar_page = $_; # Target page.
      my $tar_doc_id = $$pid{$tar_page}; # Target document ID.
      my $tar_orig_id; # Target doc id in control matrix.
      if (defined($control_links)){
        $tar_orig_id = $$control_pid{$tar_page};
      }
      if (defined($control_links) && $$orig_out_links[$tar_orig_id]>0){
        my $linked_file = $tar_page . html_suffix;
        push(@outgoing_orig, $linked_file);
      }
      if ($$out_links[$tar_doc_id] > 0){
        my $linked_file = $tar_page . html_suffix;
        my @dat = ($$out_links[$tar_doc_id], $linked_file);
        push(@outgoing,\@dat);
        if (defined($control_links)
            && $$orig_out_links[$tar_orig_id] > 0){
          $orig_match{$linked_file} = 1; # True positive.
        } elsif (defined($control_links)) {
          $orig_match{$linked_file} = 0; # False positive.
        }
        if ($lk){
          my $keyw_key = $c_doc_id . sep . $tar_doc_id;
          $linking_kw{$linked_file} = $$lk{$keyw_key};
        }
      }
    }
```

```perl
@outgoing = sort links_by_nr_shared_keywords @outgoing;
my $txtf = $indir . $c_page . file_suff;
my $txt = read_txt_file($txtf);
my $htmlf = $outdir . $c_page . html_suffix;
open HTMLF, ">$htmlf" or die "Couldn't write $htmlf";
select HTMLF;
print '<html>' . "\n";
print '<head>' . "\n";
print '<title>' . "\n";
print $c_page;
print '</title>' . "\n";
print '</head>' . "\n";
print '<body>' . "\n";
print '<p>' . "\n";
print $$txt;
print '</p>' . "\n";
print '<h5>' . "\n";
print 'CrossRef links:';
print '</h5>' . "\n";
print '<ul>' . "\n";
foreach (@outgoing){
  print '<li>' . "\n";
  my $target = $$_[1];
  if (defined($control_links)){
    print "$orig_match{$$_[1]}: ";
  }
  print '<a href="' . $target . '">'
        . $target . '</a><br/>' . "\n";
  my $kws = $linking_kw{$target};
  if (defined($kws)){
    foreach(@$kws){
      my $ng = $_;
      my $keyw;
      if (defined($$ng{ng_txt()})){
        $keyw = $$ng{ng_txt()};
      } else {
        $keyw = n_gram_text($$ng{ng_start()}, $$ng{ng_length()});
      }
      my $keyw_file = key_html_prefix;
      my $keyw_string = '';
      foreach (@$keyw){
        $keyw_string .= "$_ ";
        $keyw_file .= $_. "_";
      }
      $keyw_file .= html_suffix;
      if (exists($lk_docs{$keyw_file})){
        my $doclist = $lk_docs{$keyw_file};
        unless((grep {$target eq $_} @$doclist) > 0){
          push(@$doclist, $target);
        }
      } else {
        my @doclist = ($target);
        $lk_docs{$keyw_file} = \@doclist;
      }
      print " [ ";
        print '<a href="' . $keyw_file . '">'
              . $keyw_string . '</a>';
      print "]";
    }
  }
  print '</li><br/>' . "\n";
}
print '</ul>' . "\n";
if (defined($control_links)){
  print '<h5> Original links: </h5>' . "\n";
  print '<ul>' . "\n";
  foreach (@outgoing_orig){
    print '<li>' . "\n";
    if (defined($orig_match{$_})){
      print "$orig_match{$_}: ";
    } else {
```

```
          print '0: ';
        }
        print '<a href="' . $_ . '">' . $_ . '</a>' . "\n";
        print '</li>' . "\n";
      }
      print '</ul>' . "\n";
    }
    print '</body>' . "\n";
    print '</html>' . "\n";
    close HTMLF;
  }
  foreach (keys(%lk_docs)){
    printprog("Writing $_\n");
    my $keyfilename = $outdir . $_;
    open KEYWHTML, ">$keyfilename" or die "Couldn't write $_";
    select KEYWHTML;
    print '<html><head><title>Keyword: '
          . $_ . '</title></head><body>' . "\n";
    my $doclist = $lk_docs{$_};
    my @out_doclist = sort @$doclist;
    print '<ul>' . "\n";
    foreach (@out_doclist){
      print '<li><a href="' . $_ . '">' . $_ . '</a></li>' . "\n";
    }
    print '</ul>' . "\n";
    print '</body></html>';
    close KEYWHTML;
  }
  select STDOUT;
  printprog("HTML created.\n");
}

# Writes results from counting corpus data to a frequency lexicon file.
# The resulting file has a headline.
# Coulmns: | TF | DF | RIDF | MI | Text... |
# PARAMETERS:
# Filename for lexicon file.
# Reference to N-gram list.
sub create_freq_lex_file {
  my $lexfile = shift @_;
  my $ngrams = shift @_;
  open LEXFILE, ">$lexfile" or die "Couldn't open lexicon file.";
  select LEXFILE;
  print ng_tf . sep . ng_df . sep . ng_ridf . sep
        . ng_mi . sep . ng_txt . "\n";
  foreach (@$ngrams){
    unless(defined($$_{ng_start()})){
      die "No start";
    }
    unless($$_{ng_length()}){
      die "No length";
    }
    print $$_{ng_tf()} . sep . $$_{ng_df()} . sep
          . $$_{ng_ridf()} . sep;
    if ($$_{ng_mi()}){
      print $$_{ng_mi()} . sep;
    } else {
      print no_mi_val . sep;
    }
    my $ng_txt = n_gram_text($$_{ng_start()}, $$_{ng_length()});
    unless(@$ng_txt){
      die "No text returned for "
        . $$_{ng_start()} . " with " . $$_{ng_length()};
    }
    foreach (@$ng_txt){
      print $_ . sep;
    }
    print "\n";
  }
  close LEXFILE;
  select STDOUT;}
```

```
# Filters the found N-grams according to a given
# criterion. Creates a filtered output file.
# PARAMETERS:
# Filename for raw N-grams.
# Output filename.
# Threshold data: a hash ref:
# meas => relevant measure,
# thresh => threshold.
# SIDE-EFFECT:
# Selects STDOUT.
sub filter_results {
  my $infile = shift @_;
  my $outfile = shift @_;
  my $threshold = shift @_;
  open LEX, "<$infile" or die "Couldn't open $infile";
  open OUTF, ">$outfile" or die "Couldn't open $outfile";
  select OUTF;
  my @keys;
  my $measure;
  my $thresh = $$threshold{thresh()};
  while (<LEX>){
    my $line = $_;
    chomp($line);
    unless (@keys > 0){
      @keys = split(sep, $line);
      foreach (0..$#keys){
        print $keys[$_];
        print sep;
        if ($keys[$_] eq $$threshold{meas()}){
          $measure = $_;
        }
      }
      print "\n";
    } else {
      my @data = split(sep, $line);
      if ($data[$measure] >= $thresh){
        print "$line\n";
      }
    }
  }
  close LEX;
  close OUTF;
  select STDOUT;
}

# Removes frequent markup from all files
# in the input directory.
# PARAMETERS:
# Input directory (should contain only text files).
# File with a list of markup chunks.
# SIDE-EFFECT:
# Selects STDOUT.
sub remove_markup {
  my ($indir, $m_file) = @_;
  my @markup = ();
  open MARKUP, $m_file or die "Couldn't open markup file $m_file";
  while (<MARKUP>){
    chomp($_);
    my $line = $_;
    push(@markup, $line);
  }
  close MARKUP;
  opendir INDIR, $indir or die "Couldn't open directory $indir";
  foreach(readdir INDIR){
    unless ($_ eq '.' || $_ eq '..'){
      my $file = $indir . "/" . $_;
      print "Cleaning $file\n";
      my $txt = read_txt_file($file);
      foreach (@markup) {
        $$txt =~ s/$_//ig;
        write_txt_file($txt, $file);}}}}
```

```perl
# Writes a given keyword-list to file.
# PARAMETERS:
# Hash with the following keys (constant):
# keywords => Keyword-list,
# SIDE-EFFECT:
# Selects STDOUT.
sub write_keywords_to_file {
  my $params = shift @_;
  my $outf = $$params{keyw_file()};
  my $keywords = $$params{keywords()};
  open OUTF, ">$outf" or die "Couldn't create $outf";
  select OUTF;
  foreach (@$keywords){
    my @words = @$_;
    foreach (@words){
      print $_ . sep;
    }
    print "\n";
  }
  select STDOUT;
  close OUTF;
}

# Appends the input string to the input file.
# PARAMETERS:
# Output filename.
# String to be appended.
# SIDE-EFFECT:
# Selects STDOUT.
sub append_to_file {
  my ($file, $line) = @_;
  open OUTF, ">>$file" or die "Couldn't open $file";
  select OUTF;
  print $line;
  close OUTF;
  select STDOUT;
}

# Writes text to file.
# PARAMETERS:
# Ref to text to be written.
# Output file name.
# SIDE-EFFECT:
# Selects STDOUT.
sub write_txt_file {
  my ($txt, $outfile) = @_;
  open OUTF, ">$outfile" or die "Couldn't write to $outfile";
  select OUTF;
  print $$txt;
  select STDOUT;
  close OUTF;
}

# Writes hash data to file.
# PARAMETERS:
# Reference to hash to be stored.
# Output filename.
# SIDE-EFFECT:
# Selects STDOUT.
sub write_hash_info {
  my $params = shift @_;
  my $conff = shift @_;
  open HASH, ">$conff" or die "Couldn't open $conff";
  select HASH;
  foreach(keys(%$params)){
    my $key = $_;
    unless(ref($$params{$_})){
      print $key . sep . $$params{$_} . "\n";
    }
  }
  close HASH; select STDOUT;}
```

```perl
# Stores a user defined configuration.
# PARAMETERS:
# Ref to hash with user settings.
# Ref to hash with setting descriptions.
# SIDE EFFECT:
# May select STDOUT.
sub store_user_config {
  my $params = shift @_; # Parameter settings.
  my $param_desc = shift @_; # Parameter descriptions.
  my $settings; # Config file.
  print "Save as:\n";
  $settings = <STDIN>;
  eval {
    open CONFF, ">$settings";
    select CONFF;
    foreach (keys %$params){
      print comment() . ' ' . $$param_desc{$_};
      print $_ . sep() . $$params{$_} . "\n\n";
    }
    close CONFF;
    select STDOUT;
    print "Config file created: $settings\n";
  };
  if ($@){
    print "Couldn't write configuration file.\nTry again? {1,0}\n";
    my $tryag = <STDIN>;
    if ($tryag == 1){
      print "Save as:\n";
      $settings = <STDIN>;
      store_user_config($params, $param_desc, $settings);
    }
  }
}

# Reads text from file.
# PARAMETERS:
# Input filename.
# RETURNS:
# Reference to text.
# PRECONDITION:
# It's an existing textfile.
sub read_txt_file {
  my $inf = shift @_;
  open INF, "<$inf" or die "Couldn't open $inf";
  my $txt;
  while (<INF>){
    $txt .= $_;
  }
  close INF;
  return \$txt;
}

# Loads the relevant page identification data
# (namely a mapping from filename to document ID)
# into a hash.
# PARAMETERS:
# Filename of PID-file.
sub load_pid_data {
  my $pidfile = shift @_;
  my %pid = ();
  open PID, "<$pidfile" or die "Couldn't open $pidfile";
  while (<PID>){
    my $line = $_;
    chomp($line);
    my @dat = split(sep, $line);
    # Map filename to adjacency matrix doc-ID.
    $pid{$dat[pid_fn_index]} = $dat[pid_lm_index];
  }
  close PID;
  return \%pid;
}
```

```perl
# Loads an adjacency matrix modelling
# a link structure into an array.
# PARAMETERS:
# Filename of linkfile to read.
# RETURNS:
# Reference to link table.
sub load_link_matrix {
  my $linkfile = shift @_;
  my @adjmat = ();
  open LINKF, "<$linkfile" or die "Couldn't open $linkfile";
  while (<LINKF>){
    chomp($_);
    my @line = split(sep, $_);
    push(@adjmat, \@line);
  }
  close LINKF;
  return \@adjmat;
}

# Loads configuration data from file.
# The file should have the following format:
# key1;value1
# key2;value2
# ...
# It may contain comments introduced by #.
# PARAMETERS:
# Filename of config file.
# RETURNS:
# Reference to parameter hash.
sub load_config {
  my $conff = shift @_;
  my %params = ();
  my $com = comment;
  open CONFIG, "<$conff" or die "Couldn't open $conff";
  while (<CONFIG>){
    chomp($_);
    if ($_ eq ''){
      next;
    }
    my $data = $_;
    if ($data =~ m/^$com.*/){
      # It's a comment.
      next;
    } elsif ($data =~ m/([^$com\s]+)\s*#.*/){
      # Ends with a comment.
      $data = $1;
    }
    my @dat = split(sep, $data);
    unless(@dat == 2) {
      die "Couldn't interpret line $_ in $conff";
    }
    $params{@dat[0]} = $dat[1];
  }
  close CONFIG;
  return \%params;
}
```

```
# Loads the collection to be processed from
# files. Assigns each document a unique numeric
# ID that funktions as the index of the document
# in the adjacency link matrix.
# PARAMETERS:
# A hash with the following content (use constant as keys):
# page_dir => directory with collection (text-only).
# limit => maximal N-gram length.
# lex_file => output file for raw lexicon.
# map_fn => recall filename?
# RETURNS:
# Reference to collection array.
sub load_collection {
  my $params = shift @_;
  opendir DATDIR, $$params{page_dir()}
    or die "Couldn't open page directory";
  my @collection = ();
  my $count = 0;
  foreach (readdir DATDIR){
    my $f = $_;
    chomp($f);
    if ($f eq '.' || $f eq '..'){
      next;
    }
    printprog("Reading $f\n");
    my $txt = read_txt_file($$params{page_dir()} . "/" . $f);
    push(@collection, $txt);
    push(@collection, $count);
    $f =~ s/\..*//g; # Remove filename extension if any.
    $fn_to_doc{$f} = $count;
    $count++;
  }
  return \@collection;
}

# Loads raw N-gram data from file.
# PARAMETERS:
# Hash with the following keys:
# lex_file => Input lexicon file.
sub load_ngram_data {
  my $params = shift @_;
  my @ng_data = ();
  open LEX, $$params{lex_file()}
    or die "Couldn't open frequency lexicon";
  my $headl = readline(LEX);
  chomp($headl);
  my @headl = split(sep, $headl);
  while(<LEX>){
    chomp($_);
    my @words = ();
    my @line = split(sep, $_);
    my %ng = ();
    foreach(0..$#headl-1){
      $ng{$headl[$_]} = $line[$_];
    }
    foreach($#headl..$#line){
      push(@words, $line[$_]);
    }
    $ng{$headl[$#headl]} = \@words;
    push(@ng_data, \%ng);
  }
  close LEX;
  return \@ng_data;
}
```

```perl
# Loads the content of the specified column
# from the input file (where columns must be)
# separated by ';' into an array.
# PARAMETERS:
# Filename of list to load.
# Index of column (0 for first).
# RETURNS:
# Reference to loaded list.
sub load_list {
  my $filename = shift @_;
  my $col = shift @_;
  my @list;
  open LIST, "<$filename" or die "Couldn't open $filename";
  while (<LIST>){
    chomp($_);
    my @dat = split(sep, $_);
    push(@list, $dat[$col]);
  }
  close LIST;
  return \@list;
}

# Prompts for user defined configuration settings.
# Optionally saves settings to file.
# The procedure is rather tyring and typos, etc. might
# cause many errors. So, if possible, edit the config
# file directly.
sub prompt_for_settings {
  my %params = (); # Parameter hash.
  my %param_desc = (
    alpha() => "Weighting factor for f-measure:\n",
    df_min() => "Parameter: Minimum document frequency:\n",
    html_dir() => "Output: HTML directory (should EXIST):\n",
    keyw_file => "Output: Key n-gram file (only key n-grams)\n",
    length_min() => "Parameter: Minimum key n-gram length:\n",
    lex_file() => "Output: N-gram lexicon (complete, with details):\n",
    limit() => "Parameter: Maximum n-gram length:\n",
    link_cr => "Output: CrossRef link matrix:\n",
    link_o => "Input: Gold-standard link matrix:\n",
    page_dir() => "Input: Collection directory:\n",
    param_result_file => "Output: results of parameter testing:\n",
    pid_o() => "Input: Gold-standard index->page map:\n",
    print_prog() => "Parameter: Print progress info? {1,0}\n",
    ridf_min() => "Parameter: RIDF threshold:\n",
    score_file() => "Output: Eval of linkS against gold-standard:\n",
    stop_list() => "Input: Stopword list:\n",
    store_lk() => "Parameter: Include linking n-grams in HTML? {1,0}\n",

  );
  print "\nPlease specify parameters.\n\n";
  foreach (keys %param_desc){
    print $param_desc{$_};
    $params{$_} = <STDIN>;
  }
  print "\nStore settings? {1,0}\n";
  my $storit = <STDIN>;
  if ($storit == 1){
    store_user_config(\%params, \%param_desc);
  }
  return \%params;
}
```

```
__END__

=head1 NAME

CrossRef - Code to perform the experiment described in the essay
'Linking Documents by Distinctive Phrases'.

=head1 SYNOPSIS

Don't use any of the alternatives below in sequence, since this may
corrupt your results!

  use CrossRef;

  # This will run a demo. Demo prompts for user input!
  my $config =
       '/your/absolute/path/to/a/configuration/file/configuration.txt';
  CrossRef::demo($config);

  #----------------------------------------------------------------------

  use CrossRef;

  # This will link your collection, create HTML output and write
  # results of evaluation against a gold-standard to file.
  my $config =
       '/your/absolute/path/to/a/configuration/file/configuration.txt';
  CrossRef::link_and_match_directly($config);

  #----------------------------------------------------------------------

  use CrossRef;

  # This will link your collection and create HTML output.
  my $config =
       '/your/absolute/path/to/a/configuration/file/configuration.txt';
  CrossRef::link_directly($config);


  #----------------------------------------------------------------------

  use CrossRef;

  # This will test linking results against a gold-standard for several
  # values of RIDF.
  my $config =
       '/your/absolute/path/to/a/configuration/file/configuration.txt';
  CrossRef::test_parameters($config);



The most convenient way to use the C<CrossRef.pm> module is by means of
a configuration file in the following format:


   # TEMPLATE FOR A CrossRef CONFIGURATION FILE
   # ===========================================
   #
   # FORMAT:
   # --------
   #
   # Each setting is represented as a line beginning with a hash key
   # followed by a semicolon followed by the actual setting.
   # Comments are marked with a '#', the rest of the line is ignored.
   # You may disable optional settings by commenting out or
   # removing the respecting line.
   # Distinct functions may require distinct parameters.
   #
```

```
# INPUT FILES:
# -------------

# Directory containing the collection to be linked
# (obligatory, no slash at the end).
page_dir;/your/path/to/collections/collection/texts

# Gold-standard link adjacency matrix. Necessary for matching.
links_orig;/your/path/to/collections/collection/data/links_orig.txt

# Gold-standard filename->doc-ID mapping.
# Maps the array index used internally to the filename of the text.
# Necessary for matching.
pid_orig;/your/path/to/collections/collection/data/pid_orig.txt

# Stopword file. Optional.
stop_list;/your/path/to/lists/stops.csv


# OUTPUT FILES:
#---------------

# This file will contain a list of mere key N-grams.
# Optional.
keyword_file;/your/path/to/output/keyword_list

# Mapping from file name to the doc-ID used CrossRef internally.
# Optional.
pid_cr;/your/path/to/output/pid_cr

# CrossRef-generated link matrix.
# Optional.
linkfile;/your/path/to/output/cr_link_matrix

# Results of comparing CrossRef link matrix to gold-standard.
# Optional.
score_file;/your/path/to/output/scores

# Frequency lexicon with TF, DF, RIDF and MI.
# Optional.
lexfile;/your/path/to/output/lexicon

# Output file for results of parameter optimization.
# Nexessary for parameter testing.
param_res;/your/path/to/output/parameter_test

# Output directory for HTML-files. Should EXIST.
# Necessary for HTML creation.
html;/your/path/to/output/html

# Output directory for evaluation sheet.
# Optional
eval_sheet;/your/path/to/output/eval_sheet


# PARAMETERS:
# ------------

# RIDF threshold. Optional but recommended!
ridf_min;1.5 # Default.

# Minimum DF. Optional.
df_min;2 # Default.

# Minimum key n-gram length. Optional.
length_min;2 # Default.

# Maximum key N-gram length.
# Optional but highly recommended!
lim;5 # Default.
```

```
  # Store linking keywords? If true, the linking keywords will be
  # included in the HTML-output. Obsolete for parameter testing.
  store_lk;1 # Default.

  # Weighting factor for f-measure (0.5 for equal weigthing of precision
  # and recall).
  alpha;0.5 # Default.

  # Print progress info?
  print_prog;1

You can also run the demo from the command-line:
C<perl CrossRef.pm
/your/absolute/path/to/a/configuration/file/configuration.txt>

=head1 DESCRIPTION

This module has been created to perform an experiment in the field of
automatic document linking.

For this purpose, the module contains subroutines to:
tokenize an input collection,
calculate I<collection frequency> (called I<term frequency> in
Yamamoto/Church (1998)) and I<document frequency>
for n-grams from an input text collection,
calculate I<residual inverse document frequency> (I<RIDF>) and I<mutual
information> (I<MI>) for the n-grams,
link the input collection,
create C<HTML> output and
match the resulting link-structure against a gold-standard.

In particular, the linking strategy employed here is to link two
documents if they contain at least one
common keyword, where keywords are determined by an I<RIDF>-value equal
to or above a given threshold.

This module presupposes that you have a text collection available in
file format and that
you have a gold-standard link structure in the form of an adjacency
matrix.

=head2 EXPORT

Nothing to
export. This module is too specialized (and too premature) to be
included in a distribution.

=head1 SEE ALSO

For a nice tool to retrieve Wikipedia articles (as a text collection)
see
L<http://search.cpan.org/~bricas/WWW-Wikipedia-
1.92/lib/WWW/Wikipedia.pm>.


=head2 LITERATURE

JOHANNSEN, Heike (2007). 'Linking Documents by Distinctive Phrases'.
Soon to appear at L<http://www.sfs.uni-tuebingen.de/iscl/English/Thesen-
en.shtml>.

MANNING, Christopher D. / SCHE<Uuml>TZE, Hinrich (1999). I<Foundations
of Statistical Natural Language Processing>.
Cambridge, Massachusetts / London, England: MIT Press.

YAMAMOTO, Mikio / CHURCH, Kenneth Ward (1998).
'Using Suffix Arrays to Compute Term Frequency and Document Frequency
for All Substrings in a Corpus'.
In: I<Proceedings of ACL Workshop on Very Large Corpora>. P. 28-37.
Montreal.
L<http://acl.ldc.upenn.edu/J/J01/J01-1001.pdf> (03/2007).
```

```
=head1 AUTHOR

Heike Johannsen, H_Johannsen@web.de

=head1 COPYRIGHT AND LICENSE

Copyright (C) 2007 by H. Johannsen

This module is free software; you can redistribute it and/or modify
it under the same terms as Perl itself, either Perl version 5.8.8 or,
at your option, any later version of Perl 5 you may have available.

=cut
```