Identifying and Interpreting Context on the Web: An Application-driven Approach

Daniel Kelleher

A thesis submitted to the University of Dublin, Trinity College

in fulfillment of the requirements for the degree of

Doctor of Philosophy

October 2007

Identifying and Interpreting Context on the Web: An Application-driven Approach

Approved by
Dissertation Committee:

Declaration

I, the undersigned, declare that this work has not previously been submitted to this or any other University, and that unless otherwise stated, it is entirely my own work. This thesis may be borrowed or copied upon request with the permission of the Librarian, University of Dublin, Trinity College.

[illegible]

Daniel Kelleher

Dated: September 22, 2008

"Taken out of context I must seem so strange"

Ani DiFranco

Abstract

This work describes the use of contextual information in Web document processing. Contextual information is defined as the contents of a 'context set' of a document of interest on the Web. The context set of a document is determined by the hyperlink structure of the Web around the document of interest. This thesis suggests that, as in other text media, contextual information on the Web is a vital component of the information content of a document, and should be taken into account when interpreting or processing that information. Most existing hypertext document processing applications either ignore hyperlinks, or use them in a restrictive manner, to identify a particular form of contextual information. For example, many information retrieval applications use hyperlinks as indicators of document *prestige* or authority, conferred by the referring document to the referred document. Others use them to locate (or help to locate) *similar* content in order to augment a document index or provide additional information about a document's relevance. A related method is the clustering of Web documents, using the Web hyperlink structure to identify clusters of related documents, either to generate an aggregate that can be used in document indexing, or to simplify a visual representation of the graph structure in order to aid browsing. These approaches typically apply the information provided by hyperlinks to a particular application, such as information retrieval, or Web browsing. In contrast, this thesis proposes a flexible method for the inclusion of hypertext contextual information in a number of document processing applications, based on an adaptation of a term weighting measure based on the frequency of a term in a set of documents. The resultant non-linear measure can be incorporated into Web applications using a probabilistic model trained on pre-annotated data suitable for the domain of the application. In this work, the measure is implemented and evaluated on a number of Web document content processing applications. Specifically, three applications are presented: an adaptation of an existing automatic keyphrase extraction application, a Web document retrieval ranking algorithm, and a document collection homogeneity measure with a related homogeneous corpus generation application.

Each of these applications is evaluated against the state of the art in the application domains. Contextual information on the Web is found to provide valuable additional information in each of the applications, demonstrating that the method described in this thesis is both a flexible and effective approach to the question of context on the Web.

Acknowledgements

First of all, I would like to thank my supervisor, Dr. Saturnino Luz, for guidance, encouragement and suggestions throughout my research. I took great comfort in the knowledge that I could rely on his prompt and focused feedback whenever needed. I would also like to express my sincere gratitude to Dr. Carl Vogel. Throughout my time at Trinity, he has always been forthcoming with support, advice and wisdom. Thanks also go to Michael Davy in the Artificial Intelligence group for his part in our continuing collaboration, and for his help in clarifying my thought processes.

This work was funded with the generous support of Trinity College Dublin, both with the Ussher Fellowship grant and the invaluable Scholarship award.

Finally, this thesis would not exist without Kirsten Bratke, who has been my principal source of support and encouragement from the beginning. In particular, her watchful and astute editing have helped shape the thesis, and her presence as a sounding-board have helped shape the ideas therein.

Daniel Kelleher
University of Dublin, Trinity College
October 2007

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Context

A passage of text cannot be fully understood without inspecting its context: text that surrounds it and plays a part in determining its meaning. The location of a document's context depends on the text medium. A chapter in a book exists in the context of the chapters located before (and after) it, and a reader of the chapter would not be expected to fully grasp the meaning of the content without first having read the preceding chapters. Likewise the content of the chapter will have a bearing on the understanding of subsequent chapters. The same is true for individual pages, paragraphs, sentences or words within the book. The degree to which context affects the understanding of text also varies with respect to the medium and content matter. Some texts, such as a chapter in a collection of short stories, can be read independently from other chapters, and context on the chapter level is therefore unimportant. Other texts, such as a chapter in a novel, are heavily influenced by its context.

The definition of context in the case of a book is linear, and unidirectional. In certain cases, especially in academic texts, the linearity of this definition is broken by the inclusion of confer (cf.) references to other locations in a text. There, the context of the document includes the information at the other end of the confer reference. Academic papers also contain references to previous works by other authors, which in turn will reference earlier works, resulting in a directed acyclic graph structure of context for the document. It is expected that directly referenced texts are more relevant than indirectly referenced ones, and as a general rule, the further the reader travels (or 'browses') along this graph structure from the original document, the less relevant the information found is likely to become. The context of the document is therefore not a clearly-defined set of information, but a region, with blurred edges, surrounding the document.

1.2 The Web as a document collection

Web documents are similar to academic journals in that they can contain references (hyperlinks) to other documents, and there is an assumption that referenced documents are related, i.e. they have content which is, to some degree, semantically connected to the content of the referring document. However, in the Web, the chronological element is removed – documents can reference pages that will change their content in the future, without the references being considered invalid. Cycles can therefore be introduced into the connections between pages, resulting in a cyclic graph structure. This graph structure is known as a Hypertext, the largest and most well-known of which is the Web, but others exist.

In a large number of document processing techniques, such as document retrieval, documents are treated as independent resources; the content of one document will have no bearing on the interpretation of the content of another. Printed documents are supplied as a set of pages, and the act of browsing within the document – turning a page – has a low cost to the browser in comparison to locating a different document. However, Web documents exist in a wider, easily browsable, structure in conjunction with other documents. While the contents of some documents are more independent than others, some documents are intended to be read along with, or after other documents in the structure. On a typical Web site, documents often perform specific functions in relation to the other documents. A Web site may have a Frequently Asked Questions page, an ordering page, a contact page, etc. Each of these documents themselves may not contain sufficient information to fully understand the subject on which they are based, but anyone browsing to one of these pages would have no trouble understanding their purpose, having been exposed to their context. Web-based document processing algorithms therefore should interpret the context of a document as well as its content in order to maximize the potential of the hypertextual nature of the Web.

In addition to the hyperlink structure of the Web, a number of other differences between Web documents and traditional documents exist. One important difference is that the notion of a collection is often not well-defined on the Web. This is due to a number of factors:

- The Web is practically unlimited in size, as new documents are being created by the minute, and it is impossible to index or collect the entire set. Domain names are useful delimiters of a document collection, but documents within the domain are free to link to documents outside the domain, and vice-versa, meaning that collections based on domain name only are incomplete from the point of view of identifying context. This distributed nature of the Web is potentially exacerbated by a trend towards user-driven content described below.

- The Web is dynamic - documents are regularly added, removed and changed. A collection may vary over time as a result. Whereas documents can be added and removed from other document collections, this is a more common occurrence on the Web. Web documents also have large variation in authority and authorship, and are written with a variety of intentions and registers. Unlike conventional document collections, which often pass editing and reviewing stages before being published, and have a comparably small author set, publication of Web content is far less restrictive.

A consequence of these factors is that the Web is less *collection-orientated* than traditional documents. Web document collections are typically defined by the Web site or server on which they are found, but this is a convention, rather than a requisite, and this convention is being broken down, as described in the next section.

1.3 The Changing Web

The lack of constraints on the Web has led to its development as a huge, organic, democratic information resource, in which structures emerge due to convention, co-operation and technological capability. A result of this organic nature is the capability of the Web to change and develop as trends change and as a response to advances in technology. For example, during the early years of the Web, manually-generated directories, containing hierarchical categories of Web documents, such as the Yahoo directory, were popular information resources. As storage cost and bandwidth became cheaper and advances were made in Web-based Information Retrieval (IR) (see section 2.1), search engines were able to present more comprehensive information to a user in response to a query, and were preferred due to their coverage and cost. Another example is the change in HTML use over time. As malevolent or commercial Web sites aimed to obtain more Web traffic by abusing meta-information tags by including irrelevant keywords, HTML meta-tags became less reliable sources of document information. In addition, as the skill levels and intentions of HTML authors became more varied, the use of structural HTML tags (such as `table`, `address` and headings: `h1`, `h2` etc.) began to deviate from the W3C specifications, with authors using them for their visual effects in the Web browsers of the time, rather than to designate syntactical structure in a document.

Another, more social, change to the Web is presently occurring as a result of the prevalence of low-cost broadband services and pervasiveness of the Web across all age and social groups. This change is the increase of user-driven content and interactive Web sites, sometimes described as 'Web 2.0'. Many Web sites, such as social networking services (e.g. MySpace, Facebook), Web log software (Blogger), image and video hosting sites (YouTube, Flickr) and open-content information resources (Wikipedia and IMDb) are now among the most popular sites on the Web. They exist purely as a repository of user-contributed content, and include robust and straightforward authoring and uploading tools that allow users to add content without requiring a large amount of technological knowledge. An analysis of Google results for twenty popular queries in March 2007 shows that Web sites based primarily on user-driven content make up approximately 27% of the top ten search results.[1]

This current Web trend of user-contributed content has effects on the structural nature of the Web. Whereas previously, Web users were generally disjoint sets of information providers and information consumers, these distinctions are now blurring. As a result, the physical structuring of information on the Web is changing. Information consumers do not have access to some physical mechanisms previously used by information providers, such as directory structures of hosting servers and domain names. Sites with user-driven content instead tend to have flat URL structures, and hierarchical structure emerges from logical (as opposed to physical) frameworks such as manually-defined categories (in Wikipedia) and tags (in Flickr). Inter-site linking is also common, as users on MySpace and Web logs may provide links to their images on Flickr or videos hosted on YouTube, for example.

The changes in Web use described here have important effects on the identification of context on the Web. As the Web moves away from traditionally hierarchical structures, fewer structural indicators of context, such as meta-data and URL directory structure, are available. A robust mechanism for the incorporation of context should be tolerant of changes in Web use, and should not be linked to unreliable mechanisms such as URL directory structures and meta-data.

1.4 Context on the Web - A proposal

At this point, we have established the following:

- Document context is important for the complete understanding of a text.
- The definition of context depends on the document medium.
- Traditional sources of contextual information – meta-data, collection information etc. – is not reliable on the Web.
- A conventional indicator of context on the Web – URL directory structure – is becoming potentially less reliable as the Web moves towards a paradigm of user-contributed content.
- Due to the non-linear hypertextual nature of the Web, the simplicity of use of the hyperlink browsing system, and low ergonomic cost of browsing to new documents, contextual information is potentially more vital to the understanding of Web documents than traditional documents.

We therefore define the term *context* in respect to Web documents to indicate the information about a document obtained by inspecting documents that are connected by a path of hyperlinks to it.

[1]The queries are taken from `www.google.com/press/zeitgeist.html` between 18/3/2007 and 1/4/2007. Sites are based on user-driven content if the principal source of information on the site is obtained from multiple contributors and if these contributors can provide information without significant restrictions.

This local definition of context avoids difficulties of collection identification, and is general enough to be interpreted in a variety of manners. This thesis focuses on the use of contextual information to give robust information at the level of individual document terms (words and phrases), and we demonstrate the use of this information in a number of applications.

1.5 Comparison with Related Work

Arguably, the hyperlink-analysis method that has had the greatest impact on the way we use the Web, since its inception, is the PageRank measure used in the Google search engine (Brin and Page, 1998; Page et al., 1998). PageRank uses in-links (hyperlinks pointing to a document of interest from other documents) as an indicator of document authority or prestige. These measures can then be used to influence a document's rank (derived from its RSV - Retrieval Status Value) in a result set from a search engine. Similar methods are Kleinberg's HITS algorithm (Kleinberg, 1999) and methods using in-degree (Pinkerton, 1994). These methods typically examine only the hyperlink topology of the Web, i.e. they do not take the contents of the pages into account in their calculations (these methods are described in more detail in chapter 2. In this respect, the definition of Web context used in this work differs from these approaches.

However, a number of methods do use the contents of linked documents in the calculation of RSVs. Relevance Propagation methods (Savoy, 1994, 1996; Frei and Stieger, 1992, 1995; Marchiori, 1997; Shakery and Zhai, 2003; Sugiyama et al., 2003; Shakery and Zhai, 2006) examine this information in order to give an indication about the relevance of a document. A context set similar to that used in this work is now established – the contents of a document's context set are used in these methods to alter its ranking score (RSV). These methods are usually applied specifically to IR, and are suitable only in the case of linked documents that are 'about' the same subject. Some methods (Frei and Stieger, 1992, 1995; Shakery and Zhai, 2003; Sugiyama et al., 2003) assign weights to members of the context set according to their textual similarity with the source document, or to the original query.

The above relevance propagation methods are carried out post-search (i.e. they alter existing RSVs of documents obtained from an external source, such as a text-based search engine). A more flexible method adapts the representation of a document in an index to include this information (Sugiyama et al., 2003). Alternatively, a separate index is generated that contains 'contextual information' (Aguiar, 2003). These methods still favour document content similarity over a more general definition of 'context', however. The value of contextual information provided by a linked document is relative to its similarity to existing information. These methods therefore focus on the ability of contextual information to reinforce existing information, rather than to provide new information. This focus is a consequence of early work on pre-Web document networks (Salton and Zhang, 1986) that found that indiscriminate inclusion of contextual information in a document index often results in greater noise in search results. The intention of this work is to treat Web context more comprehensively, so that not only similar document content but also contrasting content can be used.

Another use of hyperlinks is in document clustering. Using hyperlinks or 'hyperlink similarity' measures to identify clusters, sometimes along with document content similarity measures, documents are averaged or aggregated in order to improve searching or browsing. The method used by Aguiar is an example of the use of clustering in an IR environment. Other works based on clustering methods are Zamir and Etzioni (1999); Weiss et al. (1996); Eiron and McCurley (2003); Gloor (1991); Liu et al. (2002); Botafogo and Shneiderman (1991); Botafogo (1993); Zhu et al. (2004); Pitkow and Pirolli (1997). The work described in this thesis also has potential uses in the field of browsing.

Clustering techniques are often intolerant of changes to the structure and contents of documents within clusters. Changing the hyperlink structure or content of a Web document may necessitate a recalculation of all document clusters. This renders such methods unsuitable for large-scale implementation on the Web, due to its naturally volatile and transient nature. In addition, the use of content similarity measures when generating clusters restricts the use of contextual information in a similar fashion to the Relevance Propagation methods.

A more comprehensive use of context in Hyperlink-based IR applications can be found in structured Web query languages (Konopnicki and Shmueli, 1995; Lakshmanan et al., 1996; Spertus, 1998; Lacroix et al., 1997; Mendelzon et al., 1996, 1997). However, these methods increase the complexity of the query language by requiring the user to specify the nature of the relationship between an information source and its context. In contrast, none of the cost of incorporating or interpreting contextual information is incurred by the user in an IR environment based on this work. Other approaches rely on meta-data to provide indications about semantic relationships between documents (Frei and Stieger, 1992, 1995). However, such semantic indicators are rarely available on the Web. Similar approaches involve inferring semantic relationships from the physical directory structure of a Web site or the URL of a particular document (Dyreson, 1997, 2002). This is an unreliable information source on the modern Web, as mentioned previously.

1.6 Motivation

The principal motivation behind this work is to develop a method of addressing Web context that is not restricted to a particular application domain. For this reason, the method proposed in chapter 3 is implemented and evaluated on three separate applications:

- Automatic keyphrase extraction
- Document retrieval and ranking

- Corpus homogeneity

Each of these applications are described in detail, and together they demonstrate the flexibility of the proposed method. Further applications, and extensions to those described here are proposed in the conclusions chapter.

Secondary motivations for this work were based on those deficiencies described briefly in the above section, and are explored in more detail in the following chapter. In particular, a methodology based on Web context should not be restricted simply to a content similarity measure, and should not be reliant on meta-data or URL directory structure. The methodology presented in this thesis is designed to avoid these problems, and to be flexible enough to be used alongside existing hyperlink-based methodologies (such as PageRank and clustering methods), in order to provide complementary information about document context.

1.7 Thesis Structure

In chapter 2, the works described in section 1.5 are described in more detail. Chapter 3 proposes the method used in this thesis: an adaptation of the TF$\times$IDF (Term Frequency x Inverse Document Frequency) term weighting measure for use on Web collections, named TF$\times$INF (Term Frequency by Inverse Neighbour Frequency). This measure captures the relationship between a document and its 'context set' with respect to a word or phrase. The context set is defined as the set of documents connected to the document by hyperlinks. The resultant measure is a non-linear term-weighting measure that represents a term as either highly specific to a document with respect to its neighbours, or commonly found in a document's neighbours. Probability distributions based on this measure can be used in a variety of applications, and in the following chapters we present the implementation and evaluation of three such applications: an automatic keyphrase extractor, an Information Retrieval system, and a measure of Web corpus homogeneity.

The automatic keyphrase extractor (chapter 4) is an adaptation for the Web of an existing application that uses machine-learning techniques to identify keyphrases in documents. We suggest that document context plays an important role in the identification of suitable keyphrases in Web documents, and demonstrate this by extending the keyphrase extractor to incorporate TF$\times$INF. We apply the application to four courpora obtained from the Web, which contain manually annotated keyphrases, and we find that doing so significantly improves the number of keyphrases suggested by the application that match the manually annotated keyphrases, outperforming the state of the art of domain-independent automatic keyphrase extractors on Web data.

In order to demonstrate the wider applicability of our method, we develop a document ranking algorithm using the probability-based model generated by the keyphrase extractor. We compare this algorithm to TF$\times$IDF-based ranking algorithms and other hyperlink-based ranking algorithms described in chapter 5. Promising results were obtained that indicate that TF$\times$INF can be used in conjunction with existing IR technologies in order to more comprehensively interpret Web contextual information.

The third application is a measurement of Web corpus homogeneity (chapter 6). We describe content homogeneity and discuss its importance in NLP applications, and propose a method that calculates the difference between TF$\times$INF and TF$\times$IDF distributions to determine the homogeneity of the corpus. Unlike the other applications, this method does not include machine-learning, and does not require a probabilistic model to use TF$\times$INF. The method is evaluated on a corpus based on the Wikipedia Web site, and the measure is found to correlate strongly with a homogeneity measure based on the subject categories associated with documents in the corpus.

A final, related, application generates homogeneous Web corpora using this homogeneity measure by selecting documents that increase the heterogeneity of the corpus the least when added (chapter 7).

Finally, in chapter 8 we provide an overview of the results obtained from all applications, and present a discussion of the implications of these results in future Web applications.

Chapter 2

Background

This section gives an overview of the main directions of work involving the use of hyperlinks in Web document processing. The majority of these works aim to improve document retrieval, either by extending the query language or by improving the indexing process. These processes typically use non-hypertextual content-based document retrieval systems as their starting point, and propose methods for the inclusion of hyperlink information. Some works aim to improve users' browsing experiences and solve the 'lost in hyperspace' problem. Other works aim to provide tools for describing aspects of the Web, for use in the applications mentioned above, among others.

Most of the methodologies used in these works fall broadly into four categories:

- Link topology analysis
- Relevance propagation
- Clustering methods
- Structured Web query languages

These four categories form the basis for the sections of this chapter. Works in different sections will have commonalities with each other, however. Aside from these categories, each methodology can be described according to the following factors:

a Nature of the Hypertext: Does the methodology apply to Web data, pre-Web hypertext data, hypertexts induced from academic document citations, or hypertexts induced from text-based nearest neighbour (NN) techniques?

b Link topology: Are document in-links (links from other documents pointing to a document of interest), or a representation of the Web graph or a large section of it required?

c Content similarity: Are text-based (content) similarity measures used?

d Collection based: Does the methodology use collection-based term weighting measures, such as TF×IDF?

e Site Hierarchy: Is information obtained from documents' URL directory structure? Is a hierarchical Web site structure assumed?

f Meta-data: Does the methodology use other meta-data attached to the links or the documents?

Each of these factors has an effect on the scope, generality and applicability of the method to Web documents on the whole. These factors are revisited at the end of the review, and in the next chapter.

Some works explicitly define document context in terms of the link structure, but most either treat context implicitly, or treat links as providing different information to context as defined in chapter 1. At the end of this review, we identify those works that are most relevant to our definition, and their restrictions.

2.1 Link Topology Analysis - A problem with authority

One of the central challenges faced by a Web document ranking system is the problem of authority. Due to the democratic nature of the Web, authors are free to publish content on any subject and make it as available as all other documents on the Web. This is in contrast to other publishing media, in which documents are often required to pass an editing or reviewing phase. Additionally, authors may have different motives or intentions (such as commercial, informative, trivial) when writing a document about a subject. As a result, the Web contains documents that are of varying authority on a subject. For example, document *a* may be the microsoft.com homepage, and document *b* may be a personal site entitled "Microsoft - My experiences with its software". A user carrying out a search with the simple query "Microsoft" would presumably expect document *a* to be ranked above document *b*, but if the word Microsoft occurs equally frequently in both documents, a term-frequency-based weighting scheme such as TF×IDF is unable to differentiate between them. In fact, if document *b* contained more occurrences of the word Microsoft, it would be ranked higher by TF×IDF. 'Web spam' documents took advantage of this by flooding documents with large numbers of repeated irrelevant words, sometimes hidden using special functionality present in the HTML language.

Due to the scale of the Web, a search engine may return a large amount of documents that are adjudged relevant using term-frequency-based weighting schemes. The returned list may be too large for a user to comfortably read in its entirety. This is known as the Abundance Problem. Within this result list are documents of varying degrees of authority. Authoritative documents are more suitable to the user's information need, but may be ranked equally, or lower than less authoritative documents using TF$\times$IDF. A ranking system should therefore take into account the authority of a page when sorting the result list.

In 1998, two separate algorithms, HITS (Hypertext Induced Topic Search) and PageRank, were proposed to solve the authority problem. These approaches fall under the category of 'link topology analysis' ranking (Henzinger, 2000, 2001). Both approaches are similar and are based on the intuitive notion that authority is conferred onto entities within a set by 'votes' from other entities in the set. Early network-aware search engines, such as WebCrawler (Pinkerton, 1994) used document in-degree to estimate the authority of a document, reasoning that if a large number of Web documents contain links to a specific document d, then d is likely to be more important than another document e with low in-degree, all other factors being equal. In this way, hyperlinks are treated as votes for a certain page. This idea also exists in the field of bibliometrics, in which citations of academic papers are regularly used to quantify the output of academics (Hirsch, 2005).

The innovation of the HITS and PageRank weighting schemes is that they are iterative algorithms that calculate a document's authority based on the authority of other documents in the network. The relationship between the authority of a document and the authority of other documents in the network is one of the main differences between the two approaches. These schemes are described in detail later in this section.

2.1.1 The Web as a Scale-free Network

Numerous studies have described the scale-free nature of the Web network (Donato et al., 2004; Broder et al., 2000; Barabasi et al., 2000; Barabasi and Albert, 1999). In a scale-free network, the in-degrees of nodes follow a power law. The power law is colloquialised in the Pareto, or 'long-tail', principle that states that a small number of high-frequency events are balanced by a large number of low frequency events. In the context of the Web, high-frequency events are documents with high in-degree, and low-frequency events are those with low in-degree. The power law states that a small number of documents will have an in-degree greatly above the average, while the vast majority of documents will have an in-degree below the average. The power law is defined as:

$$F(k) = \beta k^{-\alpha} \tag{2.1}$$

where $F(k)$ is the number of nodes having an in-degree of k
β is a linear constant (or a function of k)
and α is the coefficient of proportionality on the log-log scale i.e.

$$\log(F(k)) = \log(\beta) - \alpha \cdot \log(k) \tag{2.2}$$

(Note that the power-law is linear on the log-log scale)

Networks that follow the power law are called scale-free networks due to the scale-invariance of

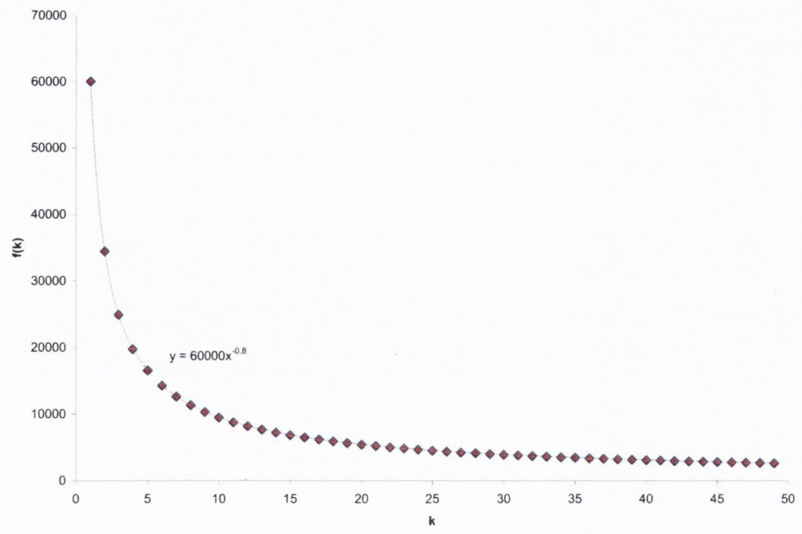


Figure 2.1: The long tail. k = rank of event, $f(k)$ = frequency of event under function $f(k) = 60000k^{-0.8}$

the power law. i.e. for some scaling factor c:

$$\begin{aligned} F(ck) &= \beta(ck)^{-\alpha} \\ &= \beta c^{-\alpha} k^{-\alpha} \\ &= c^{-\alpha} F(k) \end{aligned}$$

On the log scale:

$$\begin{aligned} \log(F(ck)) &= log(\beta) - \alpha \log(ck) \\ &= log(\beta) - \alpha \log(c) - \alpha \log(k) \\ &= log(F(k)) - \alpha \log(c) \end{aligned}$$

Therefore, $\log(F(ck))$ is concurrent with $\log(F(k))$. A scaled version of a power law frequency distribution will contain the original power law distribution, thus a network based on a power law is scale-free.

Origin of scale-free behaviour on the Web

A number of organic networks, or networks that are based on observable real-world phenomena, such as social networks or the neural network structure of the brain, follow the scale-free pattern. On the Web, this is due to the rule of preferential attachment in the generative model of the Web. When a new node is added to the network, the probability that it will contain a link to an existing document is based on the visibility of the document, i.e. the probability that a random surfer will happen upon it. This probability is related to the in-degree of documents, as documents with more in-links are more likely to be reached than documents with fewer in-links. Therefore, a document

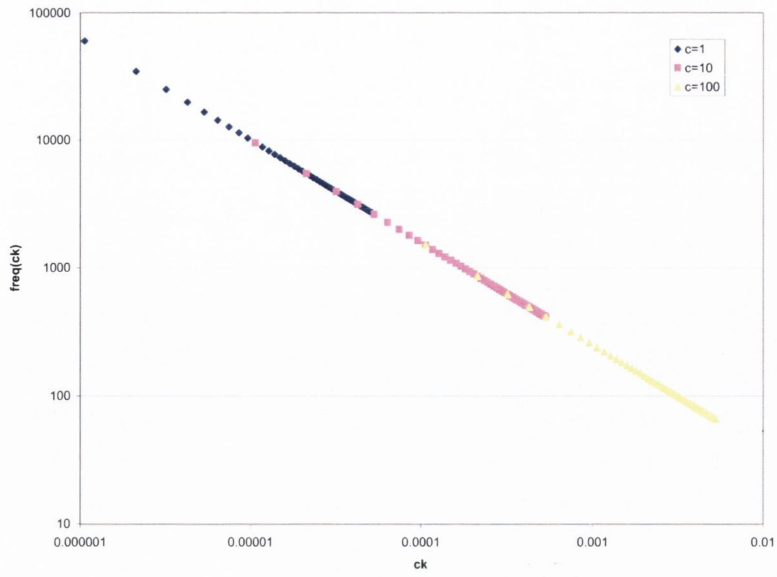


Figure 2.2: Demonstration of concurrence of scaled power law distributions

with a high in-degree is proportionally more likely to receive another in-link, further increasing its in-degree. This 'rich gets richer' relationship allows the scale invariance of the power law, as the probability of a node doubling its in-degree in any space of time is proportional to its existing in-degree.

Effects of the Scale-Free property of the Web

The fact that the Web is a scale-free network has been a contributory factor to its current form. A scale-free network naturally contains a few 'core' nodes, which have disproportionately more links than the vast majority of other nodes. These nodes are the cornerstones of the network, containing as they do a large proportion of its structure, as well as being highly 'visible'. There is some evidence of a relationship between Web document in-degree and traffic, as documented in Fortunato et al. (2006); Sydow (2005).

Since a large amount of the structure of a scale-free network[1] is contained within a small number of nodes, a scale-free network is more robust than random networks (in which the edge distribution follows a scalable distribution such as a uniform or normal distribution). In the same way as adding nodes preserves the overall structure of the network, a large number of randomly selected nodes can be removed from the network without losing much of the overall structure, so long as the few core nodes remain intact. This makes the Web fault-tolerant, on the whole, to server failure and downtime. However, a co-ordinated attack on the few core sites would result in the loss of a large proportion of the Web structure.

[1]The structure of a graph is defined by the edges between its nodes. The structure that a node 'contains' refers to the edges that are attached to it.

Scale-Free Networks and Authority

Due to the effect described in the above section, these core documents can be seen as 'important' documents in the Web. These documents and, consequently, documents that are pointed to by them, receive a large amount of page views. If a document is pointed to by a large number of core documents, this is interpreted as the document containing highly valuable information. Therefore, the authority of a page is related not only to its in-degree, but to the in-degree of pages that are linked to it. It is on this basis that PageRank is defined.

The PageRank and HITS algorithms take advantage of the connection between the scale-free nature of the Web and document authority in their search results ranking algorithms. Other hyperlink-based methods described later, such as Relevance Propagation and document clustering, do not typically assume a scale-free structure, or use this information in their algorithms.

2.1.2 PageRank

PageRank was introduced by Brin and Page, in 1998 (see also Page et al. (1998)) and has subsequently become the backbone of the Web search engine Google. Google has developed to become the most successful and ubiquitous Web search engine, partly due to PageRank, and partly due to other innovative projects (such as Google Maps, Google Image search), and its strong technical back-end that allows the swift processing of a large number[2] of queries on a databank of billions of Web documents. As stated in the previous section, PageRank measure document importance by identifying that the importance of a document is not only a function of its own in-degree, but of the importance of those pages that point to it. In other words, not all links are born equal - some confer more authority than others. For example, if Joe Bloggs' personal homepage contains a link to the microsoft.com homepage, that link will increase the authority of the microsoft.com homepage a little. However, a link on the microsoft.com homepage to Joe Bloggs' personal homepage would have a huge positive effect on its authority.

In PageRank, the authority of a document is independent of the search query.[3] This means that the authority of a document can be calculated off-line, and no further processing of the Web graph is required at query time. The authority value of a Web document, its PageRank, is used to order the document in a result list from a search query. The PageRank of a document d is based on the sum of the PageRanks of the documents that point to d, normalised by the in-degree of d.

$$R(d) = c \sum_{d,d' \in Ed_d} \frac{R(d')}{|Ed_d|} + cE(d) \tag{2.3}$$

Where: $R(d)$ is the PageRank of the document d

[2]ComScore (www.comscore.com) reports over 91 million queries made to Google in the U.S. per day

[3]Note that the Google search engine calculates the rank of a page by combining PageRank with an unknown number of text-based ranking schemes. The PageRank component itself is query-independent, but other components of the overall Google document ranking scheme may not be.

$Ed_d = \{x | (x, d) \in Edges\}$

$E(d)$ is a dampening factor designed to reduce feedback caused by mutual reinforcement within enclosed hyperlink loops.[4]

c is a constant normalisation factor.

This recursive definition is applied iteratively until convergence. In order to calculate the PageRank of a document according to this function, it is necessary to know the in-links of all documents on the Web. Since it is practically impossible to cover the entire Web (see (b) in the summary below - section 2.6), a large subsection of the Web graph must suffice. The Google search engine utilises a large number of Web crawlers that index new pages and add them to a representation of the Web graph. This information requires a large amount of data storage capability.

The Random Surfer Model of PageRank

The PageRank value of a document corresponds with the probability that a random surfer will visit the page. The dampening factor in the PageRank definition, designed to prevent mutually reinforcing feedback in hyperlink loops, is modelled by a probability that the surfer will jump randomly to another page in the Web, as opposed to following a hyperlink from their current location, thus exiting the loop. The behaviour of the random surfer can be modelled as a homogeneous Markov chain, where the state set is the set of Web pages (of size N) and the transitional probability between states i and j is:

$$\begin{cases} \frac{a}{\text{outdegree}(i)} + \frac{b}{N} & \text{if a link exists between } i \text{ and } j \\ \frac{b}{N} & \text{otherwise} \end{cases} \tag{2.4}$$

For some constants a and b such that the transition matrix of the Markov chain is stochastic.

PageRank then emerges as the stationary probability distribution of all states (Web pages) in the Markov chain.

Effects of PageRank

The effects of the PageRank algorithm are widely felt across the Web. Its effectiveness in document ranking on the Web led to Google becoming the most successful Web search engine. PageRank is now the most important factor in obtaining a high rank in Web search engines. As a result, a significant industry has emerged in PageRank optimisation. An advantage of PageRank over the earlier form of link analysis ranking, in-degree, is that in-degree is easily manipulable. To artificially increase their document's in-degree, an author need only solicit links from external

[4]This is typically uniform for all documents d, but the algorithm can be personalised by varying E.

pages to their document. In PageRank, the relationship between in-degree and rank is less direct;[5] these external pages need to have high authority themselves to confer authority to other pages. As a result, the search engine optimisers have been required to become more complicated, and link farms appeared. These link farms are made up of a collection of highly interlinked documents. Members of the link farms would benefit from the artificial PageRank score generated by the high number of interconnected links within the farm. This process is called 'link spamming' or 'spamdexing' (Metaxas and DeStefano, 2005; Gyongyi and Garcia-Molina, 2005).

Advantages and Disadvantages of PageRank

The principal advantage of PageRank is its ability to adequately capture the notion of authority on the Web. The authority calculation is carried out off-line, and does not increase the complexity of the querying operation. Once an internal representation of the Web graph is generated, altering this representation by adding or removing nodes is a relatively low-cost operation.

However, the cost of carrying out the authority calculation off-line is to make it query-independent. The authority of a document on a query subject is equal to its authority for all queries that return the document. Also, hyperlinks confer authority to a destination document regardless of the degree to which those documents are connected semantically, or the nature of that semantic connection. The semantics of the hyperlink on the Web are not well-defined, and the intention behind a hyperlink may vary greatly from hyperlink to hyperlink. For example, some hyperlinks are purely functional or structural (e.g. "Best viewed using Firefox", or links to indices or contents pages), as opposed to semantic. Therefore, some links may be of greater value than others when calculating the authority of a document with respect to a subject. A side-effect of query-independence is that, while PageRank is more stable than link analysis weighting methods such as in-degree, or purely text-based methods such as TF$\times$IDF, it is still susceptible to manipulation by unscrupulous search engine optimisers using 'link spamming'.

Another disadvantage of PageRank is that it requires detailed information about the topology (or hyperlink structure) of the Web, both to identify a document's in-degree and to allow the iterative update of PageRank. A large-scale internal representation of the Web graph is therefore required to calculate any document's PageRank. This requires large data storage capabilities, and makes it difficult to apply PageRank on top of the results of an external search engine.

Since Google has become the most popular search tool on the Web, it has the ability to influence the popularity of documents that it returns in results lists. Critics have argued that the PageRank model represents a dangerous form of 'search engine bias' by exacerbating the 'rich get richer' effect of the power-law distribution due to the larger α value observed from a PageRank distribution than that observed from an in-degree distribution, amplifying the dominance of already-established sites to the detriment of new ones (Cho and Roy, 2004; Vaughan and Thelwall, 2004). However, recent

[5]PageRank can be estimated from in-degree, however (see Fortunato et al., 2005a; Litvak et al., 2006).

empirical work (Fortunato et al., 2005b) refutes these claims, stating that while the random surfer model indeed causes a 'rich get richer' effect, search engines in fact serve to mitigate this effect, as the ability to perform text-based query matching against an index of documents causes a user's behaviour to diverge significantly from that of a random surfer.

2.1.3 HITS

In the same year as PageRank was introduced, Kleinberg (1999) presented the HITS (Hypertext Induced Topic Search) algorithm. The HITS algorithm also aims to identify authoritative documents on the Web, but does so by directly differentiating between hubs and authorities. Hub documents point to a number of authorities and authorities are pointed to by a number of hubs. The definitions of hub and authority documents are therefore mutually reinforcing. Each document has a hub and authority score - the hub score is the normalised sum of the authority scores of the documents it points to, and the authority score is, in turn, the normalised sum of the hub scores that point to it.

Formal Definition

In a directed Web graph $G = \{N, E\}$ of nodes N and directed edges $\langle n, n' \rangle \in E$ representing the existence of one or more hyperlinks from node n to node n':

$$\text{Auth}(n) \propto \sum_{<n',n>\in E} \text{Hub}(n')$$
$$\text{Hub}(n) \propto \sum_{<n,n'>\in E} \text{Auth}(n')$$

where: Auth(n) is the authority score of node n,
and Hub(n) is the hub score of node n

In vector terms, if A_G is the adjacency matrix of G, a is the vector of authority values of the nodes in N, h, in turn, is the vector of hub values, t is a time-stamp, and σ, σ' are normalising factors (real numbers).

$$a_t = \sigma A_G^T h_{t-1}$$
$$h_t = \sigma' A_G a_{t-1}$$

Therefore,

$$a_{t+1} = \sigma\sigma' A_G^T A_G a_{t-1} \tag{2.5}$$

and

$$h_{t+1} = \sigma\sigma' A_G A_G^T h_{t-1} \tag{2.6}$$

Since $a_{t'}$ is a simple scaling of $A_G^T A_G a_t$ (for all $t' > t$), a is the eigenvector of $A_G^T A_G$ (or the right singular vector of A_G).

Similarly h is the eigenvector of $A_G A_G^T$ (or the left singular vector of A_G). The HITS algorithm is therefore a version of the Power Iteration algorithm for eigenvector discovery.

Ranking using HITS

The definition of hubs and authorities closely associates the HITS algorithm with the scale-free nature of the Web network. However, unlike PageRank, which is calculated using the entire structure of the Web, a subset S of the Web graph is used to identify hubs and authorities. S is identified as follows:

1 The query is executed on a text-based search engine (AltaVista is used in the initial paper) and the top r ranked documents are collected and added to S.

2 This set is augmented by including all documents that are connected to any of the documents in S by one external out-link.[6]

3 In addition, if the number of external pages pointing to a document in the root set is sufficiently low (< 50 in *ibid.*), we add these documents to S.[7]

S is the base set for the HITS algorithm. Since it is generated according to a specific search query, HITS is therefore query-dependent, unlike PageRank, which generates an a priori value for each document independent of the query.

After the creation of the adjacency matrix A_G, the iterative process is repeated k times and the documents with the top c authority scores are returned as the ranked result set. The iterative procedure is shown (in *ibid.*) to converge, and a value of $k = 20$ is reported to be sufficient for the documents with the largest authority values to become stable. After iteration, the documents are then typically ranked according to their authority value, however, reporting the documents with the top hub scores can also be of interest.

Advantages and Disadvantages of HITS

HITS is query-dependent, which has both positive and negative effects when compared to the query-independent PageRank. Since HITS scores must be calculated at query-time, the computational cost of executing a query is higher than PageRank. However, since the HITS algorithm uses only a small sub-network of the Web in its algorithm, it does not need to iterate on an internal representation of the entire Web (or a large section of it). The HITS algorithm is therefore cheaper to implement and can be used on top of an external search engine. However, HITS still requires knowledge about the in-links of a document in order to locate documents for the base set, and for

[6]An external link is a link in which the source document is in a different domain to the destination document. Note, in section 1.3, it is suggested that the trend of user-driven content may weaken the effects of domain structure on the Web, and lessen the distinction between internal and external links.

[7]These documents are identified by querying a large index of the Web.

this knowledge to be reasonably complete, a large-scale crawl of the Web must have been carried out.

Another advantage of the query-dependence of HITS is that the conferral of authority by hyperlinks is more likely to be focused by the query, since the sub-network used by HITS is defined according to the documents' relationships with the query. However, although the sub-network S is defined relative to the query, the links connecting the documents in S may have no semantic relationship with the query, since the second and third steps in the procedure for the creation of S add linked documents irrespective of the initial query text. Therefore, the problem of conferral of authority by unsuitable links remains. Chakrabarti et al. (1998b) address this problem by weighting the adjacency matrix A_G according to the relationship between the hyperlink and the query. This relationship is defined by the 'anchor' (`<a></a>`) tags that denote a hyperlink on an HTML page. Anchor tags are used to weight links in Mcbryan (1994) and Li (1997). The weight of a hyperlink with respect to a query is determined by the number of times the terms in the query occur in the vicinity of the anchor tags. The 'vicinity' is a window of a fixed size (in characters) around the anchor tags (including the 'anchor text' - the text between the anchor tags themselves). The definition of the weighted matrix A'_S is then:

$$A'_{Si,j} = 1 + n(t) \tag{2.7}$$

Where $n(t)$ is the number of occurrences of the terms in the query within the anchor window of the hyperlink from document i to document j.

Although the resultant matrix contains many values that are larger than 1, the normalisation step in the HITS algorithm ensures the convergence of the hub and authority vectors.

This variation of HITS is called ARC (Automatic Resource Compiler), and its effectiveness is tested against manually-compiled resource lists for a number of topics obtained from Yahoo and Infoseek, two of the most popular Web information resources of the time. Volunteer users were asked to rate the manually-compiled lists and the automatic lists generated by ARC according to accuracy (related to precision), comprehensiveness (related to recall) and an overall value. ARC was found to give comparable results to Infoseek and results that are encouragingly close to Yahoo. These results foresaw the impending supremacy of automatic search engines based on link analysis methods over the more expensive manual resource lists. An interesting side-result is that the authors noticed that ARC often performed disappointingly in comparison to the manual resource compilers on political or economic topics that had a large number of hand-crafted resource pages on the Web, as opposed to more general topics or non-political or economic topics which had a comparatively lesser Web-presence, and contained pages with less well-designed structure, making manual resource compilation more difficult.

2.2 Relevance Propagation

Relevance propagation (RP) is a network-based Information Retrieval technique that predates the prevalence of the WWW and the subsequent common use of hypertexts as a medium. Original relevance propagation systems were based on artificial hypertexts generated using Nearest-Neighbour (NN) links (described below), or analysis of citations in academic papers. The RP model assumes that the retrieval status value (RSV - a document's relevance with respect to a query) can be influenced, or is dependent upon, the textual contents of other documents in the 'neighbourhood' of the document.

This assumption is heavily influenced by Van Rijsbergen's Clustering Hypothesis (Van Rijsbergen, 1979):

"Closely associated documents tend to be relevant to the same requests."

According to Shakery and Zhai (2003):

"The relevance propagation model naturally captures the intuition that a Web page's value depends on the page's content value (self-relevance) as well as the values of all the pages that are linked to this page"

This definition suggests an additive model in which the RSV of the document due to its textual content (textual RSV) is added to some function of the RSVs of the neighbouring documents. This method is related to 'spreading activation' (Cohen and Kjeldsen, 1987). The effect of the additive model is that documents are considered to be more relevant to a query if their neighbours are themselves relevant to the query.

RP attempts to adapt traditional text-based IR applications for use in a non-linear text system such as hypertext.

"Most of the conventional Information Retrieval (IR) algorithms have been developed for searching in large linear text collections. They are not suited to retrieve information in non-linearly organized hyper collections" (Frei and Stieger, 1992)

"This approach suffers from an intrinsic weakness: it doesn't take into account the Web structure the object is part of" (Marchiori, 1997)

2.2.1 A Typical RP Model

Most relevance propagation models have a number of similar features:-

- A typical model is iterative, i.e. the RSV is a function of the textual RSV and the 'hypertext' RSV, where the hypertext RSV is iteratively calculated as the RSV of the neighbouring documents.

- The relationship between the textual and hypertext RSVs is linear
- The textual RSV is typically computed by using a Vector Space Model (described later in section 5.1.2) cosine-similarity (or similar) method to compare the document contents with the query.
- The hypertextual RSV is a weighted sum of the RSVs of the documents in the neighbourhood of the document. The definition of a document's neighbourhood varies, but is defined by hyperlinks (typed or untyped) and the level of iteration often defines the scope, or spread, of the neighbourhood.

The information obtained by an RP model is assumed to be different to that obtained by link topology analysis ('authority') methods (see section 2.1), and RP models do not rely on a scale-free network structure in the hypertext, and often assume a non-scale-free network, with an average out-degree for nodes. Therefore, both systems can be of value in an IR environment.

An advantage of some RP systems over some link topology analysis methods is that no internal structure of the Web is required to carry out the iterative procedure, since the neighbourhood is typically locally bounded. They can therefore often be implemented as post-processors of external search results, making them cheaper to implement. This can vary if in-links are required, however.

2.2.2 Examples of Relevance Propagation Systems

Savoy (1994), during the onset of the WWW explosion, proposed a learning scheme using RP on a network based on 'relevance feedback'. The learning system developed a network on a set of documents in which results from previous queries establish 'relevance links' between documents that were correctly retrieved together.[8] Negative information from queries, such as information about document irrelevance to a query, is not used. Each relevance link is weighted by the number of times both nodes were found to be relevant together. The result is an undirected graph.

Savoy applies a variety of traditional text-based IR systems: the Boolean model, the p-norm model (Salton et al., 1983) and the probability model used by Croft and Harper (1997) to the document collection to obtain the textual RSV of the documents with respect to a query. The RSV is then adapted to incorporate the RP component, using:

$$RSV_{\text{new}}(D_i) = RSV_{text}(D_i) + \sum_{k=0}^{\|D\|} \alpha_{ik} RSV_{\text{text}}(D_k) \tag{2.8}$$

for all documents D_i in the set D of indexed documents, where α_{ik} is the weight of the relevance link between D_i and D_k.

[8] Related work involving the use of relevance links in hypertext generation and browsing can be found in Golovchinsky (1997).

The RP method applied to relevance information is weakened by the fact that relevance itself is query-dependent, and that "a large paper is not normally concentrated on one narrow subject; rather, it must be considered as a many-faceted entity" (Savoy, 1994). Therefore, relevance links generated from the results of one query may provide misguided information for another query. Nonetheless, the paper reports an impressive improvement of retrieval scores (an improvement in average precision values of up to 74%) when compared with the text-based IR methods without the use of relevance feedback. However, this system does not use Web hyperlink information, and its use on the Web is limited as relevance feedback information is difficult to obtain and verify.

Frei and Stieger (1992) (also 1995) identified the advantage of hypertext over linear text:

"The main idea of hyper documents is that documents - or parts thereof - can be brought into relation to each other..."

and applied a similar technique to Savoy to the hypermedia domain. They suggested that hypertext links can, in general, be categorised into two broad types, referential and semantic links. Referential links exist purely to aid document browsing, and do not provide additional information to the topic of a document, while semantic links point to "similar, more detailed or additional information on a specific topic". They proposed that while referential links are merely structural tools to aid browsing and reading, semantic links can be used to augment the information content of a document, and can therefore be used in an IR system.

A number of prerequisites are associated with their system; links are assumed to have known types, i.e. they fall cleanly into one or other category, and semantic links are associated with link descriptions. These descriptions indicate whether or not a link will be of use to an IR system, given a specific query.

"Links are only followed when they promise to point to nodes containing information relevant to the query." (*ibid.*)

A strategy for creating the link descriptions is to generate them automatically by combining the contents of the source and destination documents. Alternatively, links may have user-annotated information associated with them, that can be used as the link descriptions in this system. A third approach would be to include both user-generated information and the automatically generated information described above.

The method is tested on the INSPEC collection (Harding, 1982) of science and engineering abstracts. In these tests, the links between documents were generated by comparing document descriptions. Links were established between documents that shared description terms. These descriptions were also used to artificially generate the user-annotated link information. The resultant hypertext is an undirected graph.

The RP model used in this paper is:

$$RSV_{d+1}(D) = RSV_d(D) + w_d \cdot \sum_{D' \in \mathrm{Nb}_d} (RSV_0(D')) \tag{2.9}$$

Where $D, D' \in$ Documents.

The set Nb_d is generated by ignoring all links with descriptions that are not sufficiently similar to the query, and following the resultant links to a depth of $d+1$ from document D. The documents at the end of these paths of length $d+1$ form the set Nb_d. w_d is the weight associated with documents of distance $d+1$ from D (a propagation factor), and
$RSV_0(D) = RSV_{text}(D)$.

The paper finds that in a dense hypertext such as that generated by the method above, there is no need to extend beyond a distance of 1 from the start document. In other words, the RSV of D is simply made up of the textual RSV of the document and the textual RSVs of documents directly linked to it. In this case, the model used in this paper is equivalent to that used in Savoy (1994) (equation 2.8), albeit with no relevance feedback information (α_{ik} is equal for all i, k).

This method also has limitations in a Web environment, due to the scarcity of link descriptions and the difficulty of categorising links into distinct types.

In 1996, Savoy extended the RP model he used previously (Savoy, 1994), described above, to hypertext systems. The CACM collection of computing paper abstracts and citations was used to test the model. A number of hyperlink systems were induced from this structure:

- Bibliographic reference links: In this system, links are generated between documents if one document cites the other in its reference list. These links (commonly simply called 'citation links') are similar in semantics to a number of Web links (in that they represent information that the author of the document considers to be related to the document), and share with the Web the property of being uncontrolled by the destination (cited) document. These have been used in the past to confer 'authority' to documents (see section 2.1).

- Bibliographic coupling links: Links are generated between documents that cite a number of the same papers. The intention behind this method is that such documents are likely to have similar topics. These links and the resultant graph are undirected.

- Co-citation links: Links are generated between documents that are cited by the same papers. Intuitively, if two documents are frequently cited together by a number of separate documents, their subjects are related and complementary; however, their contents may be sufficiently different to warrant co-citation without repeating information. These links and the resultant graph are also undirected.

- Nearest Neighbour (NN) links: This is the only link type that is based on the contents of the documents. In this system, documents are compared on the vector space (see section 5.1.2), and the nearest n documents to each document are linked to the document. These links can be directed, as the nearest neighbour to a document may itself have a different document as its nearest neighbour. However, it is assumed that in this work the links are undirected.

A variety of schemes were suggested and tested for the link weights (α):

- A single constant value, determined by experimentation
- A value relative to out-degree (used in bibliographic reference and bibliographic coupling links)
- A VSM similarity measure between the query and a link description generated in the same way as in Frei and Stieger (1992, 1995) (see above).

Savoy finds that all except the NN links give a significant improvement over the non-link-aware baseline, with the best results obtained when using a constant link weight and bibliographic reference links. Despite VSM-based link information being useful to Frei and Stieger, NN links were found to add noise to the IR system, suggesting that the method of incorporation of VSM-based data into a hypertextual IR system is vital. Savoy also notes that the variation between the effectiveness of text-based retrieval systems (Boolean, TF$\times$IDF Boolean, TF$\times$IDF VSM and hybrid models were tested) used as the initial text-based retrieval systems in this work is greater than the different hyperlink methods, and warns that while hyperlink information is useful, it cannot compensate for a poor choice in underlying text-based retrieval systems.

Marchiori (1997) specifically addresses information retrieval on the Web, and approaches it from a Web-browsing viewpoint. In this context, the neighbourhood of a document is identified as documents that can be reached (browsed to) from it. As a result, these documents are 'potential information content' with respect to the Web space. This is related to the concept of context in the Web. Marchiori also highlights the problems with regard to existing link-analysis based Web search engines, such as Excite, that use visibility (in-degree - see section 2.1) to rank pages:

"Visibility says nothing about the informative content of a Web object"

and

"visibility [...] provides information which is disjoint to hyper information"

Hyper information, to Marchiori, is the potential information contained in the neighbourhood of a document. Since a large amount of potential information around a document increases the value of a document to a user, this information should be included in the RSV of that document.

The RP model used by Marchiori is:

$$\text{Information}(d) = \sum_{i}^{0 \rightarrow n} (F^i \cdot \text{TextInfo}(d_i)) \tag{2.10}$$

Where d_1, d_2, d_n is a sequence of documents in the neighbourhood of d, in which documents of distance $p+1$ from d have higher indices than documents of distance p, and documents with equal distance from d are ranked according to their information content relative to a query (d_0 is equal to d).

Using this method, with F as a fading factor between 0 and 1, produces an exponential fading law. The influence of a neighbourhood document on d is inversely proportional to the 'cost' involved in reaching that document. Documents that are further away from d have a higher cost attached to them, and given a choice of links from a document, a (perfect) user is more likely to choose the one with the most relevant content, so its cost is lower than another link at the same distance.

Since this method does not use in-links to define document neighbourhoods, it is an example of an RP system that is cheap to implement.

Shakery and Zhai (2003) presented an RP method for Web information retrieval that used language models (described in chapter 6) to determine the text similarity measures. In different variants of their model, either in-links or out-links were used. In both cases, the hypertextual RSV component (the component of the RSV based on relevance propagation) is weighted by the textual RSVs of each of the target documents, modelling the fact that a user would follow a hyperlink from their current position if the target link is related to the query. Formally, for in-links:

$$RSV(d) = RSV_{text}(d) + \alpha \cdot \sum_{d' \to d} RSV(d') \cdot w_{in}(d' \to d) \tag{2.11}$$

where $w_{in}(d' \to d)$ is proportional to $RSV_{text}(d)$

and for out-links

$$RSV(d) = RSV_{text}(d) + \alpha \cdot \sum_{d \to d'} RSV(d') \cdot w_{out}(d \to d') \tag{2.12}$$

Where $w_{out}(d \to d')$ is proportional to $RSV_{text}(d')$

The decision to use in- or out- links corresponds to a different model of user behaviour. The authors show that if in-links are used, the resultant model is similar to the PageRank 'random surfer' model, whereas if out-links are used, the resultant model is related to the Marchiori system described above. The results presented in this paper are negative, and appear to be partly due to the small average number of relevant documents in the test corpora available to the system.

A variation of the RP model, that is not strictly an RP model itself, is Sugiyama et al.'s proposition for TF×IDF refinement for the Web (Sugiyama et al., 2003). This model does not use retrieval status values themselves, and is query-independent. Instead it adapts the TF×IDF scores of terms in a document according to the contents of its neighbours. The resultant TF×IDF scores will subsequently produce RSVs for the document that are influenced by the neighbours of the document, without having to calculate the relationship at query time.

Three methods are proposed to adapt TF×IDF (w in the equations below) for a term t in a document d:[9]

1. $$w(t,d) = w_{text}(t,d) + \alpha \cdot \sum_{d'} \frac{w_{text}(t,d')}{\text{distance}(d,d')} \tag{2.13}$$

Here the new weight for t in d is the combination of its $TF \times IDF$ value in d and the sum of the

[9]The formulae in Sugiyama et al. (2003) differentiate explicitly between in-links and out-links. This difference is ignored in the formulae below for simplicity.

$TF \times IDF$ values in its neighbours, each mitigated by the VSM Euclidean distance between d and its neighbour. For example, the TF $\times$ IDF value of the term 'Trinity' in the Trinity College Admissions web page would be bolstered by the fact that its neighbours each contains the word 'Trinity', and that these neighbours also share a significant amount of other terms, such as, 'University', 'Dublin', 'College'. However, the term 'Admissions' would receive less of an increase, since fewer of its neighbours would share this term.

In the second approach, for each depth level away from d, a group of documents is created, and then the k-means method is used to generate a number of clusters from each of these groups. An abstraction function using the centroid of the cluster then generates weights for terms within the clusters. Again, these values are mitigated by the VSM Euclidean distance between d and the cluster.

$$2.\ w(t,d) = w_{text}(t,d) + \sum_{level} \sum_{k \in clusters} \frac{w_{text}(t,k)}{\text{distance}(d,k)} \tag{2.14}$$

The third method is similar to the second, but the clusters are defined differently. Instead of a separate group per depth level, the group at level $i+1$ subsumes the group at level i. This approach reflects the real-life subsumption of topics within broader topics.

$$3.\ w(t,d) = w_{text}(t,d) + \sum_{k \in clusters} \frac{w_{text}(t,k)}{\text{distance}(d,k)} \tag{2.15}$$

Each of these methods is based on a compromise between hypertext similarity and VSM similarity. The purpose of this is to reduce the effects of unsuitable neighbours on the retrieval value of the document. Since this method is query-independent, this mitigation takes place by comparing the document to its neighbours (rather than by comparing the query to the neighbours as above). This assumes that if a neighbour is not sufficiently similar in terms of textual content to a document, it is of no value. The clustering methods also mitigate the effects of unsuitable neighbours by effectively averaging over the contents of the clusters, reducing the effects of 'outlying' documents. As with the RP methods, then, the focus is on document similarity rather than context. In other words, neighbouring documents are only of interest if they are textually similar to the query or to the focus document. In contrast, the definition of context used in this thesis incorporates the case in which neighbouring documents contain different information.

In 2006, Shakery and Zhai adapted their RP model by establishing it as a probabilistic model, in which all parameters are probabilities that can be set in a number of ways. This model is flexible enough to be able to combine RP and link-analysis models, such as HITS and PageRank, into a single model depending on the chosen values of the parameters (Shakery and Zhai, 2006).

In general, the method follows a similar path to other RP models:

1 Compute the content-based relevance probability of d

2 Propagate probabilities through neighbour sets.

Neighbour sets can be defined in a number of ways. Example sets are all documents that point to d, all documents that are pointed to by d, the document d itself only (the null neighbour set) or the set of all documents with links to d (in either direction). There is provision for the treatment of different neighbour sets in different ways.

The model is defined similarly to their previous RP model, as:

$$p(d) = \sum_{i}^{k} \alpha_i \sum_{d'} (p(d')p_i(d' \rightarrow d)) \tag{2.16}$$

where:

$p(d)$ is the probability that a surfer will arrive at document d (After searching, in an IR context). This is set initially according to a content similarity score generated by Okapi (Walker et al., 1997) or language models.

α is the probability attached to a particular neighbour set, and

$p_i(d' \rightarrow d)$ is the probability of a link between d' and d in neighbour set i. If such a link does not exist in this set, the probability is zero, otherwise it is a positive value.

Depending on the values chosen for the parameters, the behaviour of the model can change greatly. Example parameter settings for varieties of PageRank and HITS as well as Chakrabarti et al.'s HITS-based ARC are given in the paper. The paper reports results that correlate with results obtained from tests on models that are replicated by this system. The advantage of this system is that it provides a generalised framework for comparison of a number of link-based IR models. Further advantages and disadvantages associated with this method are those associated with the models it replicates. For example, if in-links are defined as a neighbour set, this method is also affected by the problem reported previously with regard to location of all the in-links pointing to a document.

2.2.3 Summary

Relevance Propagation methods have been extensively studied in information retrieval, on the hypertext domains of the Web, academic publications with citation information and hypertexts by relevance feedback and nearest neighbour information. In general, it is seen that incorporating information about the relevance or similarity of neighbouring documents in a hypertext is beneficial to an IR system. However, improper use of this information can have a damaging effect on IR precision.

2.3 Clustering

On the Web, the notion of context as described in chapter 1 is closely related to the notion of clustering. Clustering uses a defined distance measure between elements of a data set to classify them into subsets, so that they can be processed more efficiently, or meaningfully. In this case, the elements are Web documents. Depending on the distance measure used, the resultant clusters can be interpreted similarly to a 'context set'. A hypertext medium such as the Web has an explicit topography on which to base a distance measure in the form of the hyperlink structure. Furthermore, due to the essentially disorganised nature of the Web and the lack of uniformity of semantics of document links, document relationships cannot be easily classified simply by observing the link between two documents. Clustering allows the automatic aggregation of sections of the Web in order to generate a new structure that is either simpler to visualise or browse, or more useful to information retrieval algorithms. The most common variety of clustering algorithm used in the applications described in this section is Agglomerative Hierarchical Clustering. Agglomerative hierarchical clustering methods (e.g. Single or Complete Linkage Clustering) have the advantages of being stable (the same result is obtained with each run), and of providing a tree-structure output. A tree structure can be useful for browsing and retrieval, as it allows searches to be restricted to subsections of the dataset by following only one of the sub-nodes of the current node at each step. This section describes the most common clustering methods and their uses on hypertext or Web environments. We then compare these clustering methods to the definition of context described in section 1.4.

2.3.1 Document Similarity in Hypertexts

Traditional document clustering typically used a VSM-based document similarity measure, such as cosine similarity on documents' TF$\times$IDF vectors, as a distance metric. As mentioned above, hypertexts include an explicit means of computing documents' distance, using the hyperlink topology. According to Flake et al. (2002), linked documents are better indicators of relatedness than textual similarity, so hyperlinks can be used as a means of generating potentially more useful clusters on hypertexts. The manners in which the hyperlinks are used to produce a distance measure vary, and some examples are described below. In general, hypertext clustering applications can be separated into those that use text, links or both to calculate document similarity.

2.3.2 Uses of Clustering

Pre-Web

Aside from the large number of non-hypertext information retrieval and data mining applications that use clustering, it has also been used to induce a hypertext structure on a document collection with no explicit links. An important paper that represents early work on clustering in hypertext is Crouch et al. (1989). In this work, a single-linkage clustering algorithm is used to generate a hierarchy of document clusters using a term-frequency based VSM similarity measure. A document browser is provided that allows users to view the cluster hierarchy and search through this structure, in the context of a query. Crouch et al. also describe how such a structure can be used for efficient automatic information retrieval, using either a top-down or bottom-up method to traverse the tree to locate matching document nodes. These methods compare cluster centroids with query strings as described above.

Using the top-down method, the algorithm starts at the root node of the cluster tree, at every step, the similarity between the query and each of the child nodes is calculated and the child node with the highest similarity is followed, until either all the children of the current node are documents, or all children have a lower similarity to the query than the current node. The documents that make up the current cluster are then reported as the result list.

With the bottom-up method, the leaf (document) node most similar to the query is chosen as the starting point. At each step, if the parent node has a higher similarity than the current node, the parent node becomes the current node. If not, all documents that make up the current cluster node are reported as the result list.

In Allan et al. (1997), text-based clustering is also used as a means of inducing a hypertext-like structure on an unstructured document collection. The emphasis in this case is visualisation of a set of retrieved documents from a query. These documents are mapped onto a 2D or 3D visualisation space using a spring layout algorithm on a graph represented the document set. The edges in this graph are weighted by a text-based document similarity matrix (Allan, 1996). Relevance feedback is also used to bring documents similar to previously relevant documents closer together and spread apart likely irrelevant documents. Hierarchical clusters are then generated by clustering documents within spherical regions in this low-dimension space for easy visualisation. A similar approach is used in Zizi and Beaudouin-Lafon (1994), where documents are clustered using a spring layout on a 2D environment. This work is applied to hypertext documents, though, where the document similarity weights are based on the shortest path length between two documents.

Another hypertext-induction work, HypIR (Can and Lee, 1993), combines text-based clustering with a number of linking methods to induce a hypertext from an unlinked document collection. Links are generated between documents that share an author, co-occur in a publication or are co-referenced by another publication. Nearest neighbour links are also generated. In addition to these links, users can also browse a document's cluster. The clusters are calculated using a text-based inner product off-line (i.e. pre-query). This approach uses document meta-data such as author and publication name that do not appear regularly on the Web, but the concept of augmenting a document's existing links with text-based nearest neighbours can be, and has been, applied to the Web (e.g. CiteSeer - `http://citeseer.ist.psu.edu`). A problem with applying such a system to a broad section of the Web is that text-based clustering and nearest neighbour methods require the calculation of a document-document similarity matrix – a collection-based structure, as opposed to a localised structure such as a context set or other hyperlinked based structures. To be applied to a large Web collection, therefore, a large-scale index would be required. The application would therefore be expensive to run on large-scale Web document collections.

Web and Hypertext Clustering

The above methods demonstrate two prominent uses of clustering in hypertext environments – improved document retrieval, and enhanced document set visualisation. The following works focus on these uses of clustering hypertexts and Web documents.

Document Retrieval

The core concept behind the use of clustering in document retrieval is equivalent to the intention behind Relevance Propagation (section 2.2), the clustering hypothesis. Here, instead of spreading relevance to a query around a graph, content from the graph is aggregated in order to calculate RSVs.

Grouper (Zamir and Etzioni, 1999) is a search engine post-processor that clusters search results in order to make them easier to browse. Documents are clustered according to textual similarity only, using a measure of the overlap of document indices. Since it is a post-processor, the cost overhead of applying text-based clustering on a large document set is largely avoided. A reported disadvantage of using this similarity measure is that semantic distinctions expected by users are often not accurately captured by the clustering mechanism. A hyperlink-based analysis applied to the results may go some way to correcting this, as semantically related pages are likely to be more closely linked.

HyPursuit (Weiss et al., 1996) is a search engine based on a hypertext. It uses both text and hyperlinks to calculate a distance measure for a complete linkage clustering algorithm. A variety of abstraction functions are applied to the resultant clusters. These abstraction functions allow the clusters to be used to increase the efficiency of a query processor, and to aid browsing or query refinement.

The distance measure in HyPursuit incorporates a content similarity and hyperlink cohesion component:

$$Sim_{i,j} = F(S_{i,j}^{text}, S_{i,j}^{link}) \tag{2.17}$$

The function used in the prototype is the max function.

The content similarity component (S^{text}) is calculated as the dot product of document vectors. The term weight function is a normalised variant of term frequency, incorporating a term attribute weight that is based on its appearance in various HTML tags (e.g. Title, Header, Keyword).

The hyperlink cohesion component (S^{link}) is a linear function of three link-based variables:

- Number of common descendants between two documents (weighted by the shortest path lengths to them)
- Number of common ancestors between two documents (weighted by the shortest path lengths to them)
- Shortest path length between the two documents

This method requires the knowledge of in-links and is therefore difficult to implement on the Web, without an internal representation of the Web graph. Additionally, as with all clustering methods that use text-based similarity, calculating a document similarity matrix for a large number of documents is expensive. The use of HTML tags to identify semantic importance of content is also problematic on the Web (Hodgson, 2001).

Aguiar (2003) describes a method that uses document clustering to identify contextual information in Web collections, and to incorporate this contextual information in a Web document Retrieval model. The model is based on two indices, one using document content, and the other using document context. A two-part query language is then suggested, allowing the user to specify a set of search terms to match documents' contents and another to match their contexts.

'Context' in the above paper is defined by a 'complementarity' measure between documents. This measure is similar to the two-component similarity measure used in HyPursuit. Document pairs that have a high complementarity score are assumed to have similar subject-matter, and are clustered together. An abstraction function is then used to derive a 'context node' from each cluster. The set of terms in a context node is equal to the terms in each of the documents in the cluster, and an adapted form of TF$\times$IDF is used to assign weights to these terms relative to their frequencies in the cluster. These term weights are then used to generate the context index.

The complementarity measure is a combination of content similarity of two documents and their structural proximity, based on the link structure of the hypertext.

Content Similarity:

The content similarity of two documents is calculated using the VSM cosine similarity measure. Initial term weights are calculated using TF$\times$IDF, and terms that occur in HTML emphasis elements such as `<title>` are assigned higher weights.

Structural Proximity:

The structural proximity measure is a linear function of four components; number of common ancestors, number of common descendants, number of independent paths and the length of the shortest path between two documents:

$$Prox_{struc}(d_i, d_j) = \beta \cdot S_{i,j}^{anc} + \gamma \cdot S_{i,j}^{des} + \delta \cdot Con_{i,j} + \lambda \cdot S_{i,j}^{spl} \tag{2.18}$$

The first two terms in this equation are based on the number of common ancestors and common descendants of documents d_i and d_j, and are calculated as in HyPursuit.

The third term is a measure of how tightly the two documents are connected and is a function of the number of independent paths from d_i to d_j ($S_{i,j}^{Indep}$) and from d_j to d_i ($S_{j,i}^{Indep}$). This is calculated as the cosine of the angle between the vector ($S_{i,j}^{Indep}$, $S_{j,i}^{Indep}$) and (1,1) and is at a maximum when $S_{i,j}^{Indep}$ is equal to $S_{j,i}^{Indep}$.

The final term is simply the reciprocal of the shortest path between d_i and d_j in either direction.

In the experiments described in the paper, the parameters β, γ, δ and λ were set to be equal.

The above definitions of content similarity and structural proximity allow the calculation of the complementarity measure:

$$Compl(N_i, N_j) = \alpha \cdot Prox_{struc}(d_i, d_j) + (1 - \alpha) \cdot Sim_{cont}(d_i, d_j) \tag{2.19}$$

Context Nodes:

This measure is then used to generate document clusters. Two clustering methods, the *star* method[10] and complete-linkage clustering were used to generate the clusters. Each cluster represents a set of documents that shared contextual information. An abstraction function is applied to these clusters in order to produce the context nodes, and each original document d is attached to the context node of its cluster. The strength of the attachment is proportional to the overall complementarity between d and the other documents in its cluster.

Each context node contains the terms found in the original documents, weighted according to the following formula, based on TF$\times$IDF:

$$w_{ik} = ntf_{ik} \cdot nidf_k \tag{2.20}$$

where:

$ntf_{ik} = \frac{tf_{ik}}{max_z tf_{iz}}$,

$nidf = \frac{idf_k}{\log(n)}$,

$idf_k = \log(\frac{df_k}{n})$,

tf_{ik} is the number of page nodes in the cluster i, in which term k occurs,

n is the number of context nodes,

df_k is the number of context nodes which contain term k.

By treating context nodes as documents in themselves, and by defining links between context nodes when links exist between their underlying documents, a context hierarchy is established by

[10]The star method generates overlapping clusters according to an undirected document graph where an edge exists between two document nodes if their complementarity is above a given threshold. A cluster is generated for each node made up of the node and those attached to it by an edge. Clusters that are subsumed by a larger cluster are discarded.

repeatedly carrying out the above procedure on the context nodes themselves. In this extended definition of context nodes, the attachment between an original document d and a context node c in the hierarchy is equal to:

$$\max_i Att(d, c, path_i) \tag{2.21}$$

where $Att(d, c, path_i)$ is the product of the attachments between nodes on path i from d to c.

Indexing and Searching using Context Nodes:

The context index is generated from the context nodes by calculating the term weights as described above. The attachment values between all documents and all context nodes connect the context index to the document content index.

A search on these indices uses a two-part query: $q_{subject} + q_{context}$.

The document content index is used to return a set S of documents that contain the terms in $q_{subject}$, and the context index returns a set C of context nodes containing the terms present in $q_{context}$. The relevance of a document in S is then related to its relevance to $q_{subject}$ and its attachment to context nodes that are highly relevant to $q_{context}$.

Evaluation:

The relevance-ranking scheme described above was evaluated against the query log of an existing search engine. Two-term search queries were converted into two-part $q_{subject}+q_{context}$ versions with the help of a thesaurus. If the thesaurus defined one part of a two-term query to be a semantically broader version of the other part, this was the context term of a two-part query, and the other part was the subject term. The method was tested against a benchmark method that applied the entire query to the subject index alone. Both the single-context-level and context-hierarchy methods were found to retrieve relevant pages that the benchmark method did not. However, they also retrieved a large number of non-relevant documents.[11] The single-level method performed slightly better over the benchmark than the context-hierarchy method, and retrieved the vast majority of the relevant documents that the hierarchy method retrieved, suggesting that the context hierarchy adds little to the method's effectiveness.

Potential Drawbacks:

While the method does retrieve relevant documents not retrieved by the benchmark method, the paper does not provide precision and recall results in comparison to the benchmark.

The clustering portion of the method requires the construction and manipulation of a matrix of complementarity values between documents. This is an expensive operation for large-scale

[11]This paper does not use a precision-recall-based evaluation system, making it difficult to empirically compare its effectiveness with other methods.

collections. Furthermore, the insertion and update of documents requires a re-calculation of the matrix and the clusters. This makes the method unsuitable for dynamic Web sites, such as blogs or 'Wikis'.

The method also requires the user to split their search query into two segments, which may be unsuitable for a large proportion of users' information requirements. Also, due to the current prevalence of single-query search tools, such as Google, a system that requires a different search query formulation may be disadvantageous. Other methods that use extended query languages are described in section 2.4. However, user habits can change over time with regard to formulating their information need.

Eiron and McCurley (2003) recognise that Web pages are often more suitably treated as document segments, rather than as independent documents, and aim to identify compound documents broken up over multiple URLs using a number of heuristics based on link patterns and link types based on URL directory structure (similar to HyPursuit). They suggest that compound documents should be indexed together instead of separately in order to improve recall, and that this will encourage users to provide more specific queries (since the returned documents will be more general), eventually resulting in greater precision. Note that related documents, or documents that may contain information useful to an IR system, are not included in their method, unless they are considered part of a compound document. Their definition of context is therefore more restrictive than the definition in this thesis.

Visualisation & Browsing

A large amount of work has been carried out on using clustering techniques to improve graphical displays of hypertexts. Overview maps of hypertexts were designed to avoid the 'lost in hyperspace' problem (Conklin, 1987). However, these maps typically become cluttered as the number of documents and links increase, so attempts have been made to simplify the structure by clustering related documents. Work carried out in this area often uses hierarchical clustering methods, allowing users to control the level of granularity, or complexity, in the structure that they are shown, by choosing the level in the cluster hierarchy. An overview can show a small number of high-level clusters, and individual clusters can be explored by traversing the tree structure. An example is CyberMap (Gloor, 1991), in which documents are clustered into 'hyperdrawers' according to the inner product of their term vectors as described in Salton (1989).

Liu et al. (2002) describe a method for comparing Web sites by visualising their documents in a hierarchical tree of clusters. The method is aimed at the business world, where users may wish to investigate competing Web sites, and identify differences and similarities between them. The documents from the two Web sites are combined, and clustered by a hierarchical clustering algorithm using text-based VSM document similarity. These hierarchical clusters are then visualised as a tree, with the contents of each cluster being represented with different colours depending on the Web site they belong to. If a cluster is mostly made up of one colour, this suggests the presence of information that one site has and the other does not. If a cluster has evenly mixed colours, the information is shared, and the user can move down the tree to find a lower-level divergence.

Botafogo has proposed two link-based methods of clustering a hypertext for improved browsing and visualisation. He emphasises that clusters should be 'reasonable' or 'make sense' to a user, in that it should be possible to explain the reason why a particular document belongs in a cluster. Both methods use link-based similarity measures only to identify 'strongly connected components' in the graph. The first (Botafogo, 1993) clusters documents by the number of independent paths between them (k-components of the Web graph). Documents are put into k-components (note, a document in an i-component is also in a potentially larger j-component, for all $0 < j < i$). Documents in the strongest k-component (largest value of k) are then aggregated into a single node and placed in a new graph. The next strongest k-component is then identified and placed into the new graph, and if links are present between documents in this k-component and the strongest one, a link is added between the aggregated nodes in the new graph. This continues until all k-components have been identified and aggregated.

In Botafogo and Shneiderman (1991), the 'Compactness' C_p of a graph $G = \{N, E\}$ is defined as:

$$C_p(G) = \frac{Max - \sum_i \sum_j C_{ij}}{Max - Min} \tag{2.22}$$

$\forall i, j \in N$

where:

C_{ij} is the shortest path between documents i and j.

K is a constant used in place of infinity when two documents i and j are not connected

Max is the maximum value of $\sum_i \sum_j C_{ij}$ in an unconnected graph and is set to:

$$Max = (n^2 - n)K \tag{2.23}$$

Min is the minimum value of $\sum_i \sum_j C_{ij}$ in a fully connected graph:

$$Min = (n^2 - n)1 = (n^2 - n) \tag{2.24}$$

Botafogo defines a 'semantic cluster' as a subgraph of the Web graph for which the compactness of the subgraph is higher than the compactness of the overall graph. Note that this results in a hierarchical structure of clusters. Botafogo identifies such subgraphs using a recursive method to locate bicomponents (subgraphs with at least two (undirected) paths between any two nodes in the subgraph), and suggesting that such bicomponents are likely to have high compactness. This algorithm has linear complexity with respect to the size of the graph ($O(N+E)$), and can therefore be scaled reasonably well to large-scale Web graphs.

This method is also used by Bernstein et al. (1992), who uses strongly connected components to describe what the author calls the 'literary' nature of hypertext, and describes documents using a 'texture' or 'capacity for choice' document measure (similar to PageRank, but applied to out-links), that is related to its connectivity.

Zhu et al. (2004) describe PageCluster, an application that aims to improve visualisation of a large Web site by inducing a hierarchical tree structure on the directed graph structure of the hypertext. The tree is then simplified by clustering nodes at each level of the hierarchy.

As a first step, a document is identified as the start or home page of the Web. The initial tree is generated by classifying documents into levels according to the number of links in the shortest path from this start page. Links are then classified into two categories: structural links (links to a destination document at a lower level) and secondary links (links to a document at a higher level). Secondary links are removed and the result is a tree-structure with the start page as the root.

In order to simplify this tree structure, documents at each level are clustered by a complete-linkage algorithm using link-similarity as a document similarity measure. A document-document matrix is generated from the graph adjacency matrix using Web usage statistics,[12] where the value at position *i*,*j* in the matrix is based on the number of users that traversed a link between documents *i* and *j*. Documents are represented as vectors in this matrix, and link similarity is calculated as the Euclidean distance between document vectors. The authors differentiate between 'category clusters', that are based on in-link similarity only, and 'navigation clusters', that are based on in- and out-link similarity.

Pirolli et al. (1996) aim to automatically identify structures and document types on the Web. Their methods are a combination of text-based and link-based similarity analyses, and also incorporate document meta-data and usage statistics. Each document is defined by a vector of features based on this information and are classified according to these features into functional categories such as index, personal home page, and content page. The features are:

- Document size
- Number of in- and out-links
- Document request frequency
- Number of times the document was used as the start of a path traversal
- Textual similarity between the document and its child documents (calculated using the VSM dot product similarity measure)

[12]Web usage statistics are statistics collected by Web servers that record the pages viewed and links traversed by browsers (identified by IP addresses). They are expensive to collect, maintain, and store, which partly explains why they are not more frequently used in the applications described here

- Average depth of the child nodes of the document, determined by the number of '/' symbols in their URLs.

These features are weighted manually with respect to each category.

The authors also describe a spreading activation method of locating documents related to an initial set of documents by propagating an 'energy' value through three types of graphs, generated from (a) hyperlink structure (an edge exists between document nodes in the spreading activation graph if at least one link exists between the documents), (b) text-based similarity, where the weight of the edge between two document nodes is the dot product of the document term vectors, and (c) usage statistics, where the weight of the edge between two documents is the number of users that travelled between the documents. A start node, or set of start nodes, in one of the graphs is initialised with an energy value, which is then divided among the adjacent nodes in the graph. The process is then repeated, with energy being divided among adjacent nodes at each stage, until the energy being shared at a given node falls below a threshold, which stops it from being spread from that node.

The authors suggest a number of uses for the information obtained from the Web page category information and spreading activation on these graphs, such as predicting the interests of home page viewers. Both of these information sources incorporate usage statistics, that are often difficult and expensive to obtain. This therefore restricts the potential of this method for wide-scale implementation.

Pitkow and Pirolli (1997) propose two clustering methods to be used in Web visualisation applications. In contrast to the above paper, these methods are purely hyperlink-based, avoiding text-based similarity methods which they claim are unsuitable for the Web, and usage statistics, and employing only co-citation links between Web documents instead.

The first method uses pairs of co-cited documents as a seed for an iterative clustering algorithm. A co-cited pair is chosen at random and added to a new cluster. All remaining pairs that contain either document in the new cluster are added to the cluster, and this process is repeated until no more pairs can be added to the current cluster. A new pair is chosen from the remaining pairs to seed a new cluster and the entire process continues until all document pairs are members of a cluster.

The second method is based on a technique described in (White, 2003). A co-citation matrix is generated on the document set and a document similarity measure is defined as the Euclidean distance between two vectors. A complete-linkage clustering algorithm is then applied to the structure.

The authors suggest that clusters produced using these methods can be used to aid navigation and browsing as they "reduce the amount and complexity of the displayed information".

2.3.3 Summary of Clustering Methods in Hypertext

The Web and hypertexts in general are naturally complex and loosely structured. Links have varying semantic interpretations and can result in cycles. As a result, the hyperlink structure can be difficult to use efficiently in IR, and can pose problems for Web visualisation software. The application of clustering methods can simplify the Web structure by aggregating nodes, or by inducing a simpler tree structure that can be displayed more clearly to the user. Clustering methods use either textual similarity measures, hyperlink cohesiveness measures, or a combination of both, to determine document similarity. Some methods also incorporate document meta-data to categorise documents and usage statistics to determine the strength of connection between two documents according to the number of user traversals of a link between them.

2.3.4 Disadvantages of Clustering Methods for Context Definition

In general, document clustering methods are not tolerant of change in the contents of the document set. Addition and removal of documents requires a recalculation of the clusters in order to maintain their accuracy, as does any change in document contents, for those clustering method that depend on textual similarity, or addition or removal of links, for those that depend on link-based similarity. This reduces the applicability to the Web of clustering methods that are applied pre-query to the entire document set. Clustering methods that are applied post-query avoid these problems, as clusters can be created quickly from smaller, query-specific datasets. Post-query clustering, however, loses the time-saving advantage of having pre-calculated clusters. Furthermore, any restriction of the size of the dataset naturally reduces the amount of data available to the clustering method. In some cases this may result in less informative clusters, as a document that is not present in the query-specific dataset, but that may nevertheless belong in the same cluster as another document that is in the set (due to its structural or textual similarity, for example), will be absent. Applications that rely on pre-query clustering methods assume a certain amount of topic inertia, i.e. that most changes made to the graph will have no effect on most clusters, and only small effects on others.

Clustering based on textual similarity is also subject to equivalent problems as those faced by relevance propagation methods (described above) with respect to their narrow interpretation of document context. Text-based clustering is concerned only with documents that have similar textual content to each other. Documents with contrasting, but nevertheless related contextual information will not be clustered together using clustering methods based on textual similarity (Weiss et al., 1996; Aguiar, 2003; Gloor, 1991; Liu et al., 2002; Pirolli et al., 1996). Even if the clustering method incorporates a combination of textual similarity and link-similiarity methods to locate content that may be structurally close without being textually similar, textually similar content will be preferred to textually contrasting content under these systems.

Many link-based clustering approaches also employ methods that restrict their applicability to Web environments. Many employ in-link data, that are difficult to obtain on the Web, and often define a document similarity matrix according to hyperlink cohesion measurements (Weiss et al., 1996; Aguiar, 2003; Pitkow and Pirolli, 1997; Zhu et al., 2004). Such matrices contain similarity values for every document pair, requiring an internal representation of the Web graph, as with PageRank.

Other visualisation methods attempt to impose a tree-structure on portions of the Web, in order to improve visualisation, by hiding or displaying substructures as required, and browsing, by allowing a user to repeatedly choose from a small set of child clusters, as opposed to offering them a large number of clusters at once. These methods typically require the categorisation and removal of certain links according to the URL structure, or by other means. While this aids visualisation, it can restrict the 'free' nature of the Web. The hierarchical model is criticised by early pioneers of hypertext such as Nelson (1997), and Bush (1945):

"When data of any sort are placed in storage, they are filed alphabetically or numerically, and information is found (when it is) by tracing it down from sub-class to sub-class... The human mind does not work that way. It operates by association".[13]

2.4 Web Query Languages

A number of structured Web query languages have been proposed to allow retrieval systems to have access to document context as defined by the link structure, combining index querying and navigation (Konopnicki and Shmueli, 1995; Lakshmanan et al., 1996; Spertus, 1998; Lacroix et al., 1997; Mendelzon et al., 1996, 1997). In Mendelzon et al. (1996, 1997); Arocena et al. (1997) the Web is interpreted as a relational database made up of Document and Anchor (hyperlink) relations. The navigation is controlled by WebSQL, a SQL-like syntax that allows the user to provide a compound query of query terms and links, requiring the returned documents to match both.

e.g.

```
SELECT x.url, x.title, y.url, y.title
FROM  Document x SUCH THAT x MENTIONS "Computer Science",
Document y SUCH THAT x = |->|->.->y;
```

Represents a query: "Find all documents mentioning "Computer Science" and all documents that are linked to them through paths of length two or less containing only local links.". The W3QS language described in Konopnicki and Shmueli (1995) follows a similar approach, and has a similar syntax.

[13]Quote taken from Eiron and McCurley (2003)

In Lakshmanan et al. (1996), the query language also allows the inclusion of path information in the query as a form of 'partial knowledge' about the domain of the expected result set. Another example of partial knowledge is internal document structure and content.

In general, structured Web query languages attempt to capture document context explicitly in the query by specifying the types of relations relevant documents should have to their neighbouring documents. Queries written in these languages are often complicated and lengthy, and writing them requires a degree of prior knowledge about the result set, so they have not been widely adopted in general Web search engines. Applying a structured query language to Web data is also problematic as HTML is semi-structured at best (Abiteboul, 1997) - the semantics of HTML tags and links are loosely defined. The lack of rigid document schemata limits the effectiveness of structured languages, especially those that incorporate document structure information (Atzeni et al., 1997). Attempts have been made to move towards a more structured hyperlink system on the Web (Trigg, 1983) leading ultimately to Semantic Web (Berners-Lee, 1999), containing well-defined semantic tags on relationships between documents, allowing automatic information extraction. In the absence of such a system, Web-based query languages cannot expect too much of the user in terms of domain knowledge.

Dyreson (1997, 2002) describes an adaptation of path-based Web query languages that puts the focus on content, rather than structure, thus relieving some of the domain knowledge required when using such languages. Queries are made up of a series of query terms or phrases, with each part representing a specialisation of the document set obtained by carrying out a search on the previous part. Specialised document sets are found by following paths in the hyperlink graph from documents in the previous document set, and these paths are restricted by costs defined by the user. In a Mini-World-Web (a subsection of the Web concerned with a particular domain or topic), the author suggests a link-weighting system determined by semantics imposed on the links according to their type. Link types are defined by the URL directory structure of the source and destination documents:

Links from a document in a parent directory to one in a subdirectory, or from an index.html page, are down-links. Back-links are the opposite. Links between documents in unrelated directories are side-links. Links between documents in the same directory are side-links, if the source document cannot be reached from a document outside the directory, and down-links if they can.

The assumed semantics are that down-links lead from more general to more specialised pages. When the search engine is locating specialised information from an initial document set, it is restricted to following paths made up of a series of down-links followed by at most one side-link.

In Dyreson (2002), the cost-based system is removed, and instead, a directed acyclic 'concept graph' is induced from the Web structure that renders explicit the semantics assumed in the above method. Each document in the collection is a node in the concept graph, and edges between nodes represent a specialisation of the subject contained in the source node. These edges are generated by paths in the Web graph made up of a series of down-links, followed by at most one side-link, as above. Back-links are not incorporated in the graph, as they are seen as being the least likely indicator of content-specialisation.

This structure is then queried using a similar query language to the above method, but the user-specified costs are omitted, and queries are made up of purely a series of query phrases. An initial document set is found that matches the first query phrase, and then specialised document sets are located by following edges in the concept graph from documents in the initial set.

In these methods, the contents of source and destination pages are not taken into account when hyperlinks are classified into types, and when semantics are applied to these types. As a result, the content specialisation relationship between a destination document and a source document may be incorrect. The URL directory structure of Web documents may fuel this problem, if it is not used in the manner assumed in these works. There is also evidence to suggest that a number of highly-popular Web sites (e.g. Wikipedia) forgo URL directory structures entirely and allow hierarchies to develop implicitly and organically according to the content and user-specified links among documents. Assuming a particular semantic interpretation of a hyperlink between two documents without taking into account the contents of the two documents can be unreliable.

Mizuuchi and Tajima (1999) address a similar problem as Dyreson, that of capturing the contextual information supplied by documents in a path to a document. The authors treat this as an indexing problem, rather than a querying one, suggesting that document indices based on document content alone will be incomplete. In this work, these incomplete indices are complemented with keyphrases extracted from the document context, described as the likely path taken by a visitor to a document. Links are classified into types, similar to those identified in HyPursuit and by Dyreson, according to their URL directory structure. An 'entrance path' is then defined according to a number of stated assumptions based on those types.

Once the entrance path is identified, keywords are extracted from specific HTML tags (such as titles and headings) in the documents in the path, and added to the index of the target document.

The authors acknowledge that their method applies only to Web sites that follow their assumptions on the roles of links according to their link types, but believe that these assumptions describe a sufficient number of Web sites for the method to be valid.

2.5 Other approaches

Croft and Turtle (1989, 1993) describe two models for hypertext-based IR: spreading activation, and the inference model.

The inference model uses probabilistic inference to determine the probability that a document will satisfy a user's information need. The model is based on Bayesian inference nets, directed acyclic dependency graphs in which nodes represent proposition variables, and edges represent dependencies between propositions. Each node contains a 'link matrix', representing the probability of the proposition given all possible values of the proposition variable and those of its parent nodes (nodes pointing to it). In document retrieval, the network is made up of the combination of a document network and a query network.

The document network is made up of *Document* nodes, connected to *Representation* nodes, that correspond to representations of the information contained within the document. These representations may be an automatically-generated index, or automatically or manually-annotated keywords, for example. In this paper, the representation is an index of the document made up of the TF$\times$IDF weights of its terms.

The query network is made up of nodes representing a user's *Information Need*, connected to a number of *Concept* nodes, that represent the concepts that combine to form a given information need. At query time, the query network is connected to the document network by connecting Concept nodes to Representation nodes, and documents are scored according to the probability of each document satisfying the information need of the user; calculated by setting the probability of the document to 1 (activating it), and all others to 0, allowing these probability values to propagate through the hierarchy, and calculating the resultant probability of the node representing the information need.

In order to avoid cycles in the inference net, hyperlinks are interpreted as adding evidence to a model. The presence of a link (Croft and Turtle use citation links) from document d to document e adds evidence to document e, which increases the belief in any representation nodes connected to e when d is activated. The intersection of the document representations (in this work, the intersection will be terms in common between the two documents) are reinforced, and terms in e are effectively added to d, but with lower weights than those in d.

In the spreading activation model, hypertext documents are nodes in a directed graph, with edges made up of citation links between documents and a number of nearest neighbour links between documents with related content. Unlike other spreading activation methods (Ceglowski et al., 2003), document-term links are not included in order to control the connectivity of the graph. The top twenty ranked documents are obtained from a simple text-based document retrieval algorithm and their nodes are activated. The iterative procedure spreads activation levels around the graph. The activation level of a node at a particular iteration is increased by a function of the level of activation of nodes that point to it, the strength of the links. Nodes are not activated twice in the same path, to avoid unrealistic mutual reinforcement of activation levels caused by cycles in the graph. Once the activation levels being propagated through the graph fall below a certain threshold, the documents are ranked according to the levels of activation of their nodes.

The primary difference between the spreading activation method and the inference method is the inference method's adherence to the theory of probabilistic inference. However, this theory can pose problems, specifically in relation to the attachment of information need propositions to concept propositions, and the attachment of concept and document representation propositions. In addition, the probability calculations usually employ the independence assumption (described later in section 4.1.5) of document terms when calculating the representation or concept node link matrices. Given these difficulties, heuristic methods such as spreading activation often perform comparably to the inference method. In this paper, the inference method shows improvement over the spreading activation method, but the authors question its practical applicability nonetheless, as incorporating hyperlinks in this way requires the construction of a much larger index for each document.

The incorporation of content from hyperlinked documents as further evidence for a document is problematic in general, and Chakrabarti et al. (1998a) report that simply including terms from hyperlinked documents results can degrade accuracy in document classifiers due to the fact that link information is noisy. Salton also reported poor results when using neighbour text in a document classifier (Salton and Zhang, 1986), and Chakrabarti found that adding tags to neighbour terms to differentiate them from local terms also offers no improvement. Chakrabarti proposes an iterative algorithm that determines classes of unclassified documents according to the classes of their neighbours. This is a supervised categorisation method, as it requires the existence of a pre-classified document set. This set must be connected by hyperlinks to the unclassified documents, rendering it less suitable for general document processing applications on the Web, such as document retrieval and keyphrase extraction, and unsupervised classification.

2.6 Summary

This section gives an overview of the field of hypertext document processing, focusing on the use of hyperlinks to provide information about a document or a document set. At the beginning of the chapter, six factors were described, each of which is employed in a number of the above methodologies. Here these factors are described in more detail with explanations of their advantages and disadvantages.

(a) Nature of the Hypertext

Early work on hypertext document processing typically either used scientific document collections with citations as prototype hypertext collections (Kwok, 1975, 1984, 1985, 1988; Salton and Zhang, 1986), or induced hypertext documents by automatically generating artificial links between documents according to textual similarity (Salton, 1986), similarity of descriptions (Frei and Stieger, 1992) or relevance feedback (Savoy, 1994). Artificially-induced hyperlinks, especially those based on Nearest Neighbour techniques, are fundamentally different to typical Web links, as they are generated between documents with similar content, rather than documents with related content or connected content. Such links can be treated as special examples of methods that use content similarity measures described in (c). Academic document collections and pre-Web hypertexts also differ significantly to Web documents, as they are typically closed (i.e. the full collection is known at index time) and static (documents may be added to the index, but they do not change once they are indexed). As a result, collection-based methods described in (d) can be used, and it is easier to identify the hyperlink topology described in (b). Furthermore, authorship, register and authority of documents are typically more well-defined in such collections than on the Web, where documents are not as consistently scrutinised or edited, documents may have a variety of purposes (commercial, academic, social etc.), and links can exist for a number of different semantic reasons. Structured collections also tend to contain more meta-data or do so more consistently (see (f)).

(b) Link Topology:

A number of methods (e.g. PageRank, and measures calculated according to link vector similarity, e.g. Pitkow and Pirolli (1997)) require an internal large-scale representation of the Web or a large section of it, as well as cached document content. These methods include iterative link-weighting procedures, such as PageRank, and measures that weight document similarity according to link vector similarity or proximity (e.g. Hypursuit, Aguiar's context nodes, PageCluster). Other methods make use of document in-links. Since destination pages contain no indication of their in-links, it is impossible to obtain a complete list of the in-links of a page without trawling the entire Web for out-links to that page. According to the Web Characterization Project (Center, 2003), the Web is growing at a rate of between 4 and 6 pages per minute, rendering it impossible to obtain a snapshot of the entire Web at any time, or keep track of changes to an existing snapshot. Therefore, the known in-links of a page will always be a sample of the overall in-links, and that sample will be computationally expensive to obtain. This must be taken into account when using in-links in a definition of context. As described in section 2.1, Google uses in-links to calculate the PageRank of a page, as the presence of an in-link to a page can be seen as a vote from the source page for the destination page. In general, it is expected that in-links give a different type of contextual information to out-links (Marchiori, 1997). So this is an important design decision when developing a context-based Web application.

(c) Content Similarity:

Content similarity measures are often used to weight links between documents. They are usually defined by a cosine coefficient or Euclidean distance measure defined on document vectors in the VSM framework. Many of these measures use the TF$\times$IDF term weighting measure (see (d)). Content similarity measures are used in IR and clustering methods in order to locate textually similar documents.

The intuition behind the use of these measures is that similar documents provide extra information about the document. This information may be used by an application to describe the document in more detail or more accurately. For hyperlinks with certain semantic roles, this intuition is appropriate. For example, in an encyclopædia environment, a link with the anchor text "related articles" will point to a document about a similar subject, which may be sufficiently similar to the central document for it to merit inclusion in an IR application. Terms that occur frequently in both documents, relative to an overall distribution, may be considered to be key terms for that subject. Words that do not occur frequently in both documents are possibly of lesser importance with respect to that subject. Another, more contrived, example would be a compendium of reviews, summaries or descriptions, by different authors, on the same subject, such as a book, film, holiday destination, etc. Here, the intention of each document is the same - to provide the reader with information about the subject. Each document may focus on different aspects of the subject (e.g. main themes, character names), but one would expect there to be an overall convergence on the important aspects of the subject. Combining all the information in these linked summaries would therefore result in a filtering effect, dampening noisy information that is mentioned by few, and bolstering vital information that is mentioned by all. In this case, amalgamating content is a sensible use of the linked information.

However, in the majority of cases on the Web, hyperlinks do not connect documents with a similar semantic intent. Documents contain links to pages that focus on a single specific topic or generalise to a wider topic, or provide a specific function, such as 'FAQ', 'ordering' or 'email us' pages. Documents will be related in terms of domain, but the content of each will often be very different. All of the methods above will have a similar effect of dampening the differing information and accentuating the shared information. While it is likely that the information a document shares with its context set gives an important indicator of its own content, it is not clear why the differing information should necessarily be downgraded as a result. For example, a document on a university Web site entitled "Admissions" would exist within a hyperlinked structure along with other documents within the same Web site, that will contain the name of the university a number of times. The fact that these documents contain the name of the university is certainly an indicator that this is an important piece of information in the Admissions page. An amalgamation of the context set of the document would provide this information. Conversely, the word 'admissions' would be unlikely to occur within these documents, however it is understood that this is also an important piece of information for this page. An amalgamation of textually similar documents would necessarily decrease the importance of this word.

A variant on this technique is the 2-part index suggested by Aguiar, that splits a query into subject and context segments; the subject segment is used to query an index of documents and the context segment is used on an index of meta-documents, generated by a similar abstraction function to the above. This variation makes an important step by differentiating between document content and document context. However, the context is still an amalgamation of the contents of linked documents, as before. As mentioned in this chapter, this approach shows positive results, suggesting that there is some merit in separating content from context.

(d) Collection-based:

An important property of the Web is that it is practically unlimited in size; due to its rate of growth and its distributed nature, it is impossible to fully index the contents of all Web documents. The most famous collection-based term-weighting measure, TF$\times$IDF, normalises term frequencies according to the number of occurrences of the term in an overall collection. To obtain a useful weight using this value on the Web, either a large sample of Web content is required (random samples of Web content are difficult to obtain due to the lack of a centralised document index), or the domain of the collection that contains the term has to be well-defined. Localised search engines can define a collection as the set of documents contained on a particular Web site. However, as the Web changes as described in section 1.3, Web site structures are a less reliable source of homogeneous content; a single site may host information on a variety of subjects, and a subject-related collection of documents may be spread over a number of sites.

(e) Site Hierarchy:

Many methods use the URL directory structure as an indicator of hierarchical semantic relationships between documents. The directory structure of Web servers is not an explicit part of the Web, and no protocols or restrictions govern their use. As a result, this can sometimes give inaccurate information about these relationships as directory structures may be created for a number of different reasons, and authors may not be interested in, or aware of, such semantic roles.

As in (d), Web site structures are becoming more flexible as the Web develops, and the boundaries between information provider and information consumer are blurred. Users usually have no control over the directory structure of remote Web servers, so important user-driven Web sites such as Wikipedia have an entirely flat Web site structure, with implicit hierarchy developing from user-defined categories. Other sites, such as flickr, use tags to annotate content, resulting in a less hierarchical data structure more akin to the original concept of the 'memex' described by Bush (1945), and Berners-Lee's vision of the Web (operating "as close as possible to no rules at all" (Berners-Lee, 1999)). URL directory structure is therefore inconsistent, and may become more unreliable as user-driven content becomes more prevalent on the Web. If an application is designed to use this structure, its freedom is limited to Webs that implement the structure cleanly.

(f) Meta-data

The W3C HTML specification (Le Hors et al., 1999) defines the `<meta>` tag in the document head, and the REL attribute of anchor (`<a>`) or `<link>` tags that can be used to provide information about a page or the relationship between two pages. This information is used by the MAPA visualisation software (Durand and Kahn, 1998), and can be used to categorise links according to semantic roles, such as parent-child, index-element or sibling links. However, these META tags are rare, and are therefore an unreliable source of link information in Web documents.

The next chapter describes a method for the incorporation of contextual information that avoids the disadvantages associated with the factors described above. Authors that influenced my definition of document context are Dyreson and Mizuuchi (Dyreson, 1997, 2002; Mizuuchi and Tajima, 1999), who recognised that documents are created on the understanding that users reading them are aware of some contextual information contained in surrounding documents. In Mizuuchi, the Web is treated as a linear medium, with the context defined as the content of preceding pages in a path followed by the user, while Dyreson interprets the Web as a hierarchy; the subject matter of higher pages propagates to lower pages. As described in section 1.1, contextual information on the Web should not be treated linearly, and factor (e) above states that hierarchical URL structures may become less reliable in the future. Work on Relevance Propagation, particularly Sugiyama's adaptation of TF$\times$IDF, influenced the decision to adopt a localised strategy, where a context set is identified by spreading across hyperlinks from a starting document. However, my definition of context is strictly different to these methods; in RP, the emphasis is on content similarity, either between a document and a neighbouring document or between a document and a query. The definition of context used in this thesis applies both to documents with similar content, and contrasting content.

Chapter 3

Incorporating Contextual Information

The previous chapter provided a summary of the state of the art with respect to the usage of hyperlink information in Web-based information retrieval and natural language processing applications. In this chapter we present an alternative approach, in the form of a practical framework for the incorporation of contextual information in Web-based IR and NLP applications using Web hyperlinks. In chapter 2, we identified deficiencies in the state of the art that can be addressed by our method. The proposed method is inspired by term weighting schemes based on term frequency, which are already widely used in IR and elsewhere. We describe TF$\times$INF, a Web-based adaptation of the TF$\times$IDF term weighting scheme. The semantic intentions behind the TF$\times$INF measure are discussed, and contrasted to TF$\times$IDF. We conclude by suggesting applications that could benefit from the use of TF$\times$INF, which are evaluated in later chapters.

3.1 Introducing TF$\times$INF

3.1.1 TF$\times$IDF

Typical IR applications incorporate the concept of the frequency of a term within a document, normalised by the commonality of the word. In other words, if a term occurs often in a document, but is not very common in the overall corpus, then this document is 'about' that term to some extent. This is the reasoning behind the popular TF$\times$IDF term weighting measure.

TF$\times$IDF (Term frequency by inverse document frequency) is a term weight suggested by Salton and Buckley (1987) and widely used in a variety of IR and text data mining applications. It is made up of two components:

- TF - the number of occurrences of a term in a document, usually, but not necessarily, normalised by the size of the document. Intuitively, if a term occurs frequently in a document, it is more likely to be an important indicator of document content.

- IDF - proposed by Spärck Jones (1972), is based on the number of documents in a collection that contain the term. If the term appears infrequently in the overall collection, it is an important discriminator between those documents that contain the term and those that do not. IDF is usually calculated as the log of the ratio of the number of documents in the collection, with those documents that contain the term.

$$\begin{aligned} \mathrm{TF}\times\mathrm{IDF}(t,d) &= \frac{\mathrm{tf}_{t,d}}{\sum_{t'\in d}\mathrm{tf}_{t',d}} . \log(\frac{|D|}{\mathrm{df}_{t,d}}) \\ &= \frac{\mathrm{tf}_{t,d}}{\sum_{t'\in d}\mathrm{tf}_{t',d}} . -\log(\frac{\mathrm{df}_{t,d}}{|D|}) \end{aligned} \tag{3.1}$$

where:
t is the term of interest
d is the document of interest
tf is the number of occurrences of t in d
D is the collection
and df is the number of documents in D that contain t

In chapter 2, the example of a university Web site admissions page was given. The content-similarity approach to context would give 'admissions' a low weight in this document because it does not occur in linked documents. TF×IDF has the exact opposite effect. According to TF×IDF, terms that are unique to a document provide more information about a document than those that it shares with others. This is a term-document relation, and relationships between documents are extrapolated by identifying the important terms that documents share. If two documents share enough important terms then they are said to be 'related' in a text categorisation context.[1] This idea of document relatedness has been grafted onto the hyperlink structure of the Web; however, it ignores the fact that Web links provide information about differences between documents as well as indicating similarities. Indeed, it is the fact that the word 'admissions' occurs infrequently in the context set, and not in the overall collection of documents, that makes it important. In any case, an overall document collection is not well-defined on the Web, and a random sample of documents is difficult to obtain, and will inevitably be less focused on the subject domain than the context set.

[1]See section 5.1.2 for a description of the Vector Space Model of document similarity

3.1.2 TF×INF

We can therefore propose a variation of TF×IDF, in which the frequency of a term in a document is normalised not by the commonness of the word in the overall word distribution or corpus, but by the context set of the document. This variation is called TF×INF (where INF stands for Inverse Neighbour Frequency) and can be defined thus:

$$\mathrm{TF}\times\mathrm{INF}(t,d) = \frac{\mathrm{tf}_{t,d}}{\sum_{t'} \mathrm{tf}_{t',d}}. - \log(\frac{\mathrm{nf}_{t,d}}{|N|}) \tag{3.2}$$

where:

t is the term of interest

d is the document of interest

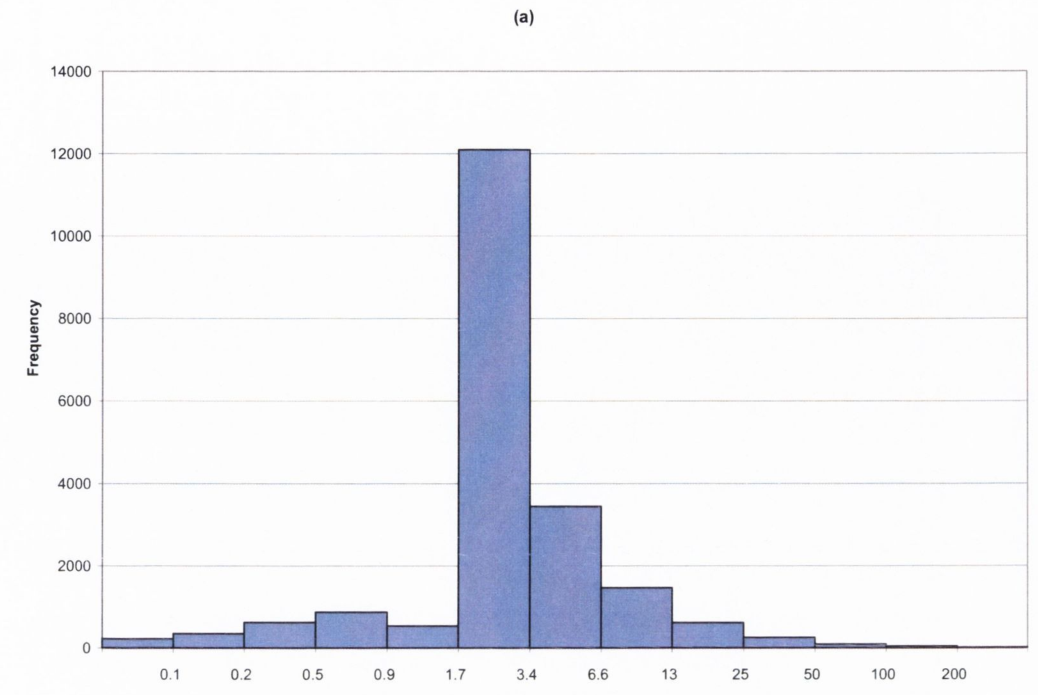
tf is the number of occurrences of t in d

N is the context set of d

and nf is the number of documents in N that contain t

Note that whereas TF×IDF captures merely the uniqueness of documents, TF×INF captures both the uniqueness of a document within its context, and the information it shares with its context. A low TF×INF value indicates a term that occurs frequently in the context set (such as 'Trinity' in the Trinity College Admissions page and is therefore similar to the information obtained by amalgamating the content of the context set, whereas a high TF×INF value indicates a term that is relatively unique to the document with respect to its neighbourhood, thus providing a focused version of the TF×IDF measure. Figure 3.1 gives a concrete example of this effect. The TF×INF values of phrases in a Web document collection taken from the Java Web site (see section 4.3) are plotted as a histogram with logarithmic bins in figure 3.1 (a). Chart *b* shows the histogram of keyphrases in the same corpus. The mean of the data in chart *b* is higher than that of chart *a*, indicating that terms with higher TF×INF values are more likely to be keyphrases. Two phrases that have high TF×INF values are indicated on the chart: phrase *iii* – 'real time' and phrase *iv* – 'ejb'. Additionally, the standard deviation of chart *b* is higher than that of chart *a*. There is therefore a region to the left of the chart where phrases are more likely to be keyphrases despite having low TF×INF values. Phrases *i* ('features') and *ii* ('xml') are examples of keyphrases of this type.

TF×INF therefore, is not a simple linear term weighting measure. Terms may be important if they have low or high values, whereas terms with middling values are neither unique enough nor ubiquitous enough to be important. Note also that some terms that occur frequently in all documents, such as function words, have no informative value. These terms are expected to be given very low TF×INF values, as they occur uniformly in all documents regardless of content.



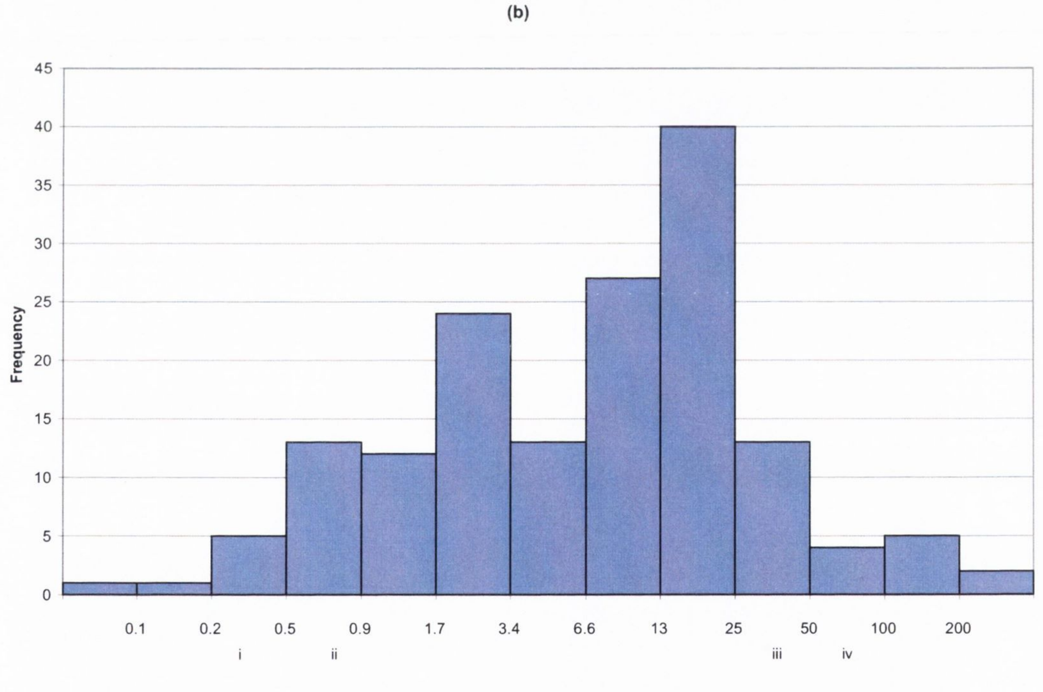


Figure 3.1: Histograms (with logarithmic bins) of TF×INF values of (a) all phrases and (b) keyphrases in a collection of Web documents taken from the *java.sun.com* Web site (see section 4.3). Normal distributions of these histograms have (a) log mean 0.4, log standard deviation 0.003, and (b) log mean 0.82, log standard deviation 0.053. Chart b is annotated with sample keyphrases: (i) 'features', (ii) 'xml', (iii) 'real time', (iv) 'ejb'

As a result, a low TF$\times$INF value does not always indicate an important term.[2]

3.2 TF$\times$INF in depth

3.2.1 Non-linearity of TF$\times$INF

TF$\times$IDF is a linear weighting measure - a higher value indicates greater importance. In contrast, the TF$\times$INF measure exhibits non-linear behaviour; as described above, Low and high TF$\times$INF values may indicate importance. Therefore, TF$\times$INF cannot be treated linearly by IR applications, and a more complex mechanism is required in order to use it to extract linear term weights. In later chapters we describe the use of machine learning techniques to estimate distributions of TF$\times$INF values of keyphrases and non-keyphrases. This distribution can then be used to derive a linear weighting scheme from TF$\times$INF values.

3.2.2 IDF in comparison to INF

The IDF value of a term is uniform across a corpus, whereas its INF value is relative to the context set in which it is defined. TF$\times$INF therefore can be treated as a version of TF$\times$IDF that is focused on a particular domain-specific section of the Web. As the context set increases in size to enclose more of the entire corpus, INF is expected to approach the IDF value. This is discussed in more detail in chapter 6.

3.2.3 Defining the Context Set

The way in which the context set of a document is defined plays an important part in the effectiveness of INF to represent a focused version of IDF. Other techniques (Weiss et al., 1996; Aguiar, 2003) use a Hyperlink Cohesion metric to identify relevant linked documents according to a number of factors including shared ancestors and descendents, and the number of independent paths between the documents. This approach is motivated by the wish to use hyperlinks to identify similar content, however, rather than contextual content. A more natural approach when identifying contextual content is simply to define the context set of a document as the set of documents directly connected to it by a path of hyperlinks of some length n. If n is 1, the context set of a document is the set of documents connected by one hyperlink to the document. Since most documents contain more than one hyperlink, increasing n results in a non-linear increase in the size of the context set. This has implications for the use of TF$\times$INF as increasing n will result in a non-linear increase in the computational cost of calculating the TF$\times$INF value of a term.

[2]Such terms are called stop-words, and are often removed from documents during a pre-processing stage. The stop-word list used in the applications described in this thesis is provided in Appendix B

This is a natural, but simple starting point for the identification of contextual information on the Web, and may result in unsuitable links being added to the context set, such as advertisements, or 'spamdexing' links (see chapter 2). The advantage of approaches such as the Hyperlink Cohesion metric is that it results in a context set that is more focused on a subject domain. Alternative approaches may use techniques such as link-weighting in order to choose certain links over others, resulting in a smaller, but ideally more focused, context set that accurately identifies contextual content as opposed to unrelated links.

3.2.4 Properties of TF×INF with respect to the factors described in chapter 2

In the previous chapter, six properties of existing work on the use of hyperlinks were identified. This section describes TF×INF in the context of these factors.

(a) Nature of the Hypertext

TF×INF assumes that hyperlinks are manually annotated, and direct the user to some form of related information. This makes it suitable for use on the Web, where these assumptions are met for the most part. It is expected that TF×INF can also be used on collections of academic documents, interpreting citations as hyperlinks. It is not designed to be used on hyperlinks generated automatically using NN methods, as the TF×INF values would be dependent on the automatic term-weighting method used to identify the nearest neighbours, rather than representing an implicit property of a document network.

(b) Link Topology

TF×INF requires a certain amount of links per document to be of value. The TF×INF values of documents with no links will be equal to simple TF, and the TF×INF values of fully connected Web graphs will be equal to TF×IDF.

The definition of TF×INF above does not identify the direction of links to be used in the definition of the context set. While both in-links and out-links represent contextual information according to the definition in chapter 1, in-links, as mentioned in chapter 2, are more expensive to obtain than out-links. In later sections we explore the possibility of removing in-links in order to increase the flexibility of the method.

Calculating TF×INF is not an iterative procedure, and requires no more information about the structure of the Web other than the vicinity of a linked document. This makes it inexpensive to implement, since it can be run locally as a search engine post-processor, for example.

(c) Content Similarity:

The main motivation for TF$\times$INF is to avoid interpreting the context set of a document as containing merely similar content, as described above. Interpreting the context set as providing complementary information in this way allows contextual information to be interpreted and used in a larger variety of ways. The following chapters describe a selection of interpretations of this information.

(d) Collection-based:

While the IDF component of TF$\times$IDF is defined on a document collection, INF requires only the neighbourhood of a document as defined by the hyperlink structure around the document. This makes it highly suitable for use on the Web as it can be used in situations in which a collection is either undefined or expensive to obtain.

(e) Site Hierarchy:

An important advantage of TF$\times$INF over a number of other context-based methods is that it does not use URL directory structure and overall site structure information. This increases its flexibility as it can be used on a variety of Web sites, including those with unconventional structures, or hierarchies induced from user-specified categories or other means, and across multiple Web sites. This is an important advantage as it allows for changes in Web publishing trends, such as those described in chapter 1.

(f) Meta-data

Document and hyperlink meta-data are rare on the Web, as mentioned previously. TF$\times$INF does not rely on the existence of meta-data.[3]

3.2.5 TF$\times$INF variation

TF$\times$INF is derived from TF$\times$IDF, and the formula for TF$\times$INF given above is therefore very similar in structure to TF$\times$IDF. In the following chapter, this formula is compared with a simpler variation, which is a simple, non-normalised ratio between term frequency (tf) and neighbour frequency (nf).

$$\frac{\mathrm{tf}}{\mathrm{nf}} \tag{3.3}$$

This version excludes the logarithm function, which may be more suitable in cases where the context set size (N) is small, as it increases the importance of nf in the formula. Furthermore, due

[3]Keywords found in HTML meta-data are used to train and evaluate classifiers in later sections, but TF$\times$INF itself does not rely on them

to the lack of normalisation, larger documents will tend to produce larger TF×INF values using this formula.

3.3 Summary

In chapter 2, a deficiency in the treatment of link information in the state of the art of Web IR and other document processing applications was identified. Namely, existing methods that use hyperlinks to identify context focus on content *shared* by a document and its context, rather than the unique terms that differ between them. To remedy this deficiency, we propose TF×INF, an adaptation of the TF×IDF term weighting measure. TF×IDF identifies terms as important if the ratio of frequency within a document and commonness in a corpus is high. TF×INF focuses the ratio on localised sections of the Web that are defined by the hyperlink structure. The intention behind these localised sections is that they are constrained by a particular subject or domain, since hyperlinks typically connect semantically related documents. The TF×INF values are therefore relative to this domain. As a result, unique terms are identified relative to the domain, and are given a high TF×INF value, and shared terms within the domain are given low TF×INF values. These domain-specific shared terms are informative within a document, as opposed to terms with low TF×IDF values which are shared across the overall corpus, and therefore do not provide information about the documents that contain them.

In later chapters, we apply TF×INF to IR and NLP applications in order to determine empirically if the information captured by TF×INF is sufficient to make a significant difference to the effectiveness of such applications.

Chapter 4

Keyphrase Extraction

This section describes the field of automatic keyphrase extraction and the application of TF$\times$INF to the field. The importance of document keyphrases and automatic discovery of keyphrases is discussed, and previous work on the subject is summarised.

In order to discover the uses of Web context in keyphrase extraction, the KEA automatic keyphrase extractor (Witten et al., 1999; Frank et al., 1999) was adapted to incorporate TF$\times$INF. This section describes KEA in detail, as well as the adjustments that were made. The datasets that were collected in order to test the application are then described. These datasets were used to test TF$\times$INF KEA, using a variety of parameters, against the standard version of KEA. It was found that TF$\times$INF significantly improves the number of good keyphrases reported by KEA.

4.1 Background

4.1.1 Keywords and Keyphrases

A keyword or keyphrase list is a list of words or phrases that provide a concise summary of the subject matter of a document. Keyphrases can be made up of multiple words or a single word, and so the more general term 'keyphrase' is used in this work to refer to both keyphrases and keywords. Keyphrase lists allow readers to quickly determine the utility of a document for their needs. They are also useful for concise indexing and categorisation of documents, as they indicate the type of search query that should reasonably be expected to return the document, and the topics under which a document may be classified in a collection. Keyphrase lists are typically different to indices, though, as indices usually contain a large number of less topical phrases (Turney, 2000). Therefore keyphrase lists are more concise than indices, and precision (see section 5.1.3) has more importance when compiling them.

4.1.2 Keyphrases on the Web

Some documents, especially academic articles, contain keyphrases specified by the author. However, the vast majority of documents do not contain author-annotated keyphrases; or any human-annotated keyphrases at all. This is particularly evident on the Web, in which there are no widespread regulations or requirements on content publishers to include them. The HTML Specification (Le Hors et al., 1999) has provision for manually-annotated keyphrases in Web documents, within the `meta keywords` tag, but browsers do not display them, and they are rarely present, and inconsistently used in most Web documents. Early Web search engines, such as AltaVista,[1] used the `meta keywords` tag to index documents; however, this made them vulnerable to manipulation by authors wishing to artificially promote their documents' rankings in the search engines' results by including a large number of popular search phrases that are irrelevant to the documents (Goodman, 2002). As a result, modern search engines such as Google ignore `meta keywords` tags, and the tags have subsequently fallen out of common use.

In spite of this, there is still an argument for the existence of keyphrases on the Web. Search engines usually provide summaries of the documents in their result lists, and these can be augmented with the use of keyphrase lists. Keyphrase lists can also be used to organise personal bookmarks and saved Web documents. Browsing may also be improved by allowing the user to see the keyphrases associated with a link before traversing the link.

4.1.3 Next Generation of Manual Keyphrase Annotation

The emergence of the Web as a social networking tool (O'Reilly, 2005) has seen the development of manually-annotated keyphrases generated independently of author influence. These keyphrases are selected, usually from an unrestricted vocabulary, by visitors to a Web resource and are stored on a public repository where they can be accessed by other visitors. This process is called 'collaborative tagging'. Such unrestricted user-generated keyphrase collections are often called Folksonomies (Mathes, 2004), and a popular example is del.icio.us.[2] Folksonomies avoid the pitfalls of author-specified keyphrases on the Web as the relevance of any given keyphrase is ranked according to its popularity (i.e. the number of users who have given the keyphrase as a tag for the document in question). Therefore the manipulation of tags for the purposes of artificial promotion of search rankings would require a concerted effort by a number of users. As folksonomies become more widely used (see figure 4.1), studies have been made into their stability and the degree to which they complement traditional keyphrase annotation and indexing techniques (Golder and Huberman, 2006; Kipp and Campbell, 2006). These reports suggest that while stable tag patterns are often reached after a document has been tagged a sufficient number of times, there are several disadvantages to the use of collaborative tagging as a reliable indexing or document summarisation

[1] www.altavista.com

[2] http://del.icio.us

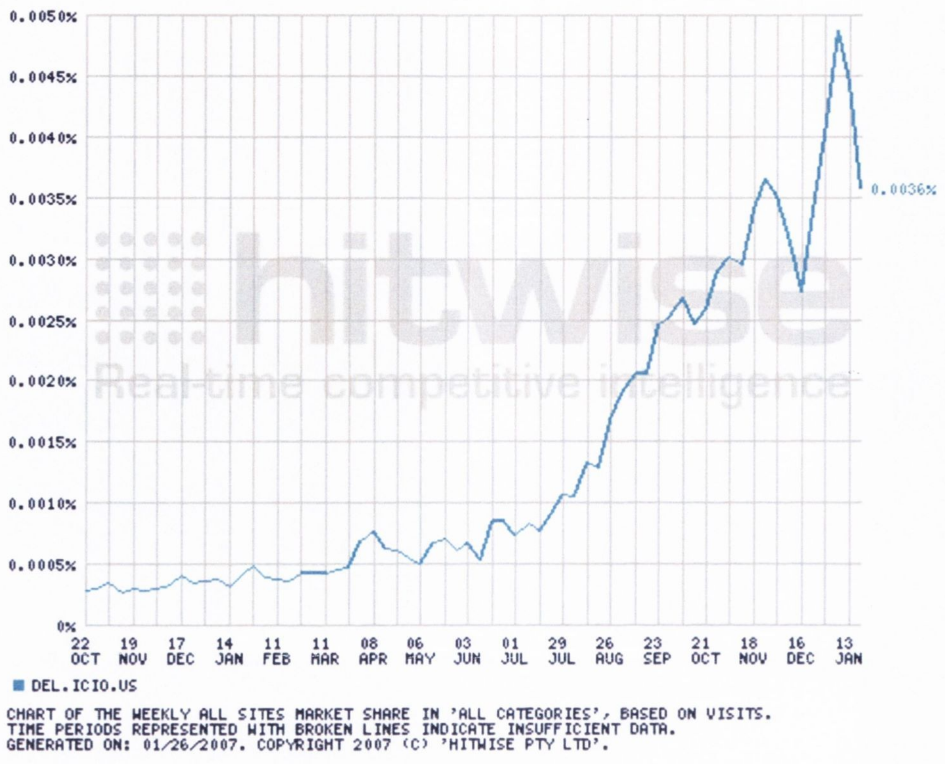


Figure 4.1: Del.icio.us traffic (taken from Rainie (2007))

tool, especially for documents that have been tagged by a small number of users. These disadvantages stem from the fact that users are not typically well-versed in information management and document classification. This, and the fact that most collaborative tagging systems use unrestricted vocabularies, result in problems with polysemy, synonymy and word-inflection (documents may contain the tags "cat" and "cats", for example), that weaken the value of keyphrases. Furthermore, folksonomy tags are personal - their informative value is relative to the tagger. This means that tags may be subjective, reflecting users' opinions on the content (such as the tag "funny"). Also, different taggers have different motives for tagging, from aggregation and easy relocation of tagged material to pragmatic tagging of material, such as adding the tag "to read" to a document. These motives, according to Golder, are typically much larger than any social interest, so the tags generated by such a system are not necessarily of use to the world at large.

Nonetheless, folksonomies are an interesting development in the Web, and as their use becomes more widespread, they may become valuable document summarisation and indexing tools.

4.1.4 Automatic Keyphrase Extraction

In the absence of manually-annotated keyphrases on the Web, we turn to automatic methods for the generation of keyphrase lists. Automatic Keyphrase Extraction is the process of marking phrases in a document as keyphrases according to some algorithm. In previous studies by Turney (2000) on document collections with author-annotated keyphrases, it was discovered that roughly 75% of these keyphrases occurred within the corresponding documents. A perfect automatic keyphrase extractor would therefore be able to provide 75% of the keyphrases that a human annotator could.

Automatic keyphrase extractors have been used in document summarisation (Jones et al., 2002; Zha, 2002), clustering (Hammouda et al., 2005) and query expansion (Song et al., 2006). Some automatic keyphrase extractors, especially those based on specific document collections or domains, have relied on explicit rules to identify keyphrases (Krulwich and Burkey, 1996). Domain-independent solutions typically use supervised machine learning algorithms to estimate a mapping function f from a domain of phrase features to the class set *keyphrase*, *non-keyphrase*. The algorithm estimates f from a training set with manually-annotated keyphrases, and can therefore be said to be supervised.

Turney (2000) evaluates two such algorithms, the C4.5 decision tree induction algorithm (Quinlan, 1993), and GenEx, based on a genetic algorithm.

C4.5 uses a set of nine term features, such as the normalised frequency of a stemmed phrase, the relative length of the phrase and the number of words in the phrase. A decision tree is built from the values of these features in all phrases in the training set and is then used to classify candidate keyphrases in the test set.

The keyphrase extractor in GenEx operates according to a set of parameters. These parameters are applied to instructions for the extractor, such as the amount of characters to truncate a phrase to. The optimum set of values for these parameters are calculated using a genetic algorithm that treats a set of parameter values as an individual in a population of potential parameter value permutations. New individuals are created by randomly mutating existing ones. Each parameter set generated from this algorithm is assigned a score according to its success when applied to the keyphrase extractor. More successful parameter sets are allowed to generate more offspring than less successful ones, usually resulting in an asymptotically more successful population.

Turney finds that GenEx performs better than the C4.5 decision tree induction algorithm and suggests that this is due to the domain-specific information (domain information here refers to the domain of keyphrase extraction) encoded in the parameterized instructions to the keyphrase extractor that is not present in the C4.5 algorithm.

4.1.5 KEA

Summary

KEA is a Java application that builds a Naïve Bayes model using training documents with annotated keyphrases and then uses the model to associate probabilities to candidate keyphrases in unseen documents. The most probable keyphrases are then chosen and reported by the program.

Candidate keyphrases are identified and prepared using a number of lexical rules described below. Two features, TF×IDF and *first occurrence* described below), are calculated for each of these phrases. When training the model, these feature values are stored alongside a class value *kp*,*non-kp*, according to whether the phrase occurs in the list of keyphrases for the training document, or not.

The extractor compares the feature values of a phrase to the discretised probability distributions of the features in the training set and computes an overall probability for the phrase.

Naïve Bayes and the Independence Assumption

The Naïve Bayes classifier is a probabilistic classifier that is based on supervised learning using Bayes' theorem. The naïveté of the Naïve Bayes classifier is due to the assumption that feature probabilities are conditionally independent of each other. In KEA, this allows us to simplify the conditional probability

$$p((t, f)|kp) \tag{4.1}$$

to

$$p(t|kp)p(f|kp) \tag{4.2}$$

t = TF×IDF value of phrase
f = *First Occurrence* value of phrase. These independent probabilities can be estimated from the training data.

The independence assumption avoids problems of data sparsity typically faced when calculating joint probabilities, and it has been shown that the independence assumption usually has little adverse effect on the accuracy of classification (Domingos and Pazzani, 1997), since binary classification uses thresholding to identify a class from a set of probability values. The probability values need not be accurate, therefore, as long as the correct class can be identified from them.

In addition, in KEA, Frank et al. (1999) explain that the TF×IDF and first occurrence features are close to being independent with respect to keyphrase extraction, further justifying the use of the independence assumption.

KEA Structure - Model Builder

Model-building in KEA is carried out by running it on a directory of training documents. The documents are in plain text form, with the suffix `.txt`. Each file is associated with a file of the same name with the suffix `.key` which contains the user-annotated keyphrases for the document.

To initialise the model builder, each document in the directory is read and stored in memory, along with its keyphrases. Once all documents have been read, they are sent to the phrase generator.

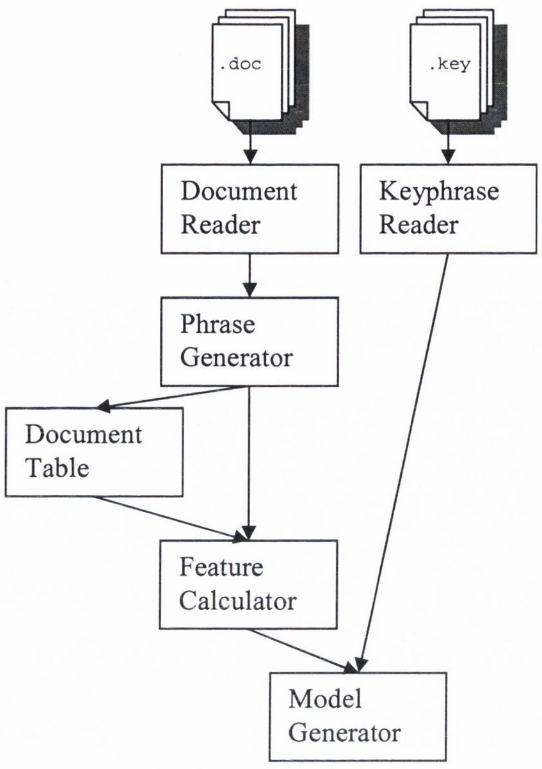


Figure 4.2: KEA Model Builder

Tokenisation and Phrase Generation

In both the model-building phase and the extraction phase, the first step when processing a document is to convert it into a set of phrases. KEA does this first by splitting the text into tokens - series of alphanumeric characters delimited by white space and punctuation. These tokens are then combined to form phrases according to the following rules:

- Phrases are made up of a number of tokens between some minimum (usually 1) and maximum (usually 3).
- Phrases are not proper nouns[3]
- Phrases do not begin or end with a stop-word. The stop-word list used here is provided with KEA and can be found in the appendix B.

Finally, the candidate phrases are processed by the Iterated Lovins Stemmer (Lovins, 1968). This attempts to remove suffixes from words in order to reduce inflected or derived versions of the same word to a common root.

[3]This requirement is the default setting for KEA, but was removed in our experiments, since a large number of manually-annotated keyphrases in the test datasets were proper nouns

Calculation of feature values

After the phrases have been generated, the following information about each one is stored:

- Stemmed form
- The most common unstemmed form
- Phrase frequency - the number of occurrences of all forms of the phrase in the document
- Position in the document of the first occurrence of the phrase

The stemmed form of each phrase is also added to a table of phrases for the overall training set. This table stores each phrase seen in the training set and the number of documents in which it occurs. This process is carried out for the entire training set before the program continues.

The following feature values for each phrase are then calculated from this information.

First Occurrence:

This feature is the relative position of the first occurrence of the phrase in the document, i.e. the number of words that precede it divided by the number of words in the document.

TF$\times$IDF:

The TF$\times$IDF value is calculated using the phrase frequency information along with the table of phrases for the training set as described in section 3.1.1. The formula here is slightly different, in that the document frequency and corpus size terms are incremented by one in order to avoid invalid values when document frequency is zero. This occurs when the term in a test document does not occur in the training set.

Both of these features are continuous variables.

Discretisation

The phrases in the training set are classified into two categories, keyphrase and non-keyphrase, and their feature values are used to estimate the probability density functions (PDF) for each feature in each class. Any suitable continuous distribution, such as the normal distribution, can be used to estimate the PDF. However, KEA discretises the features, using (Fayyad and Irani, 1993), after collecting the training data, to create a Probability Mass Function (PMF) (or histogram) for each feature in each class. Feature values from test documents are then converted to the range value into which they fall, the probability of that range is calculated from the PMF.

Model Generator

After the features for all candidate phrases have been generated and discretised and the PMFs have been estimated, the model, made up of the PMFs and the phrase table for the overall training set, is stored as a Java object for use by the keyphrase extractor.

KEA Structure - Extractor

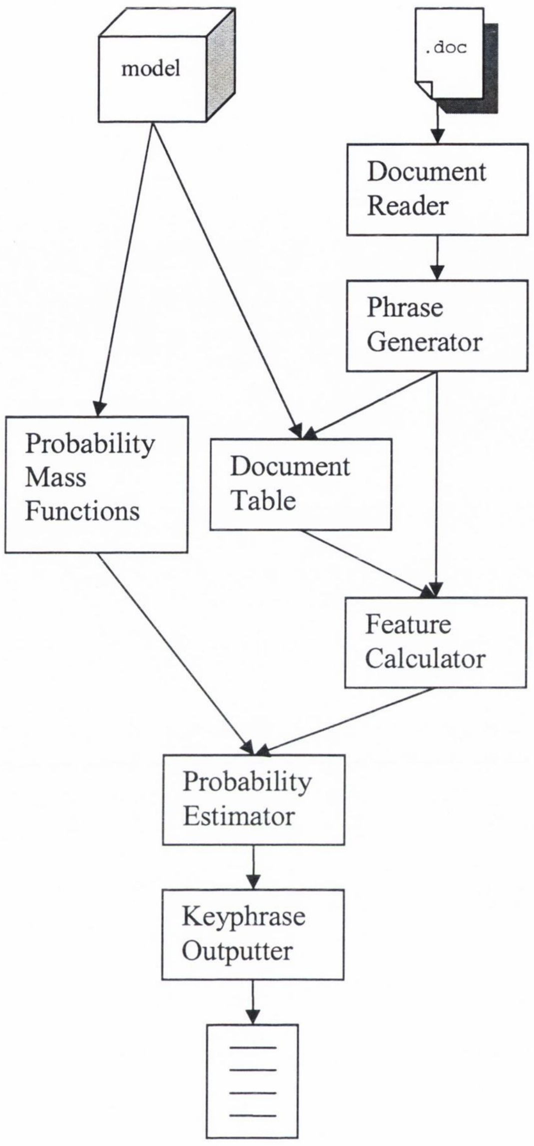


Figure 4.3: KEA Keyphrase Extractor structure

The KEA extractor processes documents individually, as opposed to in batches. A document is read and sent to the phrase generator and feature calculator. The feature calculator obtains the document frequencies from the model generated by the model builder. Once the features for all phrases have been calculated, they are sent to the probability estimator.

Probability Estimator

KEA uses a Naïve Bayes classifier to estimate the keyphrase probability of a candidate phrase given its feature values. The quantity to be calculated is therefore $p(k|(t,f))$, where $t = \text{TF}\times\text{IDF}$ and $f =$ First occurrence and $k =$ (class = keyphrase), and $\bar{k} =$ (class = non-keyphrase). Applying

Bayes' rule gives:

$$p(k|(t,f)) = \frac{p((t,f)|k)p(k)}{p(t,f)} \tag{4.3}$$

The Naïve Bayes classifier assumes feature independence with respect to the target class. Applying this assumption gives us

$$p(k|(t,f)) = \frac{p(t|k)p(f|k)p(k)}{p(t,f)} \tag{4.4}$$

Since k, $\bar{k}$ is a binary partition over all phrases, $p(t,f)$ can be rewritten as

$$p(t,f) = p((t,f)|k)p(k) + p((t,f)|\bar{k})p(\bar{k}) \tag{4.5}$$

Applying the independence assumption,

$$p(t,f) = p(t|k)p(f|k)p(k) + p(t|\bar{k})p(f|\bar{k})p(\bar{k}) \tag{4.6}$$

therefore

$$p(k|(t,f)) = \frac{p(t|k)p(f|k)p(k)}{p(t|k)p(f|k)p(k) + p(t|\bar{k})p(f|\bar{k})p(\bar{k})} \tag{4.7}$$

$p(t|k)$, $p(f|k)$, $p(t|\bar{k})$ and $p(f|\bar{k})$ are all given by the PMFs estimated from the training data, and $p(k)$ and $p(\bar{k})$ can be approximated using Maximum Likelihood Estimation as:

$$p(k) = \frac{\text{num keyphrases}}{\text{total phrases}}$$
$$p(\bar{k}) = \frac{\text{num non-keyphrases}}{\text{total phrases}}$$

Evaluation

KEA was found to perform comparably to GenEx, described above (Frank et al., 1999). As a result, it can be seen to represent the state of the art in domain-free automatic keyphrase extraction. For this reason, and the fact that it includes TF$\times$IDF as an integral component, it was deemed suitable for adaptation to TF$\times$INF.

A possible pitfall of KEA is that it does not tackle problems such as polysemy and synonymy, which can both weaken the semantic value of a phrase. Polysemy will result in inaccurately high term and document frequencies, as a polysemous term has multiple meanings that should by rights be treated as separate terms. Synonymy will also result in inaccurate term and document frequencies; in this case, they will be too low, as two synonymous terms could justifiably be combined into one term with combined frequencies. These problems are less common in domain-specific text collections than more general collections.

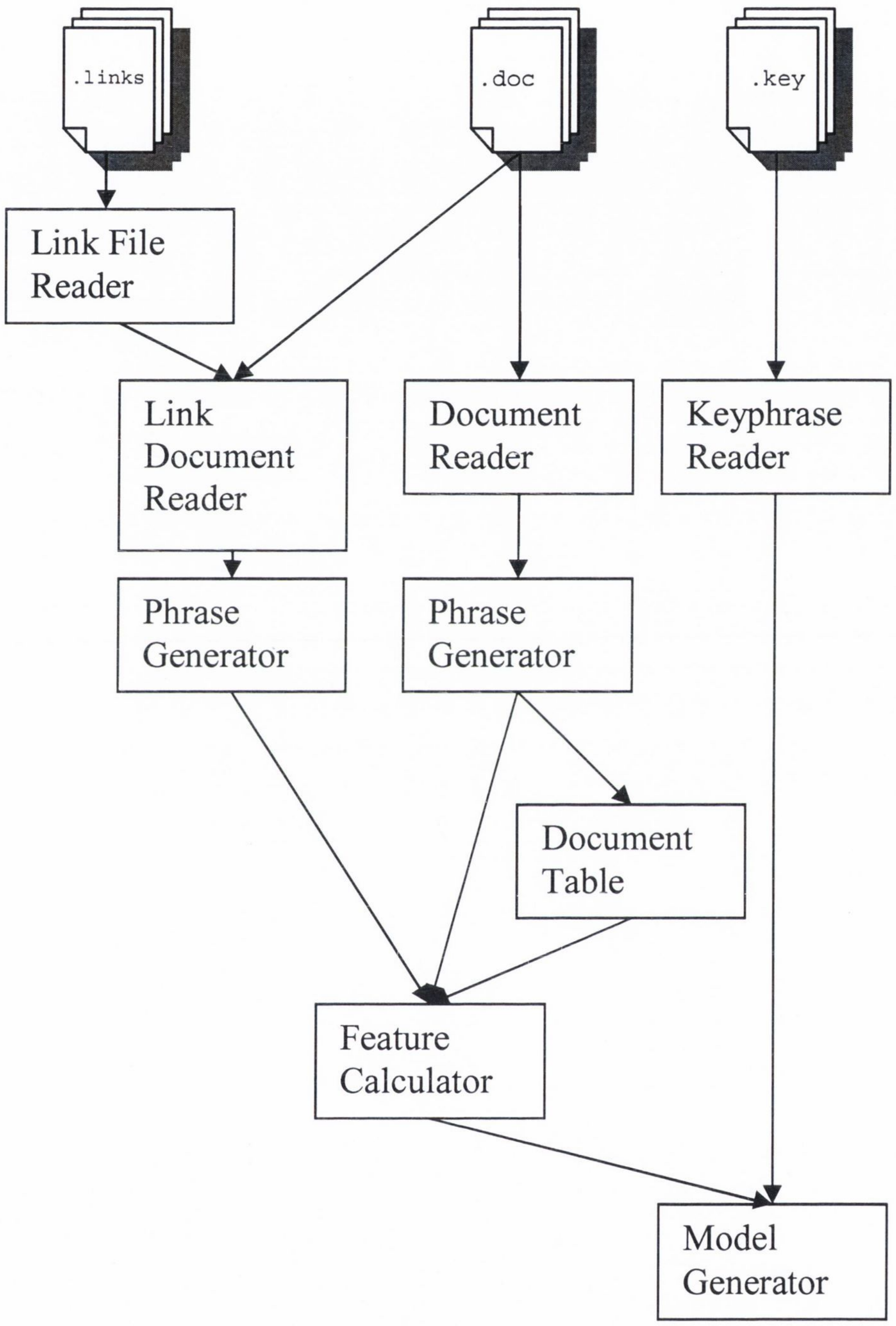


Figure 4.4: KEA Model Builder structure augmented with link information

4.2 Implementation

4.2.1 Adaptation of KEA to incorporate TF$\times$INF

The process of adapting KEA to include the TF$\times$INF term feature was two-fold. The first part of the process involved adapting the document processor to locate and tokenise text from neighbouring documents. The second part involved including the TF$\times$INF feature in the Bayes model described above.

4.2.2 Considerations when incorporating TF$\times$INF in KEA

When defining the neighbourhood of a document, two decisions must be made. The first decision is the type of links to be included in the Web graph, and the second is the distance to spread from the document when defining the context set. The available link type options are:

- In-links or out-links, or both

- Internal or external links, or both

In order to ensure a clear encapsulation of the training and test corpora, it was decided to exclude all external links, and any internal links to documents not contained in the corpora. For the purpose of simplification, initial experiments involved only out-links, and did not require the generation of in-link lists. Later experiments investigated the effects of including in-links.

4.2.3 Spreading Depth (n)

The degree to which an application should spread across the Web graph from the initial document in order to obtain the context of the document is an important decision. Under the assumption of relatedness of linked documents, the number of links travelled away from the document will determine the finesse or the specificity of the contextual information. As n increases, the risk that uninformative documents are included in the context of a document increases. Conversely, if n is too low, we risk missing potentially useful contextual information. Also, as n increases, the computational cost of locating and processing the contextual information increases. This increase is often non-linear, as documents tend to contain links to more than one document . There is therefore an ideal value of n that balances these requirements. Initially, n is set at 1, meaning that only directly-linked documents are incorporated in the definition of context for a document. This value was found to give a large amount of contextual information, on average, at a low computational overhead. Later experiments involved determining a more suitable value of n empirically.

Algorithm 4.1: A recursive algorithm for generating a context set

```
findNB:
    given:
        document $d$,
        depth $n$,

    do:
        read links file for $d$, giving list $l$ of linked documents
        remove from $l$ all elements already in the neighbour list
        store all elements in $l$ in the neighbour list
        if $n$ is greater than 1
            call findNB with $d =$ each element in $l$ and $n = n - 1$
    end
```

4.2.4 Location of linked information

As detailed above, KEA operates on a directory of text documents. In order to adapt KEA to process Web documents, we developed an application that converts a set of HTML documents to a set of text files by discarding all HTML tags and invisible content and extracting the visual textual content. This application was written in Perl and is described in detail in appendix B. All links contained in a document are extracted (from HTML anchor and link tags). These links are then processed and stored in a file with the same name as the text file and with the extension .links. The links are processed in order to remove external links and internal links to documents that are not in the corpus, to remove bookmarks (placeholders within a document) and to resolve relative URLs.

The resultant link files are used to locate the neighbourhood information that is used to calculate TF$\times$INF. For each document, a list of documents that form the neighbourhood of that document is generated using the recursive algorithm described in algorithm 4.1.

Once the neighbourhood list is generated for a document, the contents of each document in the list are read and stored in memory along with the document content and keyphrases. The phrase generator then extracts phrases both from the document text and the neighbourhood text and stores frequency information for each one. This information is then used by the feature calculator to calculate the TF$\times$INF values for the phrases.

The model builder then discretises these TF$\times$INF values as before and stores the resultant PMFs ($p(\text{TF}\times\text{INF}|k)$ and $p(\text{TF}\times\text{INF}|\bar{k})$) in the model object.

4.2.5 Probability Estimation with TF$\times$INF

The formula used by the modified keyphrase extractor to estimate the probability that a phrase is a keyphrase is similar to the formula in the standard version of KEA described above.

$$p(k|(t,f,n)) = \frac{p(t|k)p(f|k)p(n|k)p(k)}{p(t|k)p(f|k)p(n|k)p(k) + p(t|\bar{k})p(f|\bar{k})p(n|\bar{k})p(\bar{k})} \tag{4.8}$$

where $n = \text{TF}\times\text{INF}$

4.2.6 Implications for Independence Assumption

Including the TF$\times$INF feature in the Naïve Bayes classifier has an effect on the validity of the independence assumption, described above. While the assumption of independence between TF$\times$IDF and first occurrence is reasonable, it is less evident that TF$\times$INF and TF$\times$IDF are conditionally independent given the class k, $\bar{k}$, since the formulae for the two features are similar; Term Frequency is included in both, and INF, a measure of the document frequency of a localised sub-network of the Web graph, is intuitively likely to be related to IDF, a measure of the document frequency of the overall Web graph. Nonetheless, given the power and proven success of the Naïve Bayes classifier even in situations when the independence assumption is less reasonable (Hand and Yu, 2001), it was decided to retain the classifier for the TF$\times$INF case. An alternative to relaxing the independence assumption that is described later is simply to replace TF$\times$IDF in the probability formula with TF$\times$INF, rather than combining them.

4.3 Test Environment

4.3.1 Datasets

Datasets used in traditional IR experiments were unsuitable for testing the TF$\times$INF version of KEA, as it operates on a Web corpus. Formally, suitable datasets required:

- Textual content
- Links to other documents in the collection
- User-annotated keyphrases

Furthermore, in order to accurately study the effects of Web context on the keyphrase extraction, the links were required to be representative of typical links found on the Web; specifically, they had to be manually-specified, as opposed to automatically generated. The purpose of this project is to identify and use the implicit information provided by an author (or another user) when they create a link from one Web resource to another. Automatic link generation methods (e.g. Kurland and Lee (2006)) usually use some form of document similarity measure to identify possible links between documents. Any information obtained from such links would therefore be a function of the document similarity measure used, and would be more suitably exploited using such a document similarity measure directly rather than a hyperlink-based measure such as TF$\times$INF.

One of the purposes of this project is to determine if hypertext links offer a different kind of information to traditional relationships between documents based on document similarity. For this purpose, realistic Web-like hyperlink structures are required.

Hyperlinked Web corpora are quite rare, and those that exist often have a small amount of links between documents in the corpora. The most convenient way of ensuring a large number of inter-document links is to generate a corpus by starting from a single location on the Web and downloading every document up to a certain distance from the start location. For this project, a *Web spider* application was developed that performed this task, and is described in appendix C.

4.3.2 Keyphrases

As mentioned above (section 4.1), user-annotated keyphrases for Web documents are rare and difficult to obtain. This is due to the relative immaturity of the use of the Web as a linguistic corpus, and the unstructured and unregulated nature of Web publishing as opposed to traditional publishing. The growth of folksonomies provides a promising source of keyphrases for future validation of automatic keyphrase extraction.

In these experiments, it was decided that datasets should be obtained from Web sites with consistent `meta keyword` HTML tags. The advantages of using meta keyword tags are that they are included in the HTML source of the page to which they refer, so no extra Web crawling is required to obtain them (such as from a collaborative tagging site). Also, if it is the policy of a Web site to include meta keywords, they are included in most or all pages on the site. In contrast, the amount of information provided by a folksonomy is related to the popularity of a site - a folksonomy that contains tags for a Web site homepage or a few frequently visited pages on a Web site may not reliably contain tags for less popular connected pages in the site. We explained above that meta keywords have been misused in the past to artificially bolster search engine rankings. For this reason, it was important to select sites that used meta keyword HTML tags to accurately describe page contents. It was found that professional and official Web sites apply this rule adequately, and for this reason, the Web sites chosen to be the start point for corpus generation were:

- java.sun.com - The start page of the Java developer Web site
- www.oireachtas.ie - The Irish parliament home page
- www.number-10.gov.uk - The Web page of the office of the UK prime minister

It is assumed that the subject domains of these Web sites in themselves would have negligible influence on the behaviour of TF$\times$INF of their contents.

The collections generated from these Web sites and used in the proceeding experiments can be made available on request.

4.3.3 Dataset Statistics

4.3.4 Notes on Datasets

While the Web Spider started at the same location (http://java.sun.com/) for the collection of both the SmallJava and BigJava corpora, the depth to which the spider was permitted to spread

Table 4.1: Dataset statistics

	Document Count	Average document size (kb)	Average Num. Links	Average Num. Keyphrases	Vocabulary Size (total num. phrases)
SmallJava	253	11.23	14.07	11.88	83824
BigJava	1860	10.18	4.27	13.28	317539
Oireachtas	919	5.81	18.83	25.49	109494
No10	333	5.65	6.05	11.29	62068

was greater for BigJava. The SmallJava corpus contains mainly top-level documents such as front pages for Java technologies, press releases, download pages etc. The BigJava corpus, in contrast, contains a large number of highly technical components such as Java Language API pages, as well as the top-level documents contained within SmallJava. The contents of BigJava can therefore be considered to be more general than the contents of SmallJava.

The distribution of keyphrases in the corpora are skewed (see figure 5.10).

4.3.5 Training - Test Set Split

KEA is a machine-learning algorithm, and it requires a training set from which to generate the model used to classify the test examples. Typically, the training and test sets should be representative samples of the overall dataset that the classifier is expected to process, but the contents of the test set should not contain information also found in the training set, so that the error on the test set is a function of the generalization error, and we can assume that the results are unbiased by the contents of the training set.

4.3.6 Implications of the Unbiased Training-Test Split on the Web

Standard document sets can be easily split into training and test sets of any size, by assuming independence between documents. Web documents, however, contain implicit dependence between documents in the form of directed hyperlinks. Since we are assuming that these links contain important contextual data, it would be unwise to allow links in the test set to point to documents in the training set, as this would violate the assumption that the results are unbiased by the training set.

An iterative algorithm was designed (see appendix B) that would split a corpus into a training and test set according to the restriction that no documents in the test set contain links to documents in the training set. This was ensured by initialising the test set to contain documents with no links, and then iteratively adding documents that only contain links to documents already in the test set, until the test set reaches the desired size. This approach is called *Dependency Splitting* in the rest of this work.

While this approach avoids the problem of bias, it has its own problems, namely:

Table 4.2: Average Links per document in test and training sets generated by unbiased split. Note: Creating an unbiased split of the BigJava corpus required the deletion of too many hyperlinks, distorting the original information content of the corpus. Therefore, the values for this corpus are not included here.

	Training Set	Test Set
SmallJava	19.19	7.1
Oireachtas	19.69	17.97
No10	10.2	1.9

- Dependency Splitting favours documents with no or few links over documents with many links to be included in the test set (see Table 4.2).
- Furthermore, since the documents were collected using a breadth-first spread from a homepage, documents at a greater depth in the resultant Web graph are likely to have fewer links than documents higher in the graph. Using Dependency Splitting, the test set will contain a disproportionate amount of pages at a greater depth in the spread. These pages are potentially less representative of the overall dataset than a test set chosen independently of depth.
- Due to the presence of link cycles in the corpus, frequently as a result of home or back links, for example, certain corpora cannot be split in this way to produce a test set of sufficient size without removing links. Removing links results in less information for the classifier, and raises the question of which links to remove.
- Even if a sufficiently large test set can be produced, with or without removing links, the number of different test sets that can be created under Dependency Splitting is nonetheless restricted. This limits our ability to perform some forms of cross-validation.

In order to avoid these problems, an alternative dataset splitting algorithm was developed that ignores the dependence between hyperlinked documents and generates the test set by taking a random sample of the dataset. This approach is called *Random Splitting* in this work. A comparison of the results from both techniques (on the above datasets) is shown in the Results section in order to determine the effects of ignoring this potential bias.

4.4 Results

4.4.1 Corpus Specific Results

We carried out experiments on the test corpora using three different versions of KEA:

1 Original KEA with two term features: TF$\times$IDF and First Occurrence

2 KEA features + Basic TF$\times$INF term feature (see section 3.2.5)

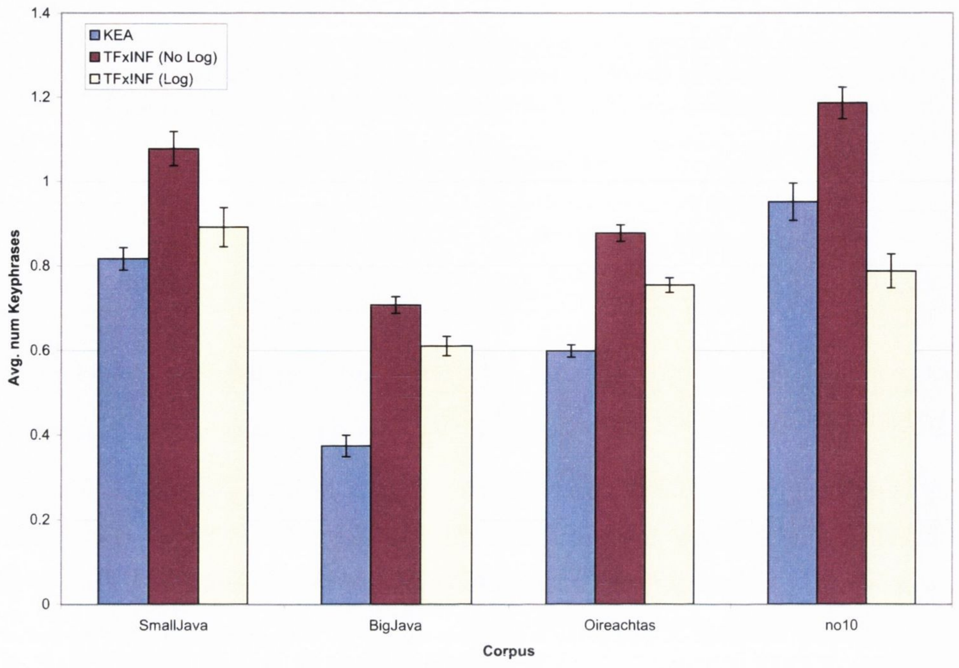


Figure 4.5: Results of KEA using different variants of TF×INF

3 KEA features + TF×INF (log form)

Each version of KEA was applied to each dataset, using Random Splitting to generate a 50-50 split into a test set and training set. This process was carried out twenty times for each dataset. The average results, with 95% confidence intervals,[4] are shown in figure 4.5. The average number of correctly identified keyphrases per document is shown on the vertical axis. KEA was instructed to report five keyphrases per document, so the maximum score is five. In the remainder of this chapter, KEA scores refer to the average number of correctly identified keyphrases, on the scale 1 to 5.

While overall success of all versions of KEA are relatively disappointing (on average, only one in five keyphrases reported by the program match the annotated keyphrases), figure 4.5 shows that TF×INF offers a statistically significant improvement over the original form of KEA. The log form of TF×INF is less successful than the other version, but still gives a significant improvement over the original KEA in three out of four cases. When applied to the No10 dataset, it decreases KEA performance. Figure 4.6 clarifies this information by showing the average improvement of each version of TF×INF KEA as a percentage of the performance of the original KEA. A value greater than zero suggests an improvement, while a value less than zero suggests a performance decrease. Figure 4.7 graphs the percentage improvement of TF×INF variations of KEA against the KEA benchmark score for each corpus. The negative correlation apparent in this graph shows that the degree of improvement over KEA is inversely correlated with the initial efficacy of KEA. This indicates that link information is at its most useful on corpora for which KEA performs badly.

[4]All confidence intervals shown in this work are 95% confidence intervals.

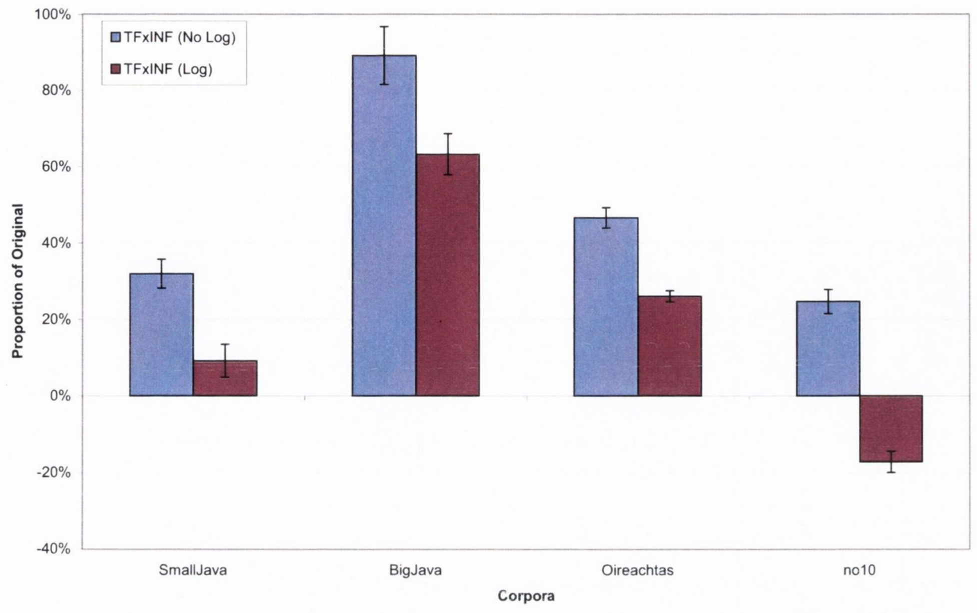


Figure 4.6: TF×INF % improvement over KEA

Table 4.3: Coefficient of linear relationship between KEA version scores and vocabulary size

Version	Coefficient	$2^{\text{Coefficient}}$
KEA	-0.57082	0.673234
TF×INF	-0.31185	0.805608
TF×INF (Log)	-0.19434	0.873973

4.4.2 Correlation with Vocabulary Size

Analysis of figure 4.5 against the data in table 4.1 suggests a weak inverse relationship between corpus size and KEA score - i.e. increasing the amount of training data counter-intuitively appears to reduce the success of the KEA algorithm. Conversely, Frank et al. (1999) report an asymptotic increase in KEA score up to a training set of roughly fifty documents, following which, the score remains steady as the training set size increases. In fact, this effect is more accurately portrayed by comparing KEA score and *vocabulary* size, where the vocabulary is the set of all distinct words used in the corpus. This inverse relationship is stronger than the relationship between corpus size and KEA score, and is roughly linear on the log-log scale, as shown in figure 4.8 (using logarithmic base 2). Table 4.3 gives the coefficients of these linear relationships calculated using linear regression with least mean squares. The inverse relationship between KEA score and vocabulary size is compatible with the intuition that it is more difficult to identify keywords in corpora with rich vocabularies than corpora with comparatively sparse ones. TF×INF variations of KEA have higher

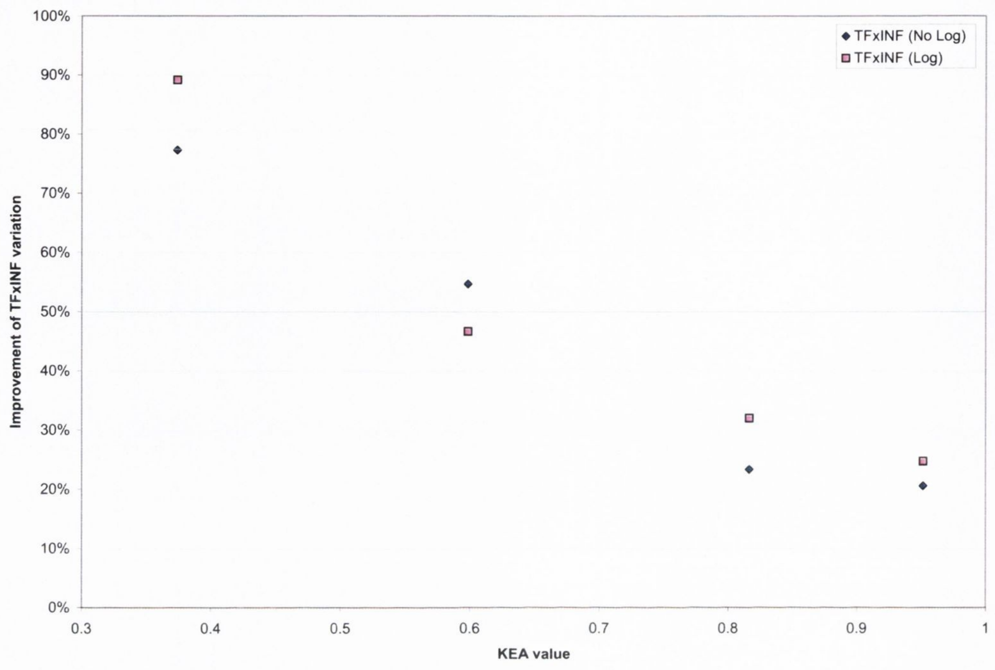


Figure 4.7: Scatterplot showing KEA scores against TF×INF improvement

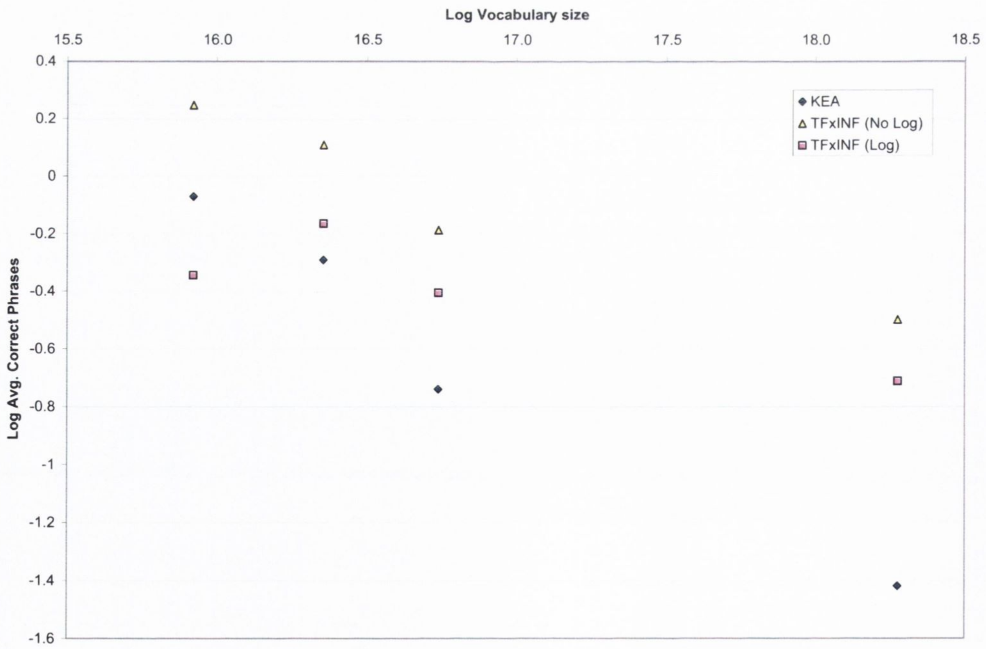


Figure 4.8: Log-log scale (base 2) chart of KEA version scores against vocabulary size

coefficient values, suggesting that Web contextual information may serve to dampen any negative effects of an increase in vocabulary size.

4.4.3 Sensitivity to Training Set Size

Given the suggestion above that TF×INF KEA is less susceptible to the negative effects of vocabulary size, it is interesting to explore the behaviour of TF×INF KEA at varying training set sizes, in comparison to standard KEA. Frank et al. (1999) report that the KEA score improves sharply from zero to 20 training documents, and continues to improve as the training set grows to 50 documents, and then remains constant. This result is roughly consistent with the results shown in figure 4.9, which graphs on the Y axis the scores obtained by evaluating KEA and TF×INF KEA on models generated from training sets with sizes graphed on the X axis. We can see that

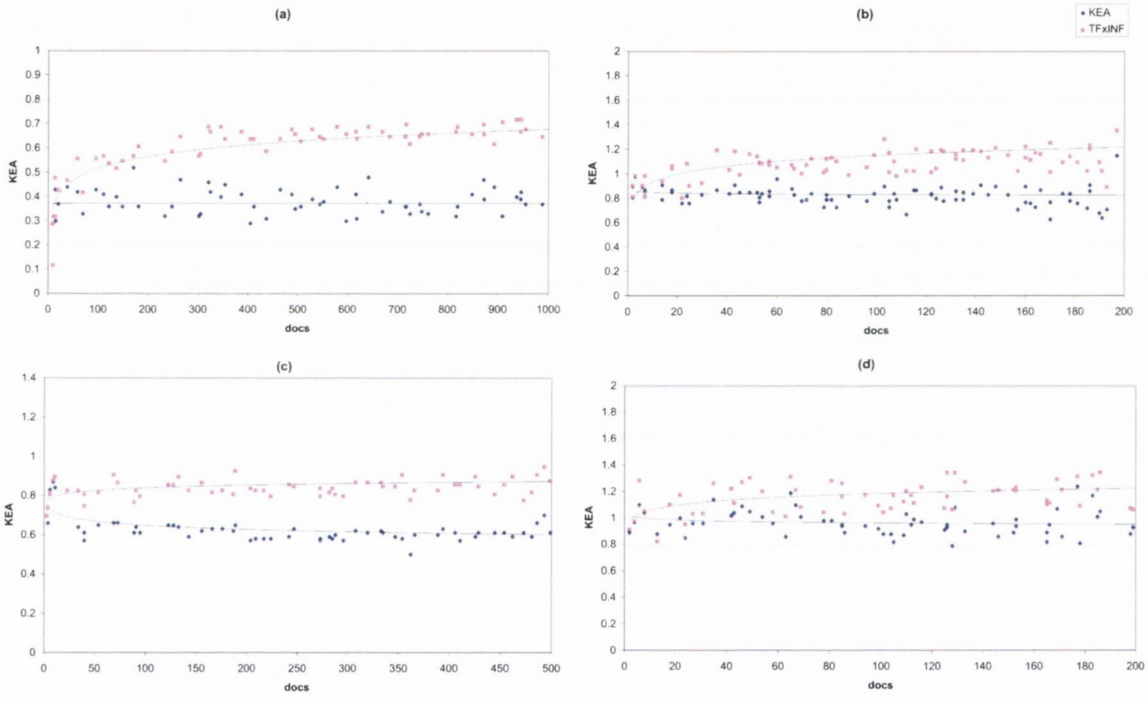


Figure 4.9: Comparison of KEA and TF×INF KEA for varying training set sizes on corpora a) BigJava, b) SmallJava, c) Oireachtas, d) No10

the improvement of standard KEA is, if evident at all, complete at roughly 50 documents (see blue points in (a) and (d)), and the score remains consistent or decreases slightly subsequently, as the training set grows. The TF×INF values also increase sharply between 0 and 50 documents, but also often continue to rise slightly as the training set increases (see pink points in (b) and (c)). This is consistent with the suggestion that TF×INF is less susceptible to vocabulary size increases.

In summary, TF×INF KEA reacts similarly to standard KEA with respect to training set size, as relatively few documents are required to generate a model that approaches the optimum level. However, it is less susceptible to an increase in vocabulary size.

4.4.4 Comparison of Set-Splitting Techniques

As described above, the manner in which a document corpus is split into a training and test set may have an impact on the score of TF×INF KEA. We described two different manners in which this split may be performed, Dependency Splitting and Random Splitting. Results up to this point have used the Random Splitting method, as it can be easily applied to all datasets, and avoids the problems described in section 4.3.5. Here we give the results of an experiment to compare these two methods of splitting the datasets in terms of their effect on the scores of KEA variations. If, after Random Splitting, the TF×INF KEA algorithm is using any links from the test set to the training set to unfairly boost its score, we should expect higher scores when using Random Splitting as opposed to Dependency Splitting for TF×INF KEA.

Ideally, since the standard version of KEA ignores link information, there should be no significant difference between the scores of the two methods for standard KEA. However, as mentioned above, a problem of Dependency Splitting is that it is less able to produce a simple random sample of the overall dataset. This could result in differences between the standard KEA scores that are unrelated to the problem of ensuring information from the training set is not contained in the test set. We can therefore use KEA as an indicator of the extent of this phenomenon in order to compare it to the results for the TF×INF KEA variations.

We tested each variation of KEA against three corpora, SmallJava, No10 and Oireachtas. Each corpus was split into test and training sets according to the two methods. The BigJava corpus could not be split according to the Dependency method without removing a large amount of hyperlinks, distorting the results of the TF×INF KEA variations. For this reason, it was excluded from this experiment.

Figure 4.10 shows the difference between scores of each respective KEA variation produced by Random Splitting and Dependency Splitting. A positive score indicates that Random Splitting resulted in higher score than Dependency Splitting, whereas a negative score indicates that Dependency Splitting had a higher score.

Confidence intervals are not visible in this figure, as the link structure of the corpora restricted the amount of different corpora that could be constructed using the Dependency Splitting method. As a result, multiple tests could not be carried out using this method, meaning that an average with a confidence interval could not be constructed for this experiment.

This figure shows that there is no consistent difference in scores when using one splitting method over another. In particular, the No10 corpus shows a positive score difference when using Random Splitting, Oireachtas gives a negative score difference, while SmallJava shows a minor positive difference. Importantly, these results are consistent across all variations of KEA, including the non-TF×INF, standard KEA. This means that the positive or negative difference between the two methods is likely to be unrelated to the problem of links in the test set to the training set, since standard KEA exhibits the same behaviour as the TF×INF variations. For this reason, and the

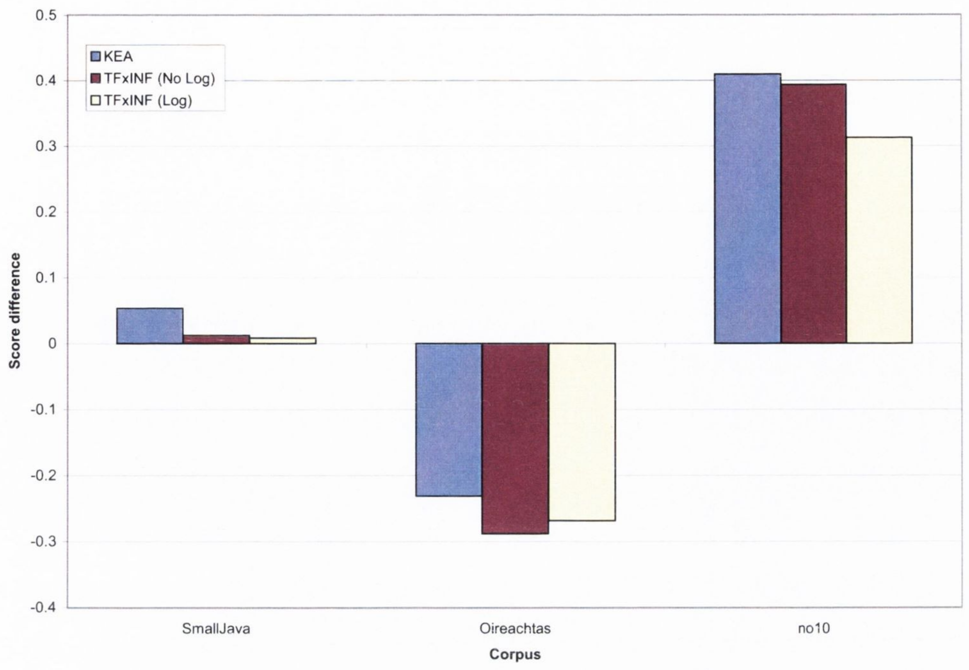


Figure 4.10: Comparison of KEA scores on datasets split by Random Splitting, as opposed to Dependency Splitting

fact that there is no consistent positive difference between using the Random Splitting method and the Dependency Splitting method, we can infer that any information about the training set contained within the test set after Random Splitting is of negligible value.

Since the argument for using Dependency Splitting has been shown empirically to be relatively weak, we argue that the relaxation of the independence between test and training sets has an insignificant effect on TF×INF KEA results, and allows us to perform more powerful tests without resorting to the removal of hyperlinks. We therefore favour the use of Random Splitting for subsequent tests.

4.4.5 Effect of Link Direction on KEA

As mentioned above, the choice of hyperlink type (in- or out-links), has an effect on the computational cost of including hyperlinks in the KEA algorithm and may also have an effect on the type or quality of information we obtain from using those links. In order to determine the extent of the latter effect, we developed a program that reverses the links in a Web corpus so that out-links become in-links, and another program that combined a document's in- and out- links to produce .links files that contain links in both directions, and carried out the tests above using these altered corpora. Note that the resultant lists of in-links are samples of the in-links obtained from the corpus. Comprehensive lists of in-links for the document on the Web are impossible to obtain, for the reasons given in chapter 2. Figure 4.11 shows the results of these experiments. This figure shows average values over all corpora, and the confidence intervals are wider as a result. There

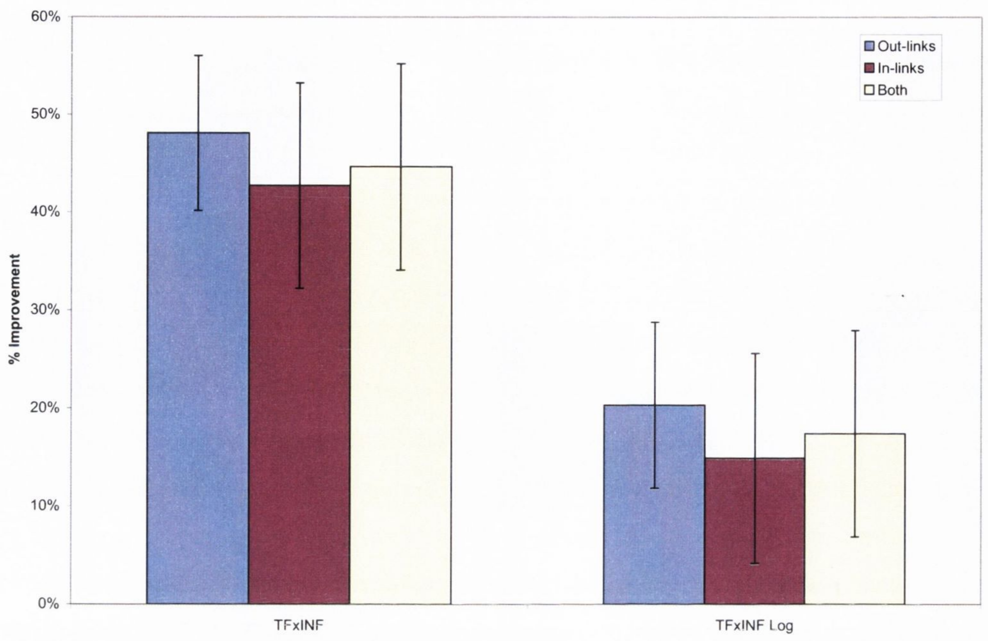


Figure 4.11: Proportions of TF$\times$INF improvements over KEA for different link direction types (proportions averaged over all corpora)

is no statistically significant distinction between the different link directions, evidenced by the overlapping confidence intervals. Importantly, including both in- and out- links does not improve the accuracy of KEA. Sufficient contextual information can be obtained from out-links. There is therefore no empirical reason to incur the computational cost of locating in-links for use in KEA.

4.4.6 Spreading Depth: n

Another decision to make when using TF$\times$INF is the value of n, the depth, in the Web graph, to traverse from the original document in order to locate its context. The semantic and computational implications of this decision are described in section 3.2.3. In the previous experiments, a value of $n = 1$ was used to identify the context set. Given the overall success of using the TF$\times$INF (non-log) feature and out-links in these experiments, tests using these parameters, but varying n, were carried out to investigate the value of widening the scope of the definition of Web context. Figure 4.12 shows the difference between the scores of TF$\times$INF KEA (portrayed as the percentage improvement over the original KEA score) by setting n to 1, 2 and 3. The scores are consistently better across all corpora when n is set to 1, i.e. restricting the size of the context set of a document d to those documents immediately surrounding d gives a greater improvement to a Web-based keyphrase extraction algorithm than extending the context set to all documents within two or three hyperlinks of d.

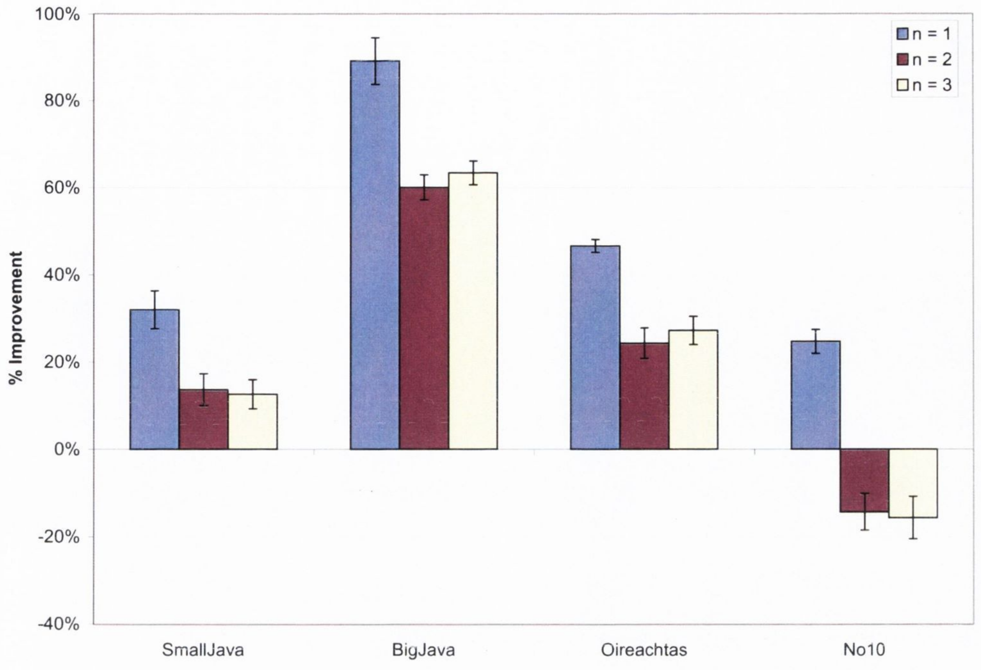


Figure 4.12: Effect of varying n on TF×INF KEA score

4.4.7 Independence of Features

As described in section 4.2.6 above, including TF×INF in a Bayesian classifier based on TF×IDF violates the independence assumption. A solution was offered that involved simply replacing TF×IDF with TF×INF rather than including both in the classifier. A version of TF×INF KEA was developed with the TF×IDF feature removed, and this was tested against the standard TF×INF version of KEA. Figure 12 shows the scores of these KEA versions as proportions of the original KEA version scores. In this test, only out-links were used to define the document context.

We can see that, despite the violation of the independence assumption, including TF×IDF in the Bayesian classifier typically results in significantly better scores than excluding it. Extensive literature supports the finding that violation of the independence assumption in a Naïve Bayes classifier often has negligible negative effects on the accuracy of the classifier (Hand and Yu, 2001; Zhang, 2004; Domingos and Pazzani, 1997). Including both TF×IDF and TF×INF has a considerable effect when applied to the BigJava data, suggesting that the TF×IDF and TF×INF values are more independent in this corpus than in the others, indicating that the information provided by contextual data in this corpus is significantly different to that provided by the overall collection data. The contextual data is highly important (as shown by the significant improvement of the KEA scores when including TF×INF on this corpus (see fig. 4.5), but does not capture the same information as TF×IDF. This could suggest that the corpus has *heterogeneous* content, described in more detail in chapter 6, as localised context sets have differing information to the overall 'general' set.

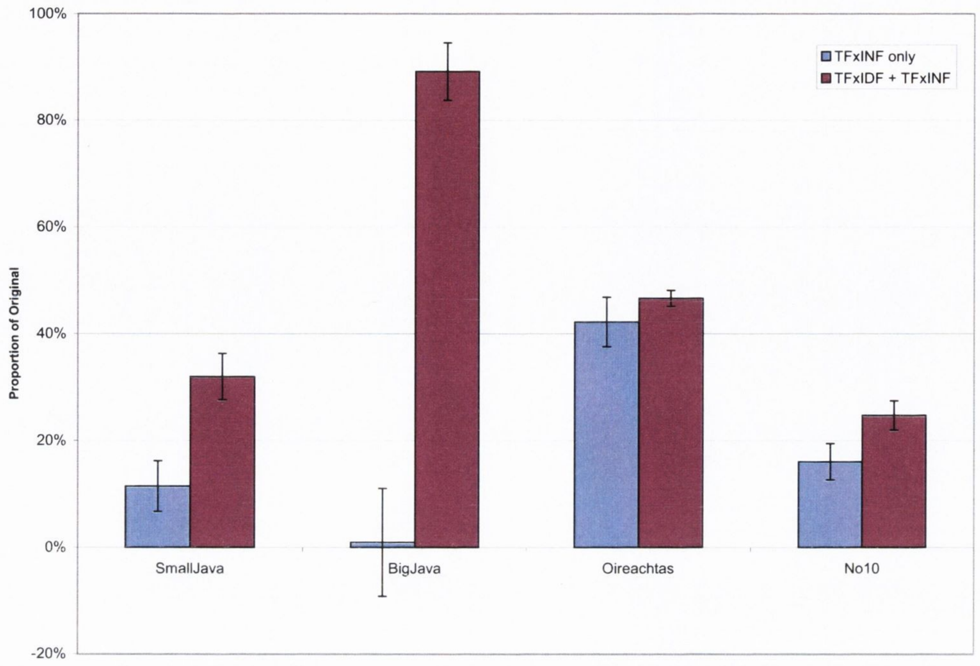


Figure 4.13: Comparison of TF×INF KEA with and without TF×IDF

4.4.8 Argument for the removal of TF×IDF

Despite the fact that retaining TF×IDF in a Web-based KEA algorithm gives improved results, an argument can be made for its exclusion in a practical keyphrase extractor. TF×INF-based keyphrase extraction without TF×IDF is nonetheless more successful than standard KEA, which uses TF×IDF. The disadvantage of the TF×IDF feature is that df, document frequency, is defined relative to the overall corpus. In a practical Web-based environment, the corpus is not well-defined, and df is therefore difficult to calculate. Conversely, nf (neighbour frequency) is calculated on a localised area of the Web graph, which is always defined, when using out-links as opposed to in-links.

Using a TF×IDF-free keyphrase extraction algorithm is therefore preferable when applying the algorithm to an undefined corpus such as the Web. When the corpus is known, including the TF×IDF feature will give better results.

As the path length used to calculate the context set increases, it encompasses a larger range of documents and topics. When the context set is sufficiently general, terms that are specific to a document with respect to the context set become more important, in comparison to terms that are shared by documents in a context set. Therefore, the TF×INF feature, at sufficiently high path lengths, may be a suitable approximation of the information provided by TF×IDF.

4.4.9 Modelling of Continuous Variables

By default, KEA uses a discretisation algorithm to convert continuous features such as TF$\times$IDF and TF$\times$INF into histograms in order to generate a PMF for each one. The reason for this is that a histogram is a robust, non-parametric model that can be applied to variables irrespective of their underlying distributions. In this section, we experiment with modelling the features as continuous variables. We use two models in this experiment:

1. The Normal distribution.

This distribution is commonly assumed when applying the Naïve Bayes classifier to continuous variables. The Probability Density Function (PDF) of the normal distribution is defined as:

$$g(x|\mu, s) = \frac{1}{\sqrt{2\pi\sigma}} \cdot e^{\frac{-(x-\mu)^2}{2\sigma^2)}} \tag{4.9}$$

where:
μ is the mean and
σ is the standard distribution of the attribute

This is a parametric function on μ and σ which are estimated via maximum likelihood estimation from the sample.

2. Kernel Density Estimation (KDE) (John and Langley, 1995) is generated by the normalised sum of a set of Normal kernels. Each kernel is calculated as:

$$g_i(x|\mu, \sigma) = g(x|\mu_i, \sigma_i) \tag{4.10}$$

where:
i ranges over the training values of the feature,
$\mu_i = x_i$: the value of the feature for training example i
$\sigma_i = \frac{1}{\sqrt{n}}$
n is the number of training instances of the class

KDE is therefore made up of a set of normal distributions centred on each training data point. As a result, it acts as a smoothed histogram, which may fit the data better than a Normal distribution, while allowing a greater degree of generalisation and avoiding the over-fitting of a discrete distribution.

4.4.10 Computation Cost and Storage Space - Comparison

The histogram method for the representation of feature values requires the storage of feature value counts for each of the bins of the histogram. The storage space is therefore constant with respect

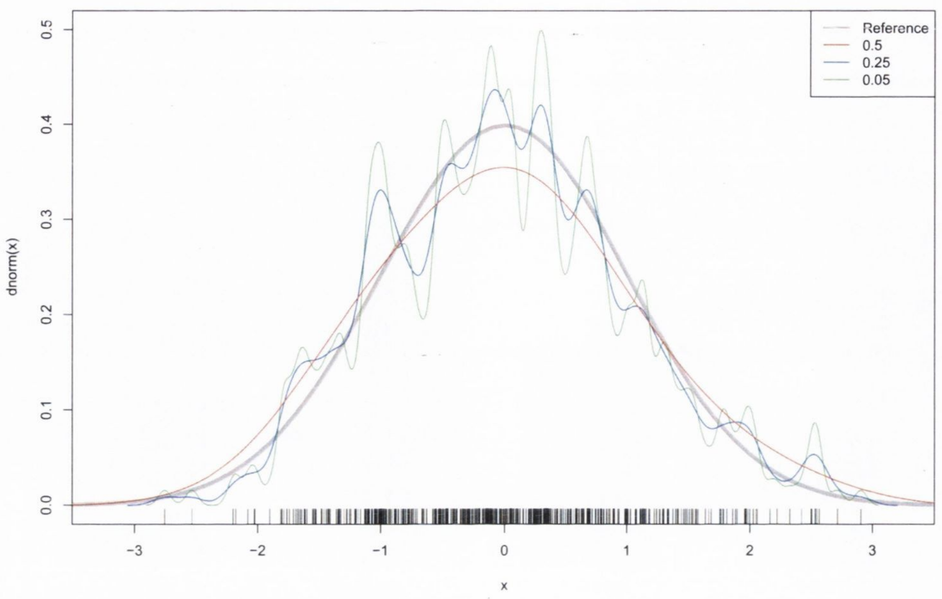


Figure 4.14: Demonstration of KDE approximation of 500 normally-distributed random numbers using different values for σ_i

Table 4.4: Comparison of complexity for the Naïve Bayes Classifier using discretisation, the Normal distribution, or KDE to model continuous feature values, given n training examples

	Discrete	Normal	KDE
Storage	$O(1)$	$O(1)(O(n)$if updateable)	$O(n)$
Computation time	$O(1)$	$O(1)$	$O(n)$

to the amount of training data, assuming the bins of the histogram do not change as training data is added. Computation time for the calculation of probabilities of unseen values is also constant, as it requires a simple lookup of the relevant histogram bin.

The Normal distribution requires only the storage of the mean and standard distribution of the training values. This is constant with respect to the training set size. However, data must be stored before the model is generated, and if the model is required to be updateable, each training value must be retained so that updated mean and standard deviation values can be calculated. Computation time for the calculation of probabilities is constant, as it amounts to the calculation of the function of μ and σ.

KDE calculates a sum across each of the training values, so the training values need to be stored in the model, whether it is to be updated or not. The storage space required is therefore linear with respect to the training set size. Likewise the computation cost of calculating a probability is linear (see table 4.4).

Experiment

In this experiment, TF×INF KEA was adapted to calculate the probabilities of unseen feature values for each class using either the Normal PDF or the KDE PDF, as opposed to the PMF calculated by discretising the features. In accordance with previous results, TF×IDF was included, out-links only were used, and the spreading depth, n, was set to 1. Figure 4.15 shows the results

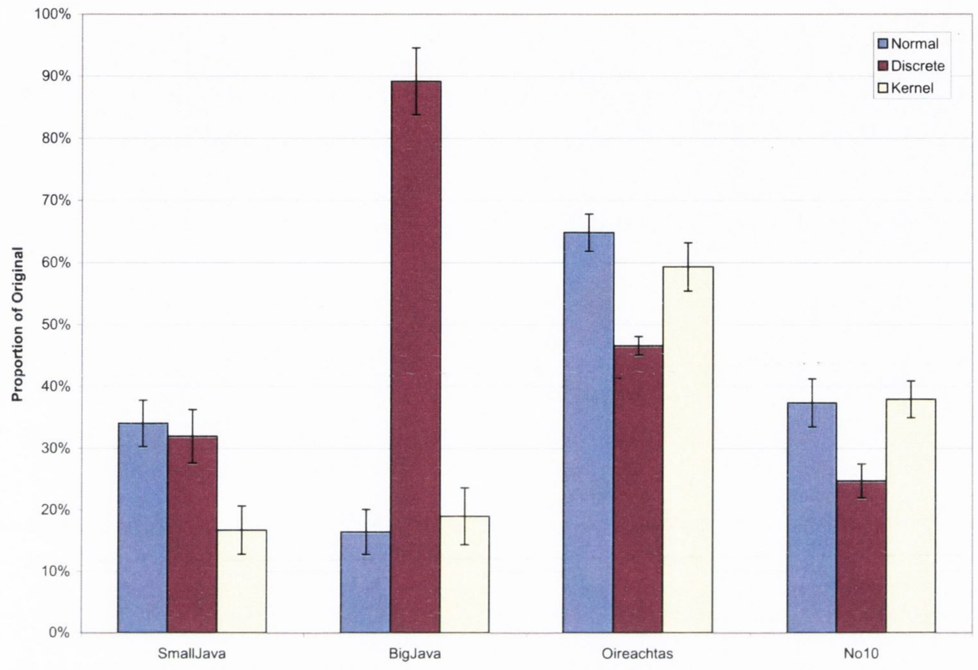


Figure 4.15: Comparison of TF×INF KEA using continuous and discrete feature modelling functions

of this experiment, with scores represented as a percentage improvement over the original KEA score (discrete, without TF×INF). The results of this experiment are inconclusive; the continuous distributions offer a significant improvement over the discrete distribution in the Oireachtas and No10 corpora, whereas in the SmallJava corpus, the Normal distribution is statistically comparable to the discrete distribution, and the KDE method is significantly worse. In the BigJava corpus, both continuous methods are significantly worse than the discrete method. This may be a result of the size of the BigJava corpus, as discrete distributions are often more accurate given the existence of a larger amount of training data. Unsmoothed PMFs suffer from sparse data more than smooth continuous functions, since a histogram category with no data will be incorrectly given a probability of zero. This danger is lessened with larger amounts of training data.

As a result of the inconclusive nature of the results, it is difficult to make recommendations on the use of continuous distributions in KEA. However, given the potential problems of data sparseness when using small training sizes, it may be preferable to use continuous distributions to model the data if this is the case. Concerning the choice between using the Normal distribution or KDE, this result offers no reason to choose KDE over the Normal distribution, and since KDE requires a greater amount of computation time and storage space, it is therefore preferable to use the Normal distribution in KEA.

4.4.11 Domain-dependence of KEA

In the experiments reported previously, the training sets and test sets are generated from the same initial dataset, so the domain of the training set and test set is the same for each experiment. If the success of the Web KEA algorithm is shown to be largely irrespective of the domain of the training data, this would have a positive effect on the applicability of the algorithm in a general context. In particular, the algorithm could be successfully applied to a domain for which suitable training data (i.e. documents with annotated keyphrases) were not available. Furthermore, even if such training data existed, it would be preferable to be able to use pre-calculated models rather than generate new models for each domain.

For these reasons, we developed an experiment to investigate the domain-dependence of the TF×INF KEA algorithm. Witten et al. (1999) report that KEA is domain-independent without the additional feature of 'keyphrase frequency' reported in Frank et al. (1999). Here we investigate whether this domain-independence is maintained when we include Web information.

In order to determine the effects of domain information in TF×INF KEA, each corpus was tested using four different probability models: one 'domain model', based on training data obtained from the same corpus as the test set, and three 'non-domain models', based on training data from the other three corpora. These are then evaluated on the test set and the difference between the scores is recorded.

Figure 4.16 shows the results of these tests, categorised according to the corpus used for the test set. In each test, the results when using the domain model are displayed on the left, followed by the non-domain models for each training set. The effectiveness of the standard KEA is shown to be unrelated to the training set domain. In only one of the four cases is the domain model the most successful, and this is when using the No10 corpus (figure 4.16(d)), a corpus with a low vocabulary count which also provides the most successful non-domain models in cases (a) and (b). This shows that the standard KEA is indeed domain-independent, as maintained in Frank et al. (1999).

The results for TF×INF KEA, however, show that the domain model is either the most effective training model (cases (b) and (d)), statistically indistinguishable from the best training model (case (a)) or close to the effectiveness of the best model (case (c)). This indicates that TF×INF KEA is more susceptible to the domain of the training set, and that a domain-specific training set should be used wherever possible. However, the non-domain-specific TF×INF KEA still outperforms the non-domain-specific standard KEA, on average (statistically significant with 95% confidence), indicating that TF×INF KEA can still be used on Web collections with no domain-specific training data, and will be expected to outperform standard KEA on most occasions. This is a promising

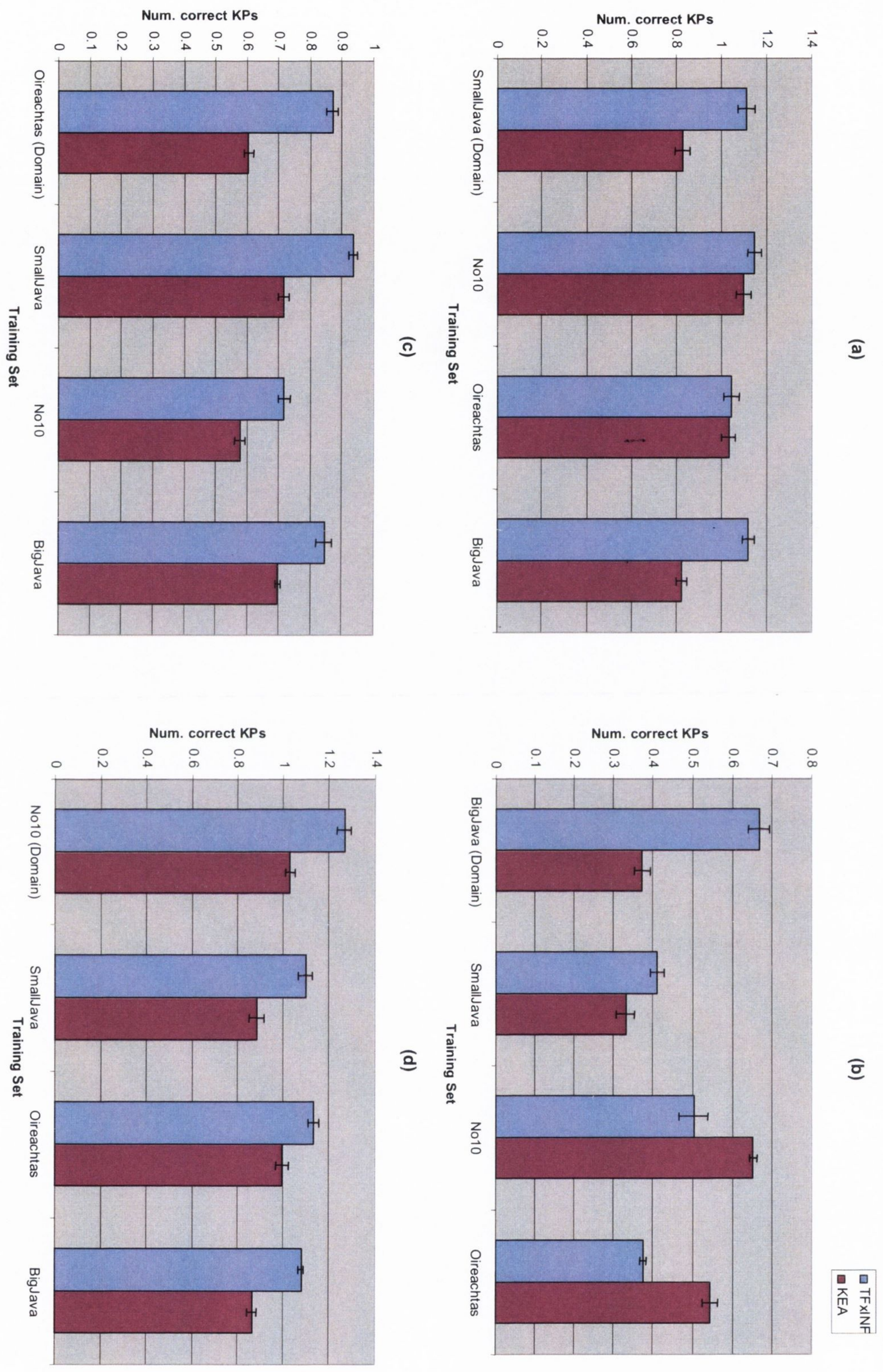


Figure 4.16: Domain dependency results using test sets from (a) SmallJava, (b) BigJava, (c) Oireachtas and (d) No10 corpora

result as it widens the practical applicability of TF×INF to corpora with no available training data.

4.5 Discussion

4.5.1 Summary of Results

We presented a Web version of the KEA Automatic Keyphrase Extraction algorithm, adapted to incorporate hyperlink information in the form of the TF×INF term feature. This feature was added to the Naïve Bayes classifier used by KEA, and the program was evaluated on corpora of documents harvested from various sites on the Web with annotated keyphrases provided by the HTML meta keywords tag. The program was evaluated by training the classifier on a subset of the corpus and applying the resultant classifier to unseen documents. The program was instructed to report five keyphrases per document and the score of the program was set to be the average number of reported keyphrases that matched those found in the meta tags.

The following results were observed:

- Including the TF×INF feature was found to improve the KEA score on Web corpora by between 20 and 90 percent.

- The score of standard KEA is negatively affected by the size of the training set vocabulary. TF×INF KEA is less negatively affected by this.

- Link direction (in- or out-links) has little effect on the score of TF×INF KEA. However, including in-links does not improve TF×INF KEA scores. Therefore, TF×INF KEA can be restricted to use only out-links, that are easier to obtain on the Web, without a loss in performance.

- The ideal spreading depth when identifying a document's neighbourhood or context is one hyperlink.

- Despite violating the assumption of independence of features, including TF×IDF with TF×INF in TF×INF KEA results in significantly better KEA scores. However, in a Web environment, TF×IDF feature values can be costly to calculate. It can therefore be excluded in practical situations, as the TF×INF version still outperforms the original version with TF×IDF excluded.

- In certain circumstances, modelling the continuous feature values used by the Naïve Bayes classifier with the Normal distribution will provide more accurate probability estimations than using a discretisation function. There is some evidence to suggest that a discretisation function is more suitable if there is sufficient training data to avoid data sparseness.

- Domain-specific training data is more useful to TF×INF KEA than it is to standard KEA. However, TF×INF KEA will still outperform standard KEA in the majority of cases when domain-specific training data is not available.

Based on these results, we can make the following recommendations regarding the use of KEA in a practical Web environment

- The following term features should be used in the KEA classifier by default: *first occurrence*, TF×INF.
- If the set of documents to which the algorithm will be applied is well-defined, include TF×IDF as a feature.
- Context sets should be defined by following out-links only to a spreading depth of n=1.
- If domain-specific training data is available, use it to develop the Naïve Bayes model.

4.5.2 Implications for Automatic Keyphrase Extraction

Web Applications

This chapter shows that link information can significantly improve the quality of automatically generated keyphrases for Web documents. Due to its relatively low dependence on domain-specific training data, this application can be incorporated immediately into existing applications that may benefit from automatic keyphrase extraction on the Web. Such applications include:

- Intelligent browsing:
  An automatic keyphrase extractor may be used as a browsing tool or as an automatic summariser of Web pages. For example, a browser may highlight useful terms or links on a page, or provide summaries of linked pages before the user visits them.
- Collaborative tagging Web sites:
  Automatically generated keyphrases may be displayed as suggestions for manual tagging.
- Bookmark organisation:
  Many Web users keep collections of bookmarks or favourite Web sites that they frequently visit or that contain useful information. These lists frequently become large and difficult to organise. Since these documents are Web pages, Automatic keyphrase extraction using TF×INF can be employed to automatically generate tags for these pages that can then be used as an index, or to cluster documents on similar subjects to aid future recall.

The most widely-used Web-based information retrieval system is the Web search engine. The use of a Web-aware keyphrase extractor can be used to improve the performance of Web search engines in a number of ways. For example, keyphrase extraction can improve the document indexing procedure by highlighting potentially important terms, that can then be given a higher weight in a document's index. Also, keyphrase extraction can be used to identify the probability that a query term is a keyphrase of a document, on the assumption that documents that contain the query term as a keyphrase are likely to be of interest to a user searching on that query term. This approach is studied in detail in the following chapter.

Non-Web Applications

Although TF$\times$INF KEA has been developed and tested with Web documents in mind, it is expected that it may also be successfully applied to non-Web environments that exhibit similar hyperlink behaviour. For example, before the explosion of the World Wide Web in the 1990s, a number of hyperlinked document collections existed in CD form, such as encyclopædias, games and reference manuals (e.g. ENQUIRE (Berners-Lee, 1980), and *HyperCard* (Goodman, 1988), used in the *Myst* computer game and the *Expanded Books* project), and hypertext remains an effective way of storing non-linear textual information.

Another example of a hypertext-like structure can be found in the reference structure of academic articles. References in articles point to older articles that cover a similar subject. This satisfies the assumption made on Web documents that hyperlinks between two documents represent a semantic link between them. In fact, the semantic link is likely to be stronger in this case, as less informative structural links, such as links to indices and menus, and unrelated links, such as advertisements, are not present. Since the destinations of the references are strictly older documents than the sources, and since documents cannot be changed once they are published, the resultant structure is a directed acyclic graph (DAG), as opposed to the structure induced by Web links, which may contain cycles. Despite this difference, the nature of the graph of academic articles is sufficiently similar to Web graphs to expect that similar properties relating to the discovery of context may apply. Intuitively, authors add references to documents in part to allow readers to locate more detailed descriptions of concepts mentioned in the article, that will allow the current article to be more easily understood. The article therefore exists in the context of the documents it cites. Using this definition, we can apply TF$\times$INF KEA to improve automatic keyphrase extraction of academic documents.

Hypertext-like structures can also be induced from other more implicit relationships between texts, such as text collocation within a magazine or book, texts that share an author, or texts that have been manually specified to be related by a reader, or a group of readers. Relationships such as these often imply a similarity of content or subject, that can be used to suggest the context of a document. In fact, the document need not be initially textual; for example, the minutes of a meeting may include an audio recording of the meeting, along with supplementary documents and notes that were relevant to the meeting. The transcribed audio recording is therefore a document that exists in the context of these supplementary documents. It therefore makes sense to incorporate the content of these documents in a calculation of the keyphrases of the transcript. 'Meeting browser' applications exist (Bouamrane and Luz, 2006) that allow users to navigate a Web-like structure based on multimedia representations of a meeting, including minutes, audio and video. TF×INF KEA may be a useful addition to such applications.

As mentioned above, the purpose of TF×INF is to exploit manual or implicit links between documents, as opposed to links automatically generated as a result of their content. This is the intention behind the TF×INF measure; however, it is possible that certain automatic links can suitably be used by TF×INF. Its value will be related to the effectiveness of the algorithm used to generate the links, and there may often be a more direct way to obtain the information, but TF×INF may remain a convenient method. For this reason, using TF×INF KEA to extract the keyphrases of documents with automatically generated links may be possible.

4.5.3 Sentence Extraction

An obvious next step from keyphrase extraction is automatic document summarisation, in which a set of sentences are generated that succinctly describe a document. These sentences can be extracted from the document or generated using templates or other methods. Jones et al. (2002) use standard KEA as a component in an automatic document summariser that extracts sentences from the document. In a Web environment, TF×INF KEA may be useful in this role.

4.6 Conclusion

This chapter described the problem of keyphrase extraction, its applicability to the Web, and the particular challenges faced by automatic keyphrase extraction on the Web. A description of the state of the art in automatic keyphrase extraction was provided and the KEA application was described in detail. A practical Web-based keyphrase extraction algorithm was then presented. This application is based on KEA, and incorporates the TF×INF measure. It is shown to outperform KEA, and by extension the state of the art in domain-independent machine-learning keyphrase extractors when applied to Web documents. We investigated the behaviour and parameter space of the algorithm and provided a recommendation for its optimal usage. Examples of applications were provided, Web and non-Web, that would benefit from this work. This chapter, and the following one, focus on the practical nature of TF×INF, and show that it can be of direct use to Web users and Web-based applications.

Chapter 5

Document Retrieval and Ranking

5.1 Background

Document retrieval is the identification of documents in a collection that are relevant to a user query. As the number of documents available on the Web has grown drastically, document retrieval has become vital to its success as an information repository. The Web has two principal mechanisms for document retrieval: Browsing and Searching.

During browsing, the user traverses hyperlinks between documents, searching for pages that satisfy an implicit information need, or query.[1] This strategy is enabled by the general rule that documents that are connected by a hyperlink contain related information.

The usefulness of browsing as a document retrieval strategy is restricted by the size of the Web, and by its decentralised nature, both of which result in a less comprehensive linking of related pages to each other. Documents that may be of value to the user may not be linked to documents they have visited, and will therefore not be found. Also, the identification of useful starting points on the Web graph is hindered by these properties of the Web. In the early days of the Web, Web sites such as Yahoo! were established that contained directories of manually categorised documents. Such directories served as a useful starting point, or Web Portal, for browsing. However, manually categorised directories soon struggled to keep up with the pace of expansion on the Web, and they became less useful as starting points for browsing.

As a result, Web searching, or retrieval by executing a query on a database of indexed Web documents, has become a hugely popular method of information retrieval on the Web. This approach was limited at first by the technology required to locate and automatically index a large number of Web documents. For a search engine to be useful, it must cover a sufficient proportion of the Web, and store a sufficient amount of information on each page for the automatic indexing procedure to compare to a manual one. As the cost of data storage decreased, search engines were

[1]Note: this is a simplification, and ignores such actions as casual browsing, where no information need exists.

able to cover a larger proportion of the Web, and return a large amount of documents relevant to a user query. The problem therefore transferred from document retrieval to document ranking.

5.1.1 Document Ranking

Document ranking is a traditional IR method that involves assigning relevance scores to documents, and sorting the result list for a query by these scores. A document retrieval application operating on a large information repository such as the Web will often return a vast number of documents in response to a query - too many to display on one page of results on a computer screen. Due to a number of reasons (including flaws in the indexing process, badly formulated queries, etc) in the IR system, it is not expected that all of the documents will be relevant to the user's information need, and among those that are, some will be more relevant than others. To speed up the location of relevant information in the search results, therefore, a document ranking procedure will attempt to estimate the probability that a document will satisfy the user's information need, and will rank those documents according to their probabilities. The user will then typically only be required to read the first few pages of search results to find a relevant document, and if not, they may consider reformulating their query instead of looking further.

5.1.2 Document Ranking Schemes

The Vector-Space Model (VSM)

Many non-Web-based information retrieval systems calculate document relevance by applying the Vector Space Model (VSM). In this method, pioneered by Salton et al. (1975), terms in a vocabulary are treated as orthogonal dimensions in a high-dimensional space. Documents, therefore, are vectors within this dimension space, with values representing the term weights of a term within the document. These term weights can be their simple frequency within the document, or their TF$\times$IDF value (see section 3.1.1). A corpus, or collection of documents, can then be represented as a sparse term-by-document matrix. In this framework, terms are treated as independent (see the independence assumption in section 4.1.5), and documents are merely 'bags of words', i.e. the location of the word in the document is not stored.[2]

Queries, too, can be represented as vectors in this space, and the relevance of a document to a query can be interpreted as the distance between the query vector and the document vector - typically calculated as the cosine of the angle between the two vectors, or by the Euclidean distance between them. During document retrieval, a set of documents (either the entire collection, or a subset identified by matching all or one of the query terms with the documents' contents) will be compared to the query in this way and the result will determine their ranking in the result list.

[2]Note: term-weighting systems other than TF$\times$IDF may incorporate term location in a document.

Extensibility of the VSM

The Vector Space Model is easily extended to incorporate more documents or a larger vocabulary, as terms are treated as independent orthogonal dimensions, and documents are independent vectors within the model. Adding a document to the term-document matrix can therefore be carried out by adding an extra row representing the document vector, and new terms can be added by appending a column vector of zeros to the matrix. The worst-case time complexity of the vector comparison operation is $O(n)$ where n is the number of terms in the vocabulary. In practice, due to the sparseness of the matrix, this will be reduced.

Dimensionality Reduction

A problem with the VSM is that it does not discriminate between informative and uninformative terms within a document or collection of documents. As a result, a large amount of space is wasted. Also, as mentioned above, the complexity of the vector similarity calculation is linear with respect to the vocabulary size. One method to reduce these problems is to apply dimensionality reduction to the term-document matrix. Some dimensionality reduction techniques simply involve removing frequent terms (stop-words), or stemming terms to combine them with others. More advanced techniques reduce the term list to a smaller abstract 'feature' list by carrying out Eigenvalue decomposition on the term-document matrix (Deerwester et al., 1990). While dimensionality reduction can have a strong positive effect on retrieval effectiveness (Zha et al., 1998), the resultant matrices are often more difficult to update as new documents are included, or as document contents are changed. This restricts their value on the Web, and Web search engines typically apply only the simpler techniques such as stemming and limited stop-word removal.

Web-based Document Ranking

The hyperlink structure of the Web is ignored by the VSM document retrieval method. Chapter 2 gave an overview of attempts to incorporate hyperlink information into retrieval systems. They fell generally into three categories -

- Using the Web to identify document authority or prestige (PageRank, HITS - see section 2.1)
- Using the Web to identify similar documents in order to increase the information available to an indexer or query algorithm. (see sections 2.2 and 2.3).
- Allowing the user to explicitly define the expected contents of related pages (see section 2.4)

The first category treats only the structural context of the Web. The second interprets hyperlinks as providing similar information, as opposed to contextual information as we define it in

Table 5.1: Contingency Table for the calculation of Precision and Recall values (taken from Van Rijsbergen (1979))

	Relevant	Non-relevant	
Retrieved	True Positives ($A \cap B$)	False Positives ($\bar{A} \cap B$)	B
Not Retrieved	False Negatives ($A \cap \bar{B}$)	True Negatives ($\bar{A} \cap \bar{B}$)	$\bar{B}$
	A	$\bar{A}$	Total docs (N)

the introduction. The third category requires the user to consider the document context explicitly when making a query.

Our method is superficially similar to the second category, in that it uses the textual contents of related documents to augment the index of a Web document. The contextual information is therefore captured at the indexing stage, rather than at query-time. The query-time complexity is not increased by using this method. The difference between this method and the methods in this category is the manner in which the contents of related documents are used.

5.1.3 Evaluation of Document Ranking - Precision and Recall

Ranking schemes in IR are often evaluated using the Precision and Recall measures. Both measures are calculated using a binary classification system of document relevance. They do not take into account the degrees to which a given document may be relevant to the query.

Precision

Precision is the proportion of documents in the set of retrieved documents that are relevant. In table 5.1, precision is calculated as:[3]

$$\text{Precision} = \frac{|A \cap B|}{|B|} \tag{5.1}$$

Recall

Recall is the proportion of all relevant documents that are retrieved. i.e.:

$$\text{Recall} = \frac{|A \cap B|}{|A|} \tag{5.2}$$

As more documents are added to the result list, the precision value will tend to decrease. It is at its highest if all the retrieved documents are relevant to the query, even if only a small number of relevant documents are retrieved. The recall measure does not take into account the number of retrieved documents that are not relevant to the query, and 100% recall can be obtained by including every document in the result list. This approach will give minimum precision.

It is common practice in modern search engines to return every document that matches a given query, regardless of its relevance, giving a recall value of 100% (or as close as is possible using term matching), and a minimum precision value. The result list is typically presented to the

[3]Precision and recall formulae are taken from Van Rijsbergen (1979).

user in a series of pages, as described in 5.1.1. The value of a search engine is measured by the success of its ranking algorithm. For this reason, the precision and recall values of the entire list are unimportant, but we can measure the success of the search engine by analysing the precision and recall values at varying intervals in the ranked list, for example after every i documents. In particular we are interested in the number of those i documents that are relevant, or the precision at cut-off i of the list, as well as the proportion of the total relevant documents that are in the set of i documents, or the recall at cut-off i. Since we know that the recall value at cut-off n for a list of n documents is always 100%, we can combine precision and recall to form a single measure by locating cut-off points in the ranked list at given recall values, and calculating the precision at these points. For example, at a recall value of 20%, the precision may be calculated to be 30%. This measurement is independent of n, the result list size, and the number of relevant documents.

Specific recall cut-off points, such as 20%, can be used to evaluate a ranking system, however, a more comprehensive system is to graph precision against recall for every document in the result list, and then analyse the resultant curve. The curve resembles a saw-tooth pattern, as each document

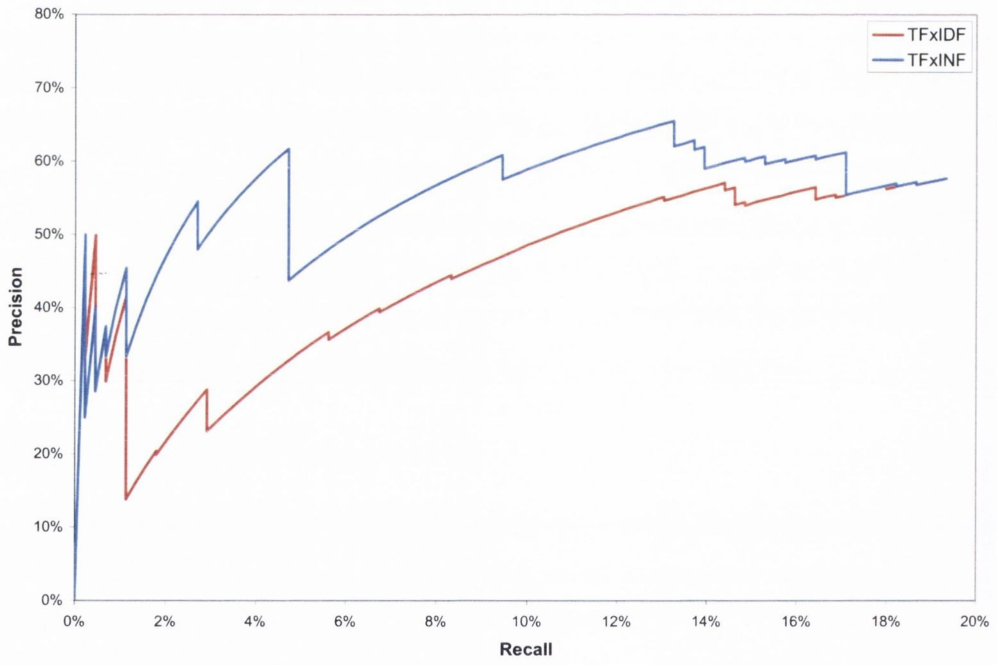


Figure 5.1: Sample precision-recall curve for the search query 'government' on the Oireachtas corpus

is either irrelevant, in which case the precision will drop and the recall will remain static, or it is relevant, in which case both precision and recall will increase (unless precision is already at 100%). As a result, each recall value can have a range of precision values.

A perfect document ranking system would place all relevant documents at the top of the list, followed by any irrelevant documents. The resultant precision-recall curve would show 100% precision until recall reached 100%. The area under the curve would therefore be 1. Any drop in precision would reduce the area under the curve. An irrelevant document near the top of the list will reduce the precision at that position by more than an irrelevant document near the bottom of the list,[4] reducing the area of the curve by a larger amount. Therefore the area under the curve is a suitable means of measuring the success of the document ranking algorithm (Salton and McGill, 1983). The area under the curve can be calculated by taking a sum of the discrete function; however, this is simplified in practise by estimating the area using a sampling technique. For example, the 3-point precision average measure involves sampling the precision curve at recall values of 25%, 50% and 75%, and then averaging the results.

5.2 TF×INF in Document Ranking - Method

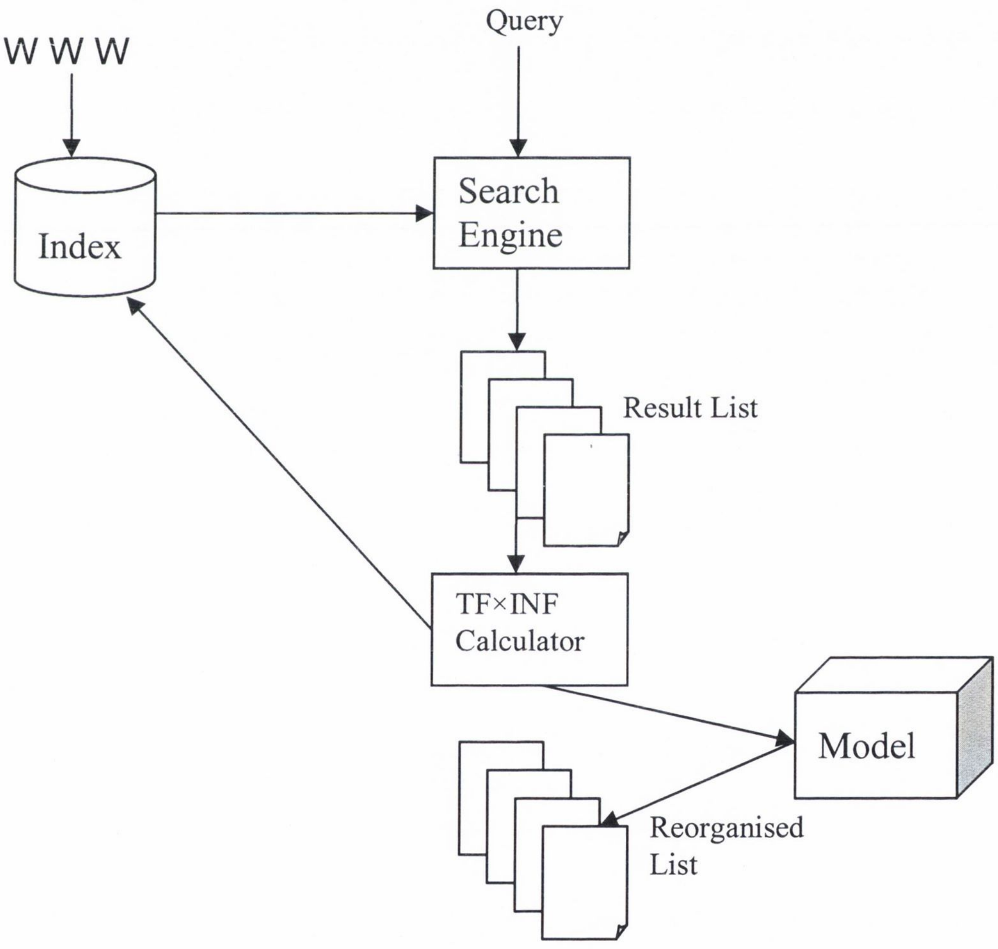


Figure 5.2: Architecture of a ranking post-processor

[4]If x is the number of relevant documents in the top n documents, the precision will drop by $\frac{x}{n+1} - \frac{x}{n}$ if the document at $n+1$ is not relevant. This reduction is lower as n rises.

5.2.1 A ranking post-processor

In this chapter, we present a post-processor for a document retrieval application. The post-processor reorganises a result list generated from a Web search engine, according to a linear combination of the RSVs assigned to the documents from that search engine, and the RSVs obtained from a TF×INF-based document ranking algorithm:

$$RSV_{final}(d) = (1 - \alpha) \cdot RSV_{orig} + \alpha \cdot RSV_{\mathrm{TF \times INF}} \qquad (5.3)$$

where $0 < a <= 1$

Using this framework, we can apply the document re-ranking algorithm to any search engine that provides individual document scores as well as document ranks.

5.2.2 The TF×INF-based Document Ranking Algorithm

The TF×INF-based ranking application has three parts (figure 5.2):

- The Indexer, which generates an index of document content and hyperlinks from which to calculate TF×INF values.
- The Model Generator, which creates a probabilistic model of document relevance according to TF×INF values of their terms.
- The Ranking application, which looks up the document index, and applies the model to obtain new retrieval scores for documents ($\mathrm{RSV}_{\mathrm{TF \times INF}}$)

5.2.3 The Indexer

Index Structure

The document ranking algorithm described in this chapter uses a document index in order to calculate the TF×INF values of terms in the result documents. The index for a document is made up of its individual terms and their locations in the documents, and a list of hyperlinks to other documents in the collection. From this index, TF×INF values can be calculated.

An alternative to this structure is to use an index that contains pre-calculated TF×INF values for each term (as opposed to their frequencies). This index avoids the need for hyperlink information, but requires a recalculation of document indices whenever a document that they link to is added, removed or changed.

Another alternative avoids the need for an index entirely, reducing the space and time costs involved in creating the index and maintaining it as documents are changed or added to the collection. In this alternative, the contents and context sets of each document in the result list are analysed at query-time by downloading the HTML documents and processing their contents. This

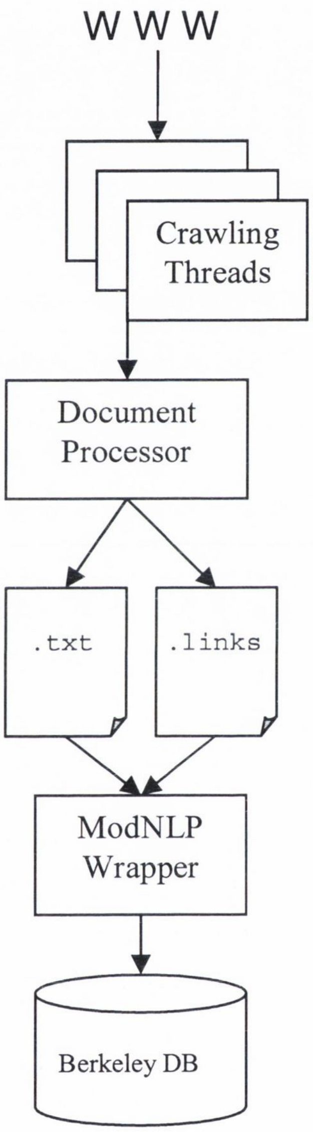


Figure 5.3: Architecture of the Web indexer

Table 5.2: Tables in the index database

Name	Fields	Function
File Table	Number, Name	Associates each document in the collection with a unique file number
Word File Table	Word, File Number	Identifies the terms contained in each document
Case Table	Word, Set of Words	Associates a lower-case form of a term with a set of forms in which it occurs in the collection
Word Position Table	Word, Set of Positions	One table exists for each document - stores a list of positions of each word in the document
Link Table	File Number, Set of File Numbers	Identifies the documents that are the targets of the hyperlinks in a document

approach allows the application to be employed in environments with tight space constraints, but results in slower query times.

In this chapter, we use the first approach, and generate an index with term locations and hyperlinks. This allows us to employ text-based and hyperlink-based search algorithms for tests in conjunction with, and against the TF$\times$INF ranking algorithm described here.

The index takes the form of a database implemented using the Java form of Berkeley DB.[5] Berkeley DB is an embedded database library that provides a mapping from database tables to Java objects. The primary motivations for its use here are that it is easily integrated with other Java applications, allowing us to incorporate it into the existing keyphrase extraction software, and it is easily portable across platforms, as it is implemented in Java.

We adapt a NLP wrapper for Berkeley DB.[6] This wrapper includes tables specifically designed for NLP applications, such as document content tables and word position tables. We add a table representing hyperlinks between documents. The tables used in this index are shown in 5.2.

The Web Crawler

The indexing application includes a threaded Web crawler application that locates and downloads Web documents for inclusion in the index. The documents are located by spreading across hyperlinks from a starting location specified when the crawler is started. The crawler application is described in detail in the Appendix C.

The Document Processor

Once a document is downloaded by the Web crawler, it is converted into a set of three documents:

- *filename*.txt The textual content of the document - i.e. the text that the user sees on the screen

[5]`www.oracle.com/technology/products/berkeley-db/index.html`
[6]ModNLP: `modnlp.berlios.de`

- *filename*.links A list of URLs obtained from the href attributes of the `<a>` and `<link>` tags in the HTML document
- *filename*.key A list of the keyphrases contained in the keyword attribute of the `<meta>` tags in the document - for evaluation purposes.

This is carried out using a perl script (described in appendix B) that strips the HTML tags from the document.

5.2.4 The Model Generator

The purpose of the model generator is to create a probabilistic model of document relevance according to the TF×INF scores of query terms that appear in the documents. We use a Naïve-Bayes model equivalent to the model described in the previous chapter, incorporating the term features TF×INF and first occurrence.

Applying the Bayes rule to the probabilistic system of document retrieval (Van Rijsbergen, 1979) gives us:

$$p(d|q) = \frac{p(q|d)p(d)}{p(q)} \tag{5.4}$$

where:
$p(d|q)$ is the probability that a document d satisfies the information need represented by the query q.
$p(d)$ is the prior probability of document d. This is typically set to be constant for all documents, with each document having a $\frac{1}{D}$ chance of being selected a priori (where D is the overall document collection)
$p(q)$ is the prior probability of the query occurring. This is independent of d, and can therefore be treated as a normalisation factor N.

Therefore:

$$p(d|q) = p(q|d) \cdot \frac{1}{D \cdot N} \Rightarrow p(d|q) \propto p(q|d) \tag{5.5}$$

So to discover $p(d|q)$, it suffices to calculate $p(q|d)$, which is typically defined as:

$$\text{TF}\times\text{IDF}(q, d) \tag{5.6}$$

or:

$$\prod_{i=0}^{n} \text{TF}\times\text{IDF}(t_i, d) \tag{5.7}$$

for a query $q =< t_0, t_1, \cdots, t_n$ (using the independence assumption)

Here, we replace TF$\times$IDF with TF$\times$INF, in order to incorporate contextual information in the calculation of $p(q|d)$. However, since TF$\times$INF is not a simple linear term weight, we use a probability model based on TF$\times$INF to calculate $p(q|d)$. We assume $p(q|d)$ to be $p((\text{q is a keyphrase})|d)$, or the probability that query term q is a keyphrase of document d, on the intuitive notion that a document that is 'about' keyphrase q is likely to be considered to be relevant to a query on q.

This allows us to use the probability models created by the KEA model builder for information retrieval, allowing us to calculate the probabilities $p((\text{q is a keyphrase})|d)$ according to the TF$\times$INF and first occurrence values of q in d.

This method provides us with a probability value for each of the documents in the original result set. By ordering the documents according to these values the new ranked set is obtained.

Ranking Application

The document ranking application orders a set of result documents obtained from a search engine according to the scores calculated using the probability model. The input of the application is therefore a set of documents and a probability model, and the output is the same documents in an ordered list. If the application is being applied as a post-processor of another ranking application, documents in the input list are associated with relevance scores, and the output order is based on a linear combination of the input scores and the TF$\times$INF probability values (as described above). The input to the application in this case is a set of document-score pairs, a probability model and a proportion value α between zero and one. Setting α to 0 will not change the scores at all, and increasing the value of α will give the TF$\times$INF probability values more influence on the document scores.

The document score calculated by this application is an estimation of the probability that the query is a keyphrase of the document. It is therefore in the range [0,1]. Other ranking algorithms may output scores in a different range, and this will have an effect on the value of α.

5.2.5 Composition of Ranking Algorithms

A document ranking scheme is essentially a function that maps a set of documents to a set of document scores. Multiple ranking schemes can be combined by calculating the weighted sum of their scores for each document. Assuming the ranking schemes use independent information, this allows a search engine to combine a number of factors or evidence about a document's relevance into a single score that can then be used to rank the document. For example, given n ranking schemes:

$$RSV_d = \sum_{i=0}^{n} a_i \cdot RSV_{d,i} \tag{5.8}$$

where:

a_i is the weight assigned to ranking scheme i ($\sum_i a_i = 1$), and
$RSV_{d,i}$ is the score obtained by ranking scheme i on document d.

In the following sections, we consider the composition of the TF$\times$INF ranking scheme with existing schemes in order to determine if it provides additional information to a search engine.

5.3 Evaluation

5.3.1 Test Environment

In order to train the model, the four keyphrase-annotated corpora described in chapter 4, based on the Java, No10 and Oireachtas Web sites, were used. In these experiments, a document is deemed relevant to a query if the terms of the query appear in the keyword list of the document. This allows us to use these corpora for evaluation as well as training. We generate a number of models from training data from each of the corpora, and test these models on each corpus in turn. In the cases where the same corpus is used for training and testing, 50% of the corpus is used to train the model, and the other 50% is used for testing, in order to avoid biased results.

The KEA model was shown to be reasonably domain-independent in the previous chapter, and we expect this result to be transferred to some degree in the field of IR.

It was decided to generate domain-specific models (i.e. models trained with content from a specific corpus) as opposed to taking samples of content from all corpora, that would give a potentially more flexible domain-free model. The problem with doing this is that the linkage in the documents used to generate the model would be low, there being no inter-corpus links. Also, in the wider context of the scale-free Web, a small number of documents are responsible for a large amount of hyperlink structure (see section 2.1). The probability of a random sample of the Web containing one of these documents would be very low, and the resultant model would have atypically low linkage.

The ranking algorithm is evaluated using the 3-point average of the precision-recall curve described above. All values reported are therefore based on the average of the 3-point precision values obtained from a set of queries performed on the system. We use the fifty most common keyphrases in each corpus as query sets. The most common keyphrases are chosen so that the resultant document lists will contain enough relevant documents to obtain meaningful 3-point average precision values.

A basic search engine was created that finds all documents in the index that contain the query terms, and gives them equal score. In these tests, we ignore the mechanics of query formulation logic (e.g. Boolean logic, p-norm, fuzzy logic etc.) and focus on the application of contextual information in simple search queries. Queries are therefore treated as single words or phrases and must appear in their entirety in result documents. This also allows us to avoid implementing the vector space model. Since each query consists of a single term or phrase, the properties (such as TF×IDF) of the query terms themselves are unimportant.

5.3.2 Benchmarks

The TF×INF ranking algorithm is compared with a number of alternative ranking applications that are implemented on the index software for testing purposes. These applications are:

TF×IDF

This ranking scheme orders documents according to the TF×IDF values of the query term in the document. The *df* value is calculated by counting the number of documents containing the term in the index. It is therefore equal to the size of the initial result list. Note that TF×IDF, as described in section 3.1.1, is based on a collection, and the *df* component will vary according to the scope and size of the collection.

HITS

We developed a variation of the HITS algorithm described in section 2.1 in order to investigate the hypothesis that TF×INF captures different information to the authority information captured by HITS. HITS ignores all internal links (links between documents in the same domain), as it treats authority as a property conferred on a document by votes from external documents (i.e. those not under the control of the document author). Our corpora, however, contain only internal links. In order to implement the HITS algorithm on our corpora, we relaxed that constraint, although doing so would naturally reduce the value of the HITS algorithm as a calculator of authority. Therefore we cannot directly compare TF×INF with HITS as a ranking algorithm, but we can test the hypothesis that TF×INF and HITS provide different information about document relevance, with the null hypothesis being that TF×INF and HITS perform similarly on these corpora.

PageRank

In order to more thoroughly compare TF×INF with authority and prestige-based ranking algorithms, we also developed a variation of the PageRank algorithm described in section 2.1. Since the PageRank algorithm is an iterative algorithm over the entire structure of the Web, and this structure is unavailable to us, our simulation of the PageRank algorithm uses a query-dependent document set, as in the HITS algorithm above, but including all documents connected by up to four hyperlinks to documents containing the search query terms. As above, internal links are used instead of external links. Due to these constraints, it is again expected that this algorithm will not perform at its optimal level, but will allow us to determine if TF×INF provides different information to PageRank.

Relevance Propagation

In section 2.2, we described a set of hypertext-based IR methods that adjust the retrieval value of a document according to the retrieval values of its neighbours. Documents that are surrounded by relevant documents in this scheme are more likely to be retrieved than those that are not. This method is intuitively suitable for hypertext graphs based on science articles, and citation links between them, as citations follow a reasonably rigid semantic framework, pointing to similar or background works that share motivation, methodology, materials or have comparable or contrasting results or conclusions, for example. Furthermore, as pointed out in Kwok (1988), scientific collections use a rigid scientific sublanguage and a large amount of technical jargon that may make retrieval easier. Consequently, the CACM collection of science article abstracts is often used to test hypertext-based IR methods (e.g. Savoy (1996)). It is not evident, however, that such a scheme captures the notion of context on the Web, where hyperlinks follow a less rigid semantic interpretation. One of the hypotheses of this section is that TF$\times$INF captures contextual information more comprehensively than methods that simply look for textually similar neighbours. In order to test this, we developed a relevance-propagation algorithm based on Savoy (1996). This is not a ranking algorithm on its own, but can be implemented as a post processor to a text-based ranking algorithm such as TF$\times$IDF. It will adjust the relevance scores of documents relative to the relevance of their neighbours. Relevant documents that are in the neighbourhood of other relevant documents will have their scores increased, while relevant documents in the neighbourhood of non-relevant documents will have their scores reduced in comparison. Chapter 2 describes the differences between this interpretation of context, and the broader interpretation suggested in this work. TF$\times$INF is expected to cover this broader interpretation of context, and as such may be expected to capture information that Relevance Propagation does not, resulting in higher precision averages.

5.4 Results

5.4.1 Comparisons with TF$\times$IDF

We tested the TF$\times$INF ranking post-processor on results lists ranked by the TF$\times$IDF algorithm, varying the constant of proportionality, α. TF$\times$IDF values are small in comparison to the probability values generated from the TF$\times$INF model, since they are normalised by the number of terms in the document. Reducing α allows us to compensate for this, so low values for α were used in this experiment. Figure 5.4 gives a summary the results of the experiment. The values shown are the differences between the 3-point precision averages of TF$\times$IDF and TF$\times$INF as a percentage of the TF$\times$IDF value. Important observations on these results are:

- The results vary greatly with respect to the test set. Figure 5.6 shows the average values for each test set. The Oireachtas and BigJava sets perform poorly, while the SmallJava and No10 sets perform well. It appears that this observation is related to the TF×IDF benchmark, shown in figure 5.5. The TF×IDF ranking algorithm has more success with the Oireachtas and BigJava corpora than the others, suggesting that documents in these corpora may contain sufficient information to adjudicate their relevance to the query without inspecting their contexts. It is interesting that the corpora that give the best results in TF×IDF-based IR experiments perform the poorest in the KEA experiments, and that TF×INF provides the most improvements in KEA experiments on these corpora (both in absolute terms and relative to the benchmarks), whereas in IR TF×INF gives little or no improvement. We can suggest the following possible reason for this. In keyphrase extraction the intention is to provide a set of descriptive keyphrases for a document, whereas in the IR experiments the intention was to select the documents that are most relevant to the query from a set of related documents. The ability to distinguish between documents is therefore important in IR. TF×IDF is more suitable for the latter exercise, since it gives higher weights to unique or rare terms than common ones. In keyphrase extraction, the emphasis is on description and not discrimination, and TF×IDF may therefore be less suitable than TF×INF for that purpose.

- The optimum value for α across all test sets and models is between 0.02 and 0.04. At these levels, the average improvement over the TF×IDF benchmark is almost 6%, although much higher improvements are reported on the SmallJava and No10 test sets when using some models.

- At $\alpha = 1$, the TF×INF ranking scheme rarely improves over TF×IDF. At $\alpha = 1$, the results of a search query are ranked exclusively by the TF×INF ranking scheme with no influence from TF×IDF. These tests demonstrate that, when *df* is calculable, it is of value in an IR application.

- The choice of training set has, on average, little effect on the outcome of the experiment, as seen in figure 5.7. The Oireachtas model can be seen to give the best performance, but it does not consistently perform better than the other models.

5.4.2 TF×IDF with Relevance Propagation

The Relevance Propagation post-processor was applied to the TF×IDF ranking algorithm using the optimum settings reported in Savoy (1996). Figure 5.8 shows the effect that Relevance Propagation has on the 3-point precision averages observed on each dataset. The No10 and Oireachtas corpora are positively affected by RP, while the two Java corpora suffer a decline in performance. The TF×INF ranking scheme was then applied to these new benchmarks, and the results are

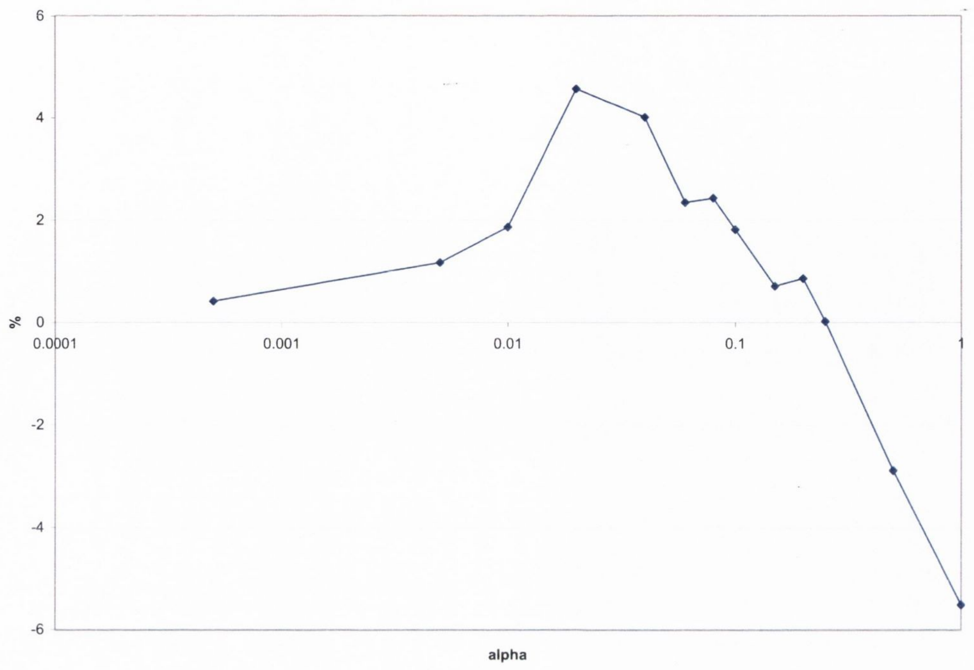


Figure 5.4: Average percentage change of 3-pt precision average from TF×IDF benchmark for all test sets and all models

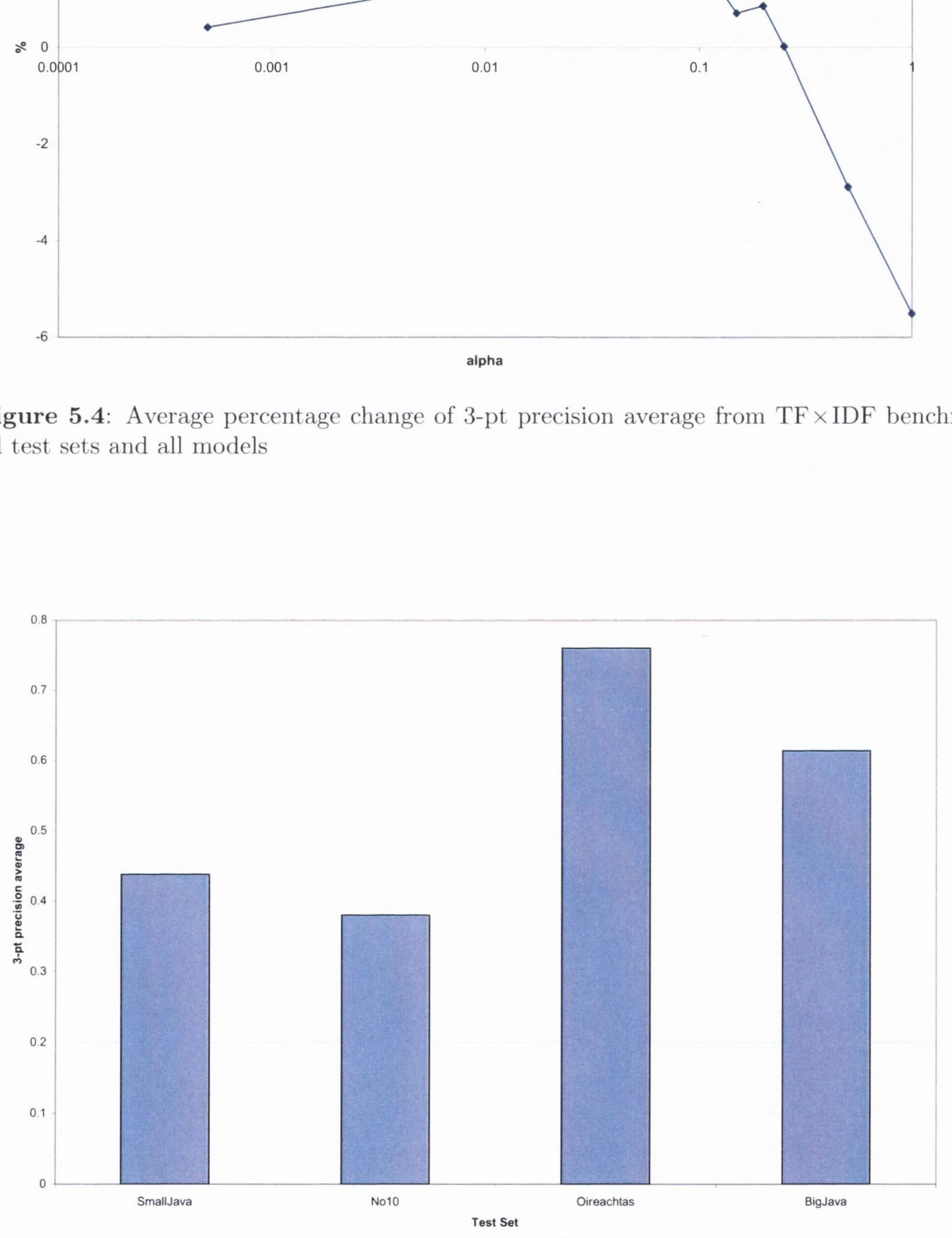


Figure 5.5: 3-point precision average values of TF×IDF benchmark

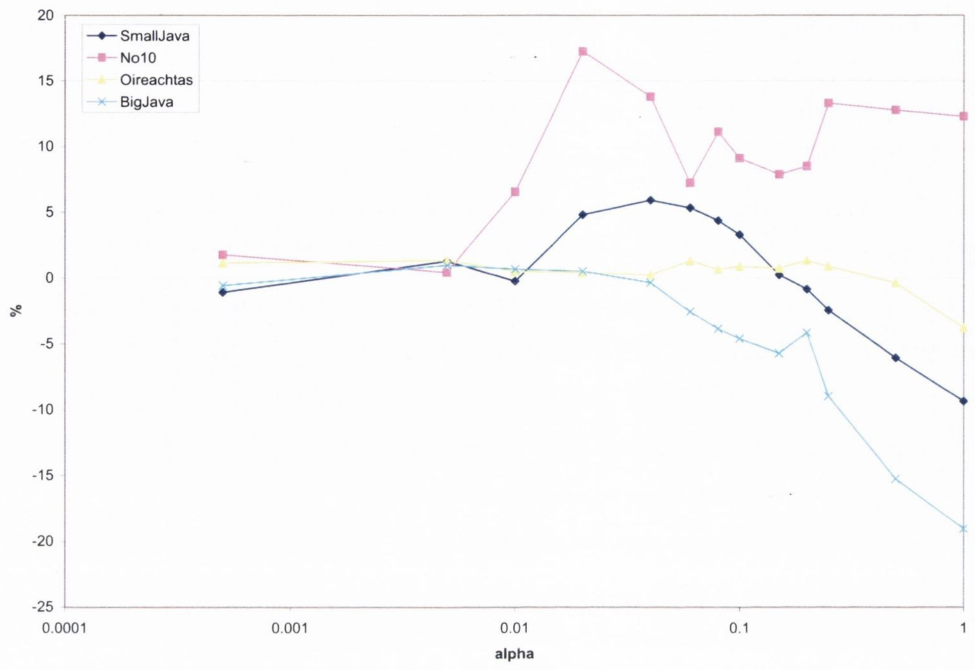


Figure 5.6: Percentage change of 3-pt precision average from TF×IDF benchmark, averaged by test set

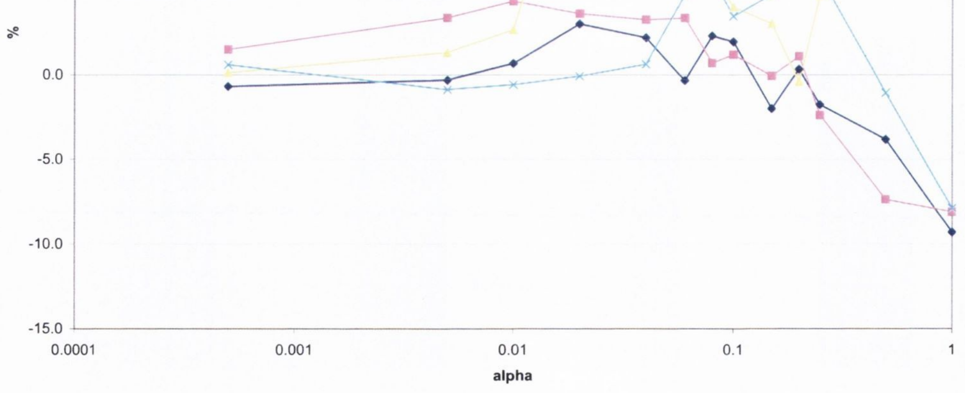


Figure 5.7: Percentage change of 3-pt precision average from TF×IDF benchmark, averaged by model

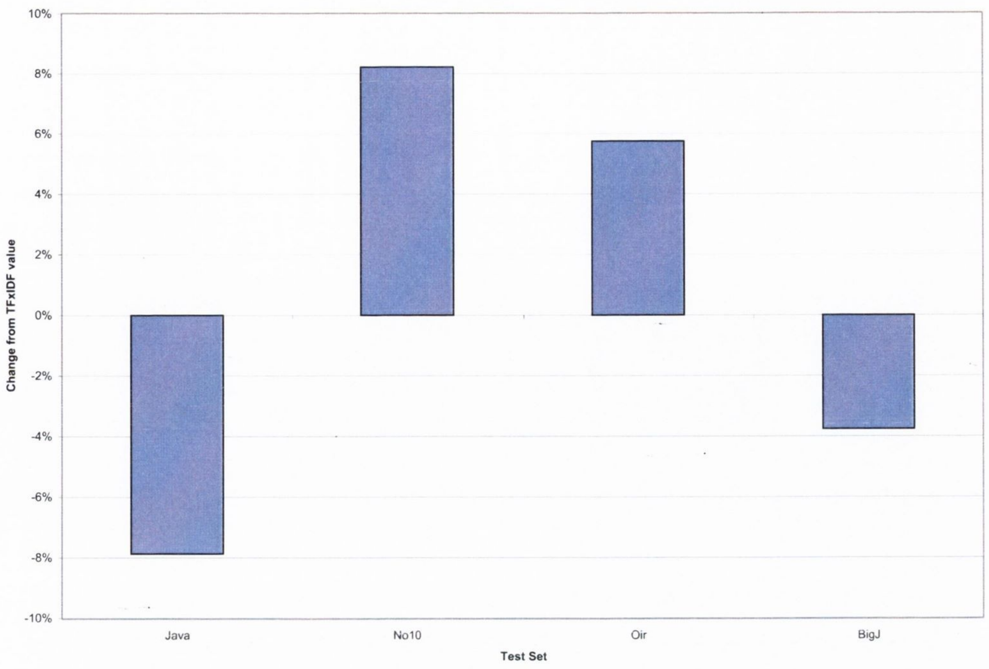


Figure 5.8: Percentage change of 3-pt precision average scores of Relevance Propagation over TF×IDF benchmark

summarised in figure 5.9. These results show clearly that two corpora, SmallJava and No10 show significant improvements over these benchmarks, especially in the range of alpha values 0.001 - 0.6, while the other two corpora give little or no improvement.

5.4.3 Analysis of Individual Datasets

Based on the information from the TF×IDF and TF×IDF + RP experiments, we can make the following observations of the role of contextual information in information retrieval on each corpus:

SmallJava

RP has a negative effect on this corpus in comparison to the TF×IDF benchmark (roughly -8%), while TF×INF has a positive effect (≈5%). The upper-bound of the improvement over TF×IDF + RP is roughly 12%, suggesting that TF×INF can undo the damage done by RP. We can conclude from these results that the information provided by hyperlinks in the SmallJava corpus allows TF×INF to act as a form of specialisation of the discriminatory effect of TF×IDF, as opposed to an indicator of similarity, as required by RP.

No10

The No10 corpus is improved both by the TF×INF ranking algorithm and by the RP algorithm. The improvement over the TF×IDF + RP benchmark is approximately 7.5%, which, when combined with the RP benchmark improvement of 8% gives approximately 15.5%. This is roughly

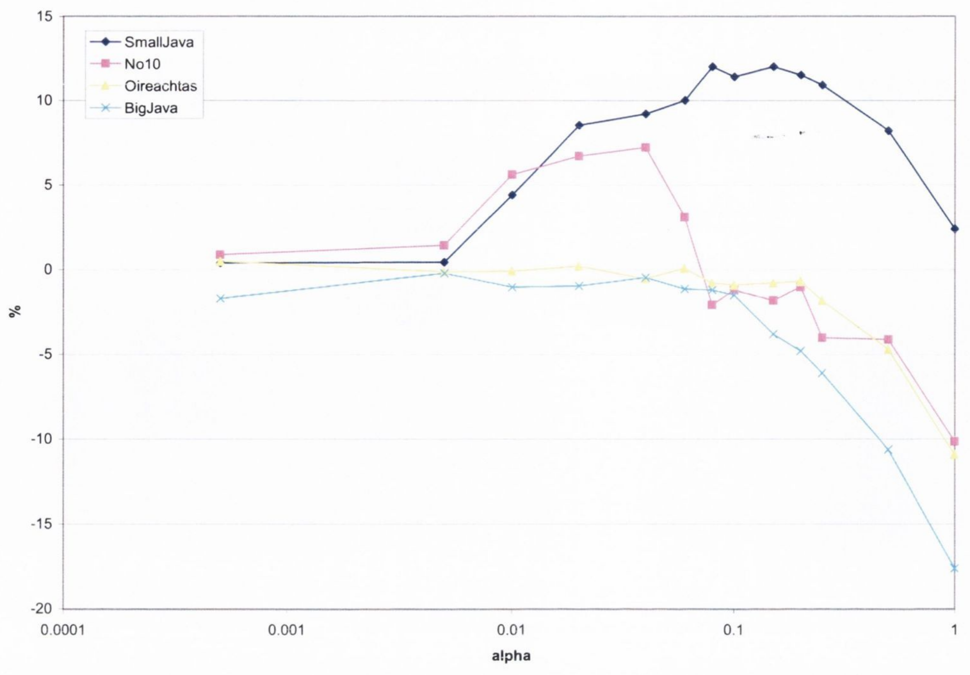


Figure 5.9: Percentage change of 3-pt precision average scores of TF×INF over TF×IDF + RP benchmark, averaged by test set

equal to the optimum TF×INF improvement over the initial TF×IDF benchmark (without RP) of 14 to 17%. We can therefore conclude that in this case, TF×INF captures two separate forms of contextual information: propagation of relevance across hyperlinked documents, and the information obtained by contrasting the contents of hyperlinked documents.

Oireachtas

In this corpus, RP records a 6% improvement over the TF×IDF benchmark, while TF×INF offers little or no improvement. We can conclude from this that while the relevance of a document is influenced by the relevance of related pages, this information is not adequately captured by the TF×INF ranking algorithm. The most likely reason for this is the distribution of keyphrases in the Oireachtas corpus that are used to formulate the queries and relevance information for each document. Although each document in the Oireachtas corpus contains approximately twice as many keyphrases as documents in the other corpora (see chapter 4), the average number of unique keyphrases in each document is much smaller (figure 5.12). Figure 5.10 shows the frequencies of the fifty most common keyphrases in each corpus as a proportion of the number of documents in which they occur. These follow an apparent power distribution, with varying power coefficients. In general, the skew on the Oireachtas data is less than the others (after the first few popular keyphrases), and on average, the proportion of documents in the Oireachtas corpus that contain one of the top fifty keyphrases is more than twice that of the No10 corpus. Consequently, hyperlinked documents are more likely to share the same keyphrases (and therefore relevant to the same queries in our tests) in the Oireachtas corpus than in other corpora. This could reflect the semantics of the hyperlink structure of the Web site, or a lack of specificity in the keyphrases.[7] This property of the corpus makes it ideal for RP, since the hypothesis of RP is that linked documents are relevant to the same query. The optimal parameters suggested in Savoy for the RP algorithm and adopted here include weighting the RSVs of neighbouring documents independently of the out-degree of the document (i.e. the number of neighbouring documents). A highly-linked document will therefore benefit from this property of the corpus even if the contents of the neighbouring documents are only slightly relevant to the query. In order to verify this effect, the TF$\times$INF ranking algorithm was re-tested on the Oireachtas corpus, this time by generating the queries from the list of all 814 keyphrases in the corpus, rather than the top fifty most common keyphrases. Those keyphrases that are present in a large proportion of documents, regardless of content, are therefore less likely to be used as queries. Figure 5.11 shows the results of this experiment, averaged across models generated from the other corpora. The improvement over the TF$\times$IDF baseline by TF$\times$IDF is much larger here, suggesting that TF$\times$INF performs well if the keyphrases used as queries are not over-general.

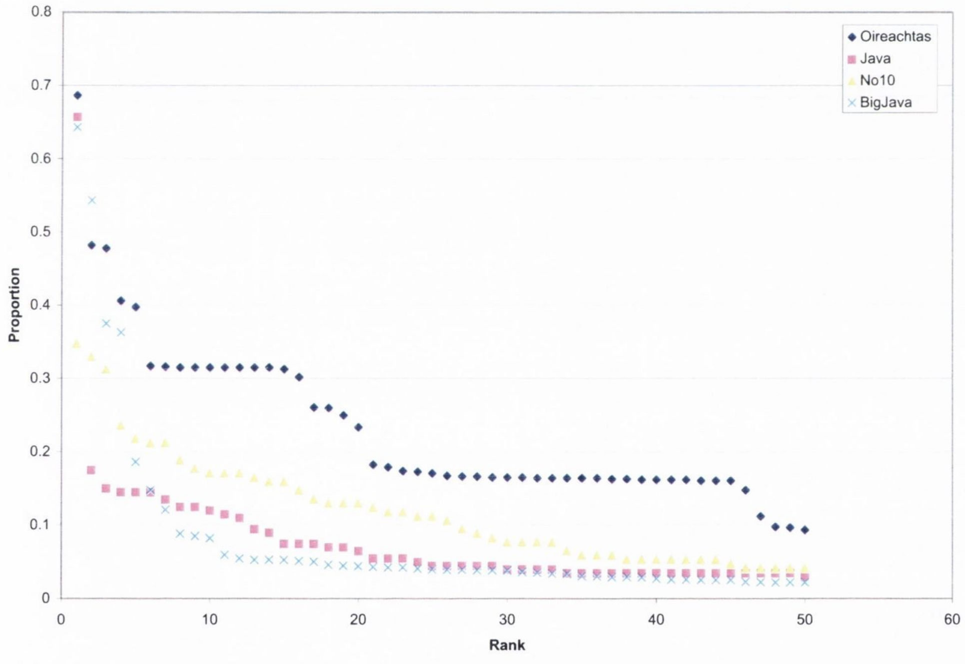


Figure 5.10: Relative frequencies of keyphrases in the corpora

BigJava

This corpus performs poorly using both RP and TF$\times$INF, suggesting that hyperlinks offer little information to the IR system that is not already available under TF$\times$IDF. The reason for the

[7]Sample keyphrases in the top fifty in this corpus are: *Government, Ireland, Irish, Dublin, Parliament, Oireachtas.*

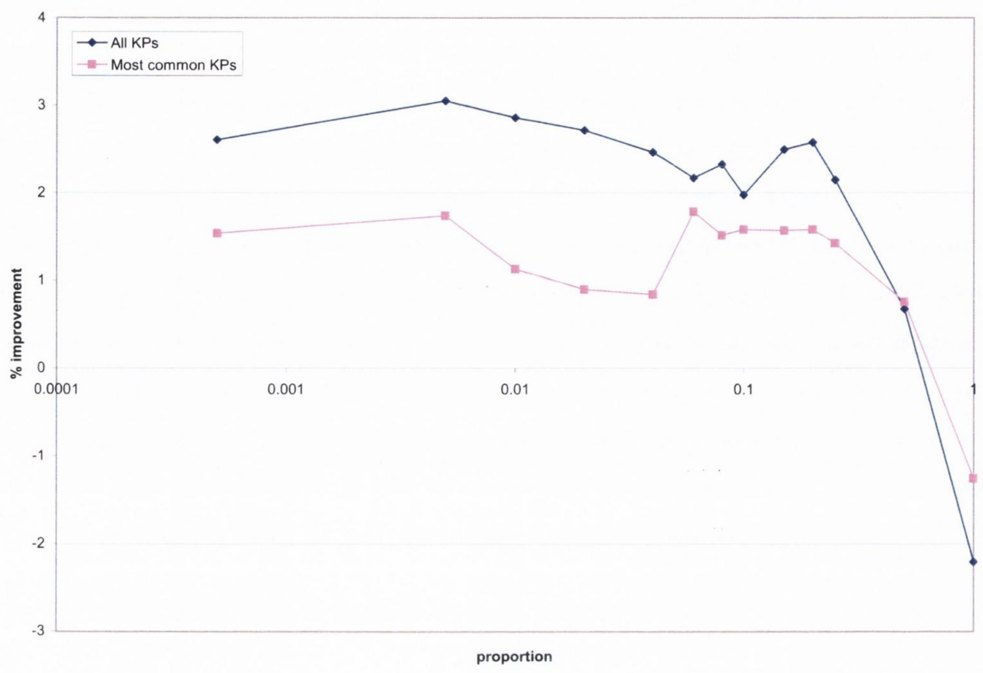


Figure 5.11: Percentage change of 3-pt precision average from TF×IDF benchmark of Oireachtas test set, using queries taken from set of all keyphrases, or most common phrases

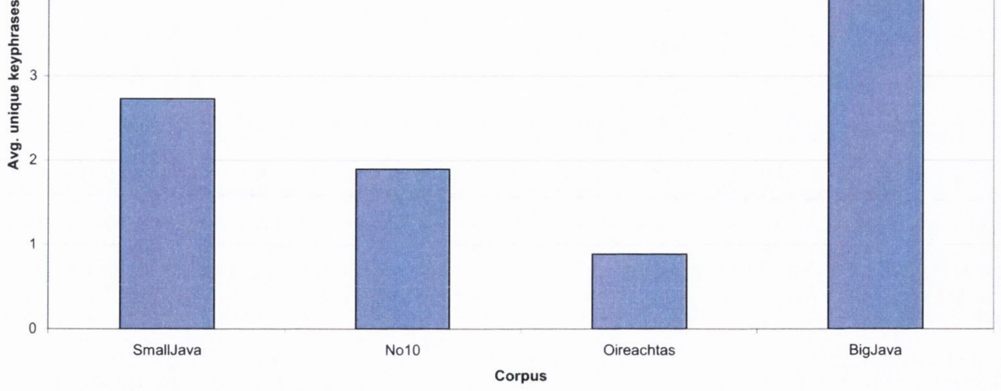


Figure 5.12: Average number of unique keyphrases per document

poor performance of RP is evident from figure 5.12. On average, each document contains over six keyphrases that are not present in any other documents. The probability that hyperlinked documents are relevant to the same query is therefore much lower in this corpus than in the others. This also suggests that rare terms are likely to be useful identifiers of document relevance, an application to which TF×IDF is well suited. This explains the success of the TF×IDF benchmark on this corpus.

5.4.4 Link Topology Ranking Schemes

Figures 5.13 and 5.14 show the effects of applying the TF×INF ranking scheme as a post-processor of relevance percentage change in 3-point precision average values. In all cases, TF×INF pro-

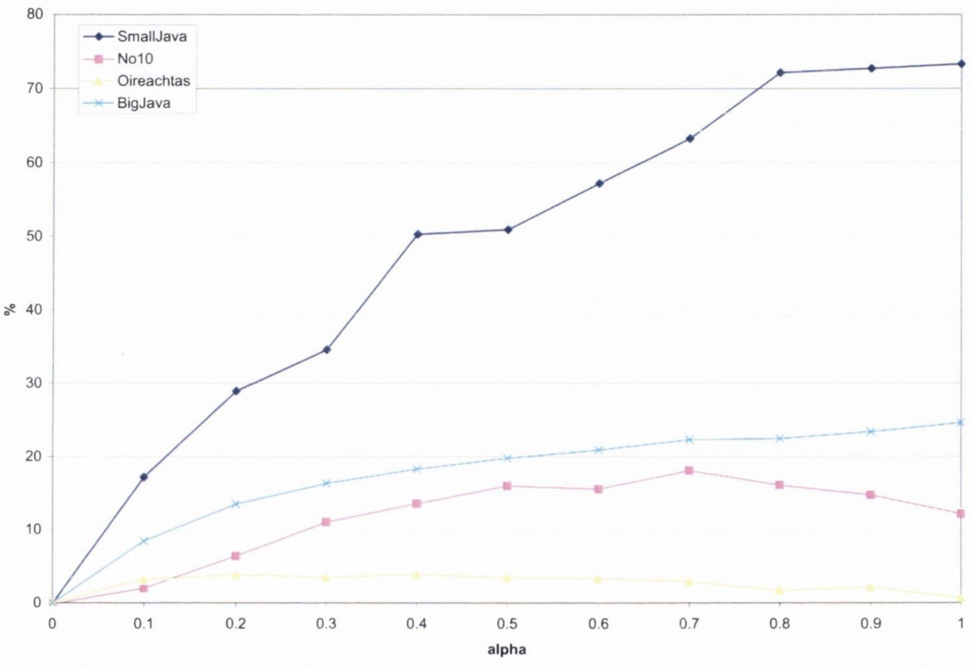


Figure 5.13: Average change in 3-pt precision values over HITS variant benchmark, averaged by test set

vided information that PageRank and HITS did not provide, resulting in more accurate document rankings. The implementations of PageRank and HITS used in these experiments were under some restrictions as described above. Under these constraints, we are unable to evaluate the ability of these algorithms to identify authoritative pages in a large-scale Web environment, but we can compare their effectiveness on a small-scale environment, and determine if the contextual information provided by TF×INF is separate to the information provided by these algorithms. In particular, in each of these corpora, all documents are obtained from the same Web site. The identification of authoritative documents is, therefore, not necessary. However, since the TF×INF ranking scheme outperforms these schemes when applied to Web document collections containing only internal links, we can deduce that the information that TF×INF provides to an IR application is different

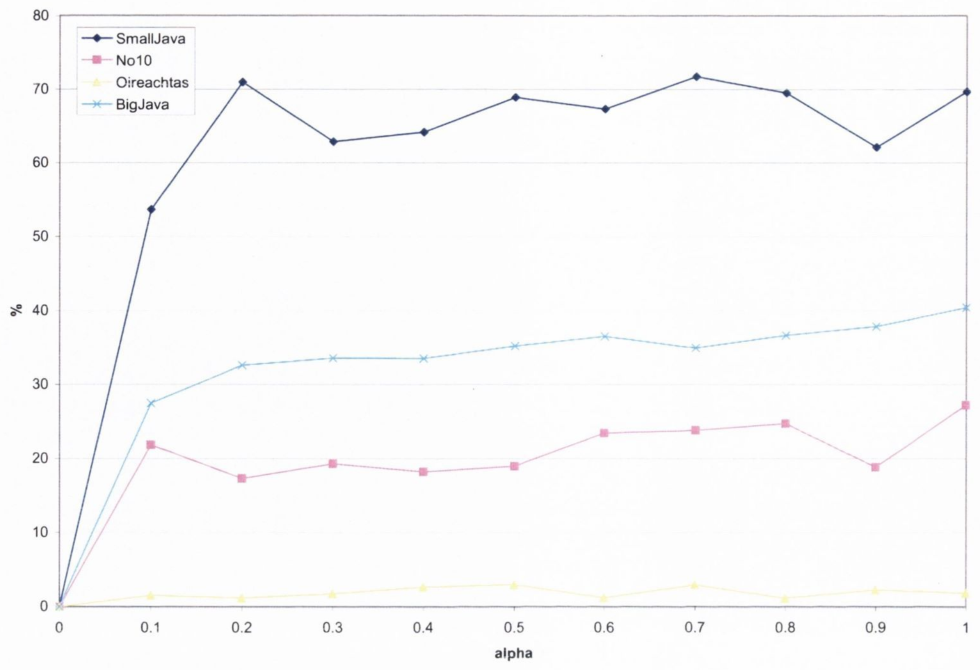


Figure 5.14: Average change in 3-pt precision values over PageRank variant benchmark, averaged by test set

to information about authority that is provided by these algorithms. In a global Web searching algorithm, therefore, TF×INF could operate in conjunction with these applications.

5.5 Discussion

5.5.1 Summary of Results

In this chapter, the TF×INF probabilistic model used in the automatic keyphrase extractor is adapted for use in an information retrieval application. The ranking algorithm reports the probability that a document is relevant to a query by using the model to estimate the probability that the query term is a keyphrase of the document. The algorithm is applied as a post-processor of results obtained from other ranking applications: TF×IDF, TF×IDF with relevance propagation along hyperlinks, and variants of the HITS and PageRank algorithms adapted for use on a corpus obtained from a single Web site.

The algorithm was evaluated on the corpora described in the previous chapter. Queries were obtained from the keyphrase lists associated with documents in these corpora, and document relevance to a query was established if a document contained the query term in its keyphrase list. The ranked result lists obtained from the experiments were evaluated using a 3-point average of the precision-recall curve of the result list.

Two of the four corpora, No10 and SmallJava, performed consistently well under TF×INF, when compared with TF×IDF and TF×IDF with RP, while the other two corpora, Oireachtas and BigJava, performed poorly, with only small improvements over the TF$\times$IDF benchmark. The poor performances of these corpora are explained by the characteristics of the keyphrases used to evaluate the system - in the Oireachtas corpus the keyphrases are too general, and in the BigJava corpus, they are too specific to be accurate representations of document relevance. The generality of the keyphrases is an artifact of the testing environment. In a working environment, keyphrases are not used to evaluate a document's relevance to a query. Tests on the corpora with more suitable keyphrases showed that the TF$\times$INF algorithm can significantly improve TF$\times$IDF-ranked results lists.

The TF$\times$INF algorithm is shown to provide more comprehensive contextual information than the relevance propagation method, which focuses on a particular form of contextual information, in which documents are deemed relevant to a query if their neighbouring documents are also relevant. The algorithm is also shown to provide qualitatively different information to the authority information provided by the link topology analysis methods, PageRank and HITS.

5.5.2 Web Spam and Search Engine Personalisation

The use of VSM methods such as term frequency in Web information retrieval has fallen out of favour due to its susceptibility to manipulation by authors of Web spam documents (Gyongyi and Garcia-Molina, 2005; Metaxas and DeStefano, 2005). In TF$\times$IDF ranking schemes, the criteria for a high rank in a search engine result set for a particular query can be controlled entirely by the author of the document content, by including a query term multiple times, for example. For this reason, search engines that are based on document authority or trust measures, such as PageRank, have had greater success at avoiding manipulation.

Although a TF$\times$INF-based ranking scheme initially appears to be susceptible to the same form of manipulation as TF$\times$IDF, the use of a probabilistic model to identify suitable TF$\times$INF values adds a greater degree of freedom to the ranking application. In this chapter, the probabilistic model was generated by inspecting the TF$\times$INF values of keyphrases and non-keyphrases in training documents, and updating the probability distribution of the keyphrase or non-keyphrase class accordingly. If Web spam documents are incorporated in the training data, the TF$\times$INF values of terms that occur in these documents could heavily influence the probability distributions in order to penalise unseen documents that contain query terms that have similar TF$\times$INF values. An alternative learning scheme would be to add a third category to the model - 'spam term', and annotate all terms in Web spam documents in the training data with this tag. Documents containing a query term that is classified as Web spam may be given a lower rank as a result. Alternatively, instead of simply using the query term to calculate the probability that the document is a spam document, the entire set of terms in the document could be used to calculate this probability. The result is a Web text classifier, that is described in the Future Work section of the conclusions to this thesis (section 8.4).

Since the mechanism used to determine document relevance is a hidden probability model, as opposed to any measure that is extrinsically calculable, this system is more difficult to circumvent and manipulate than a VSM model using a term frequency measure such as TF$\times$IDF. Furthermore, as the probability model can be changed, updated and refined as more training examples, or newer training examples are added, Web spam documents that succeed in circumventing the system may be able to do so only temporarily.

Another advantage of using a probability model in a document ranking application is that it allows a large amount of individual personalisation of search engine behaviour. This is described in more detail in section 8.4.

5.5.3 Recommendations

Based on these observations, we can recommend that TF$\times$INF be included as a component of a multi-part Web search engine, that combines information provided by TF$\times$IDF and link topology analysis measures such as PageRank. The use of TF$\times$INF in this way adds little overhead to the query time cost of existing Web-based search engines, as the TF$\times$INF value of a term in a document can be added to the index off-line, or calculated quickly at query time by inspecting the indexed term frequency values of a document and its neighbouring documents.

Chapter 6

Web Corpus Homogeneity

The growth of the World Wide Web has led corpus linguists to turn their attention towards it as a source of language corpora (Volk, 2002; Kilgarriff and Grefenstette, 2003). The Web has a number of important advantages over traditional corpora, in that it is a vast, freely available resource, and, importantly, is potentially more representative of widespread language use. This difference arises from the unrestricted nature of authorship on the Web. Documents can be written and published, not by a controlled number of authors, but by anyone with the technical ability to upload content. With the development of the modern Web trend of user-generated content, as described in chapter 1, authorship is extended to a yet wider pool of language producers, as access to a Web hosting server is no longer a requisite for publishing on the Web, and the technical knowledge required to upload content has been reduced.

This unrestricted quality of Web authorship allows for a broad range of language variation in terms of subject domain (known as sublanguage (Kittredge, 1982)), register and dialect.[1] This variety is ideal for an information resource from which language corpora are to be generated, due to the added flexibility, but it also poses problems for linguists seeking to generate these corpora. Different applications require different levels of homogeneity with respect to some or all of these variables. Sublanguage homogeneity has been shown to be an important factor in the success of automatic translators (Kittredge, 1985), while Biber (1993) demonstrates that register variation (or heterogeneity in the register of corpora) is important for general corpus-based analyses, and specifically for the development of part-of-speech taggers and syntactic parsers, as differences in the linguistic patterns of registers are averaged, resulting in a generalisation of the resultant model. However, Gildea (2001) and Sekine (1997) show that for domain-specific statistical parsing models, homogeneous training data are much more useful than heterogeneous training data for accurate parsing of unseen data.

Homogeneity is also important in the generation of Language Models. Language Models (Ponte

[1]Dialect here is an umbrella term that includes language variation according to region, as well as idiolects, sociolects etc.

and Croft, 1998) are used in a variety of NLP applications, such as speech recognition and machine translation, as well as in information retrieval. Language Models assign probabilities to n-grams according to their frequencies in a language resource. These probabilities are then used to predict the next word in a sequence (in applications such as speech recognition) or, in information retrieval, to assign retrieval values to a document. The retrieval value of a document is calculated according to the probability that a language model based on the document will generate the query terms. The problem of data sparseness is typically addressed by assigning default probabilities to terms missing from a document according to their frequencies in a corpus containing the document (Ponte and Croft, 1998). The degree to which this default probability is a suitable approximation of the true probability of a term in a document is related to the degree to which the document is representative of the corpus. Documents are more likely to be representative of a corpus if the corpus is homogeneous (Rose et al., 1997). In the case of highly heterogeneous corpora, clustering techniques are often employed in order to identify and make use of homogeneous subsets of the corpus (Kurland and Lee, 2004).

Therefore, the degree to which a corpus should be homogeneous or heterogeneous with respect to sublanguage, register or dialect depends on the purpose to which the corpus will be put. Domain-specific applications are better trained on homogeneous domain-specific corpora, while more generalised applications that are required to be domain-independent require the most heterogeneous corpora possible (i.e. a corpus with large language coverage). In either case, the homogeneity of the corpus should be known.

How can corpus homogeneity be measured on the Web? In fact, issues of language variation and homogeneity are more pronounced on the Web due to its unrestricted nature, and the hyperlink structure of the Web offers a useful device for its measurement and detection. In general, though not always, the presence of a link between two documents suggests a user-specified relationship between them. This relationship is usually seen to be domain-related, i.e. the subject of the two documents may be related, but can also be an indicator of other similarities, such as similarities of language – linked documents are likely to be written in the same language[2] – and, in the case of webloggers linking to related webloggers, there may be a similarity of dialect or register.

This chapter describes how TF×INF can be used to measure the homogeneity of Web corpora. We evaluate this method on a set of corpora generated from the Wikipedia Web site, using an information theory measure based on user-annotated document categories as a gold standard measurement of homogeneity. We then compare our measure with existing link-free measures, described in the following section, on the same corpora.

[2]This excludes the special case of links to translations of the current page.

6.1 Background: Measuring Corpus Homogeneity

Corpus homogeneity has been studied using traditional unstructured text corpora (Kilgarriff, 1997; Kilgarriff and Rose; Rose et al., 1997), but little work has been carried out on the topic in the specific case of hyperlinked corpora such as the Web. Kilgarriff (2001) treats corpus homogeneity as a variant on corpus similarity, using word frequency lists generated from a corpus in order to measure its homogeneity. A corpus is repeatedly split randomly into two subcorpora, and word frequency lists are calculated for each of the subcorpora as well as for the overall corpus. The two subcorpora are then treated as separate corpora and Kilgarriff proposes two measures that can be used to evaluate their similarity.

The first is Spearman's rank correlation coefficient, which tests if there is a correlation between the relative popularity of a term in one corpus and its relative popularity in the other. The relative popularity of a term is determined by its rank in the ordered word frequency list for the corpus. This is calculated for the top n words in a corpus. A coefficient value close to 1 indicates similarity of subcorpora.

The second measure is χ^2, which is used in this case as a non-statistical measure (i.e. without a null hypothesis). χ^2 is calculated as

$$\sum \frac{(O-E)^2}{E} \tag{6.1}$$

for each word in each subcorpus, where O is the observed frequency of the term and E is the expected frequency according to the frequency of the term in the overall corpus. This is normalised by the number of words used in the calculation (n), and this normalised measure is known as CBDF (Chi By Degrees of Freedom). A low CBDF value indicates subcorpus similarity.

These measures are calculated on a series of partitions of a corpus and their average values give an indication of corpus homogeneity.

Cavaglia (2002) uses document similarity measures, instead of corpus similarity measures to calculate homogeneity. The homogeneity of a corpus is the maximum distance, or minimum similarity value, between two documents in the corpus. The document similarity measures used are

- relative entropy, which is a measure for comparing two probability distributions
- χ^2 - this measure is used as above, but the word frequency lists are generated from individual documents as opposed to corpora
- log-likelihood, which gives a better approximation of the binomial distribution than χ^2 for events (words) with frequencies under 5

Cavaglia also validates the hypothesis that text classification on homogeneous corpora requires fewer training examples than classification on heterogeneous corpora.

Neither paper above uses Web data, and they are therefore unable to avail of the link structure available in Web corpora. This work describes a manner in which this structure can be used to measure corpus homogeneity.

6.2 Incorporating hypertext links

This section describes our method for incorporating TF×INF in a measure for corpus homogeneity. We compare TF×IDF distributions generated from the overall corpus and TF×IDF distributions from local subsections of the corpus, in order to gauge the overall cohesiveness of a corpus. The local subsections of the corpus are defined by the link structure of the corpus and the TF×IDF values of terms in these subsections are equal to their TF×INF scores. Similarity between the distributions will suggest homogeneity. In this chapter, the log version of TF×INF is used due to its property of being equal to TF×IDF if the subsection contains the entire document collection.

6.2.1 Application of TF×IDF to homogeneity

In a homogeneous corpus, under the assumption that documents discussing the same topic will tend to contain similar words,[3] the same words are important in the entire corpus. The distribution of the TF×IDF values of each term in each document in the corpus will contain a small number of popular (high frequency), but relatively low values, due to their higher document frequency, resulting in lower IDF. In a heterogeneous corpus, on the other hand, there will be more important words, but each of them will individually be much rarer, as the words that are considered important in one subset of documents will be different to those considered important in another. This will result in a significantly different TF×IDF distribution, with the popular values being more spread out (less frequent) and higher (higher IDF). However, the distributions of TF×IDF in the subject-specific subsets will remain similar to those of a homogeneous corpus. Therefore, in order to measure the homogeneity of a corpus, it suffices to compare the distributions of subject-specific subsets of the corpus with the overall distribution. However, such subsets are not typically available. After all, if it were possible to identify subject-specific subsets of the corpus, this information could be used directly to determine its homogeneity.

The TF×IDF values of the subject-specific subsets will be calculated using TF×INF, as described below.

[3]Topics are used in this context, but the assumption is also true of documents written in the same language (French documents are more likely to be linked to other French documents, for example), or, to a lesser extent, language varieties or sublanguages (officially written documents are more likely to be linked to other officially written documents, e.g. in a government Web site, etc.) This chapter focuses on subject-related differences as a generalisation.

6.2.2 Application of Content Homogeneity to the Web

An important assumption used in this work is that a hyperlink between two documents in a hypertext implies a semantic relation of some kind between them. Certainly, two random documents are more likely to be connected by a hyperlink if they share a subject than if they do not. A collection of documents connected to each other by paths of hyperlinks of length n is therefore a local subnetwork of the graph, with n being an estimator of the degree of semantic 'closeness' of the network. The subject-specific subsets required above are thus implicit in Web corpora. Web homogeneity can therefore be defined as the difference between the distributions of TF$\times$IDF values in local subnetworks (their TF$\times$INF values) and the overall corpus distribution. Nonetheless, a certain amount of 'content drift' can occur when traversing links, to the extent that after traversing enough of these links, users will find themselves in a completely different subject domain. A local subnetwork may be homogeneous; however, the homogeneity of this network will typically decrease as its scope is extended. Therefore, the spread of the subnetwork is a factor in the identification of homogeneity.

6.2.3 TF$\times$INF$_n$ described as the TF$\times$IDF of local subnetworks

In this chapter, we interpret TF$\times$INF to be a function of a subnetwork of documents, rather than explicitly as an indicator of document context. The size of the sub-network is determined by the path length n used to define the extent of the context set in chapter 3. We therefore use the subscript n to determine the specificity of the sub-network at any time.

6.2.4 Comparing TF$\times$IDF Distributions

In order to obtain the data used to generate the feature distributions, the TF$\times$IDF and TF$\times$INF$_n$ values of each term in each document in the corpus are collected. The distributions are discrete univariate distributions and can be compared using a probability-binning χ^2 statistic (Roederer et al., 2001). The distributions are split into ten bins such that the sums of the values of the two distributions are equal for each bin. The ratios of TF$\times$IDF and TF$\times$INF$_n$ values of each bin are used to calculate the χ^2 statistic. This χ^2 statistic is normally distributed when the two data sets under comparison are equivalent. We can therefore generate a significance measure, t, by normalising the distribution. Under the null hypothesis that the two distributions are equal, t will be zero. A low t therefore indicates similarity between the distributions, and a high t indicates that the two distributions are very different. Note that while the t value is derived from statistical analysis, we are using it strictly as a similarity measure, and not as a statistical test.

We use the log form of TF$\times$INF$_n$ due to its similarity to TF$\times$IDF. The log form has the property that as the size of the subnetwork n increases, the TF$\times$INF$_n$ distribution approximates

$\mathrm{TF}\times\mathrm{IDF}$.[4] When each document is considered a neighbour of each other document (i.e. the corpus can be regarded as a fully connected graph), $\mathrm{TF}\times\mathrm{INF}_n$ will equal $\mathrm{TF}\times\mathrm{IDF}$ for each phrase, so the distributions will be equal. The core hypothesis tested in this chapter can therefore be stated as follows:

Hypothesis 1 *The $TF\times INF_n$ distribution of a homogeneous web will, at low values of n, approximate $TF\times IDF$ more than that of a heterogeneous one (t will be lower).*

The size of the subnetwork induced by n will depend on the connectivity of the corpus. If a corpus has low connectivity, it might be necessary to raise the value of n in order to ensure that the $\mathrm{TF}\times\mathrm{INF}_n$ distribution contains sufficient information. The optimum value for n is therefore corpus-specific. As n increases, the computational cost of locating the subnetworks increases. Therefore, the optimum value for n is the lowest value such that the difference between homogeneous and heterogeneous corpora is evident. In our experiments, we find that for a large range of Web types, connectivity is sufficiently high that a value of $n = 1$ is enough to clearly display this difference.

6.3 Results

6.3.1 Testing Environment

In order to test the technique described above, a large set of Web corpora, ideally with labelled homogeneity values, are required. Such corpora are, as yet, rare. Kilgarriff (2001) combines subsets of corpora according to varying proportions in order to generate a set of corpora with known similarity relationships. Web corpora are graph-like structures, however, and combining two such corpora results in bipartite graphs with no links between the two partitions. Such bipartite graphs are not representative of Web corpora, so an experiment using this method is less suitable for analysing Web homogeneity.

A corpus based on the Wikipedia encyclopædia (Denoyer and Gallinari, 2006) exists, however, which contains both hyperlinks and annotated categories that can be used to infer a homogeneity value. The homogeneity identification technique is initially evaluated on this corpus using existing category labels as an extrinsic indicator of homogeneity, as described in section 6.3.2. Subsequent tests were carried out on artificially generated corpora in order to explicitly control the homogeneity of the corpora and to examine the effects of varying corpus parameters such as corpus size and connectivity. The corpora used in previous sections are largely unsuitable for this purpose due to their size and the lack of implicit or explicit homogeneity values. However, some tests were

[4] It is possible to use a different form of the $\mathrm{TF}\times\mathrm{IDF}$ formula to give us the same relationship with the non-log form of $\mathrm{TF}\times\mathrm{INF}$. Tests that were carried out using such a formula gave comparable results to those obtained using the log form, so we use this form here for simplicity

carried out on corpora generated from the Java Web site,[5] by assuming a correlation between homogeneity and, for example, the depth in a directory structure to which a Web document harvester is restricted, or the proportion of links such a Web harvester is permitted to traverse from any given document in order to collect new documents. The Java Web site corpus was useful for this purpose as it is based on a strong hierarchical structure with clear semantic differences between subdirectories. This is in contrast to the other corpora where such differences were less evident.

6.3.2 Tests based on Wikipedia Corpus

The Wikipedia XML corpus (Denoyer and Gallinari, 2006) is a collection of XML documents based on the Wikipedia open content online encyclopædia. The English section of the XML corpus comprises 659,388 documents. These documents are made up of text and links to other documents in the corpus. Almost all of the documents contain one or more subject categories, which are manually annotated by users of the encyclopædia. These categories are used to provide homogeneity values for document sets that are then compared to t values obtained by the method of comparing TF×IDF distributions described in section 6.2.4. Evidence of correlation would suggest that the method described in this paper is an accurate indicator of homogeneity.

In order to carry out this test, subcorpora of varying sizes are extracted from the Wikipedia corpus by following links from a random start document. The category-based homogeneity of each subcorpus can be regarded as a function of the combined generalities of the categories, in terms of the proportions of documents filed under each category. As a simple example, consider corpora consisting of documents of two distinct independent categories, A and B. One such corpus with, say, 90% of documents filed under category A and the remaining 10% filed under B should be considered more homogeneous than a corpus containing a uniform distribution of documents under A and B. One can therefore express a notion of *corpus heterogeneity* in terms of the entropy of the probability mass function of the category random variable C:

$$H(C) = -\sum_{c} p(c) \log p(c) \tag{6.2}$$

where $p(c) = Pr(C = c)$. Since the test corpus is annotated for category information, the probabilities can be obtained by maximum likelihood estimation, adjusting the number of documents for occurrences of multiple labels.

With an extrinsic criterion for homogeneity thus established, the t values for TF×IDF distributions of these subcorpora are then calculated and the correlation between the two measures can be assessed in order to determine the effectiveness of the technique.

[5] `http://java.sun.com`

Table 6.1: Results of multiple regression on Wikipedia data – variable: log(t)

Variable	Coefficient	SE	t	p-value
Intercept	2.32	0.077	30.31	1.7E-32
log Corpus Size	0.066	3.999	15.54	2.0E-4
Cat. Ent.	0.219	0.028	7.701	7.1E-10

The t values were found to have a strong relation on the logarithmic scale to the category entropy values. After applying the logarithm function to the t values, the coefficient of determination, R^2 (for 88 data points), was found to be 87%, indicating a strong positive correlation between t values[6] and category entropy. The corpus size is varied in this experiment, and has an effect on both the t variable and the category entropy variable. By carrying out multiple linear regression analysis in order to separate the effect of corpus size on t from the effect of category entropy (see table 6.1,[7]) it was found that category entropy has a strong positive effect on log(t), as the null hypothesis that category entropy is not related to log(t) was rejected with high probability ($p < 10^{-9}$). As the number and generality of categories increase, the category entropy and t values increase, indicating higher heterogeneity.

6.3.3 Comparison with Other Homogeneity Measures

Two alternative, link-free measures for corpus homogeneity used by Kilgarriff (2001), the Spearman rank correlation coefficient and CBDF, were described in section 6.1. Similar tests against category entropy were carried out with these measures on the Wikipedia corpus, and the results are shown in tables 6.2 and 6.3.

In the Spearman test (table 6.2), the category entropy variable was also shown to have a larger p value, which suggests that while the null hypothesis that category entropy is unrelated to the Spearman measure can still be confidently rejected (with a statistical significance level of over 99.999%), it cannot be rejected with as much confidence as the test against log(t).

Likewise in the CBDF test (table 6.3), category entropy was shown to be significantly related to the CBDF measure (significance level of over 99%), but this significance level is much weaker than in the test against log(t).

As both p values are statistically significant, this is a somewhat unreliable method of comparing the techniques. However, it suggests that the t value measure performs at least as well and possibly better than these link-free measures in calculating the homogeneity of Web corpora.

[6]For the maximum subnetwork path length, n, the value of 1 was used for this test, as it gave the strongest results.

[7]In this table, SE = standard error, t = Student's t-statistic, p = probability of result under null hypothesis. Note - Student's t statistic here is a function of the standard error of the mean and is unrelated to the t value described in this paper

Table 6.2: Results of multiple regression on Wikipedia data – variable: Spearman / term count

Variable	Coefficient	SE	t	p-value
Intercept	0.001	1.6E-4	7.841	1.27E-10
Corpus Size	-6.4E-8	2.26E-8	-2.83	6.29E-3
Cat. Ent.	-1.4E-4	2.53E-5	-5.35	1.6E-6

Table 6.3: Results of multiple regression on Wikipedia data – variable: CBDF – 20 degrees of freedom

Variable	Coefficient	SE	t	p-value
Intercept	-162.91	91.19	-1.79	0.079
Corpus Size	-0.02	0.013	-1.65	0.104
Cat. Ent.	38.71	14.5	2.67	9.87E-3

6.3.4 Applying link information to other homogeneity measures

This method uses extra information, the hyperlink structure of the Web, that is not available to the homogeneity measures described by Kilgarriff (2001). In order to more fairly compare the performance of these measures with the one proposed in this chapter, the Spearman rank correlation coefficient was adapted to incorporate this information. The adaptation is unrelated to TF×INF, and therefore allows us to determine the usefulness of TF×INF in this task by comparing it like-for-like with another hypertext-based method. The methods described in section 6.1 randomly partition the dataset and compare the term frequency rankings between the two partitions. One advantage of the hyperlink structure is that it may allow us to identify partitions of the dataset that are more divergent than those obtained randomly. Under the assumption used in this work that hyperlinks typically connect related content, such a divergent partition can be generated by placing closely-linked documents in the same partition and unlinked documents in different partitions. We use a single-link clustering algorithm to group documents into clusters according to the hyperlinks between them. The algorithm is instructed to generate a set of clusters, which are then added to two partitions. The number of clusters generated by the algorithm and order in which they are added to the partitions are varied in order to create a set of partitions that are used to calculate the Spearman coefficients. The means of these coefficients are then calculated as before.

The clustering algorithm

A symmetrical adjacency matrix M is generated from the hypertext graph. Each zero-valued entry in this matrix is set to a value indicating positive infinity. This matrix is then converted into a document distance matrix according to shortest path length between documents. This is carried out by iteratively setting each new value $M'_{i,j}$ equal to the minimum of:

$$M_{i,j}, 1 + \min\left(M_{k,i}, M_{i,k}, M_{k,j}, M_{j,k}\right) \tag{6.3}$$

for all documents k connected by a hyperlink to i or j.

This process is iterated for twenty steps, or until no values are changed by the above method, resulting in a distance matrix that can measure path lengths of up to twenty hyperlinks. If a pair of documents with a shortest path length larger than twenty hyperlinks exists in the collection, their distance is considered to be infinite in order to reduce the computational time required to run the algorithm. The subcorpora we generate from the Wikipedia corpus are obtained by spreading from a central location in the corpus. The resultant corpora are sufficiently connected that few documents are more than six hyperlinks from each other. More sparsely connected collections may require a higher cut-off point.

The single-link clustering algorithm is initialised by generating a set C of clusters, each made up of a single document in the collection. A symmetric cluster distance matrix M is maintained throughout the procedure, and this is initialised from the document distance matrix described above. At each stage of the procedure, the two closest clusters c_i, c_j are chosen by locating the smallest minimum distance between the documents in the cluster - the smallest value in the cluster distance matrix. These clusters are then combined and the resultant larger cluster replaces them in the cluster set. M is then updated with the rows and columns representing the removed clusters replaced with a single row and a single column containing the minimum values contained in the removed rows and columns. This can be expressed as:

$$M'_{n,x} = \min\left(M_{i,x}, M_{j,x}\right) \tag{6.4}$$

for all $0 \leq x < |\mathrm{C}| - 1$
where n is the index of the new cluster c_n.

This process is repeated until the required number of clusters remain. These clusters are ordered by increasing size. Variation in this section of the partitioning procedure is achieved by randomising the order of equal-sized clusters. Two empty subsets of the document collection are initialised. The documents in each cluster are then added in turn to the smallest subset. If a cluster is too large to be added to a subset without that subset containing more than half of the total documents in the collection (or one more than half, if the collection has an odd number of documents), that cluster is split into its two sub-clusters and these clusters are returned to the ordered list of remaining clusters. Once all clusters have been added to the subset, they are stored as a partition of the document collection. New partitions are generated by re-randomising the order of equal-sized clusters in the cluster list. The Spearman rank correlation coefficients of each partition are then calculated.

Limitations to this technique

The objective of the standard Spearman method is to identify heterogeneity in document collections by generating a number of random partitions of the document collection and calculating the

Spearman coefficient for each, on the understanding that the heterogeneity of a collection will be reflected on average in the dissimilarity of the partitions. The hypertext-based method described above potentially increases the likelihood that a given partition will reflect the heterogeneity of the collection, but the clustering algorithm reduces the number of random partitions that can be generated from the collection. By reducing the number of clusters generated by the method, the number of possible partitions of the collection can be increased, but these partitions are less influenced by the hypertext, and are therefore potentially less divergent.

At the extremes, if the number of clusters is equal to the number of documents, the method is equivalent to the non-hyperlinked version, and if the number of clusters is equal to two, only one partition of the collection will be possible, and will be based fully on the hyperlink structure. It is unlikely that the heterogeneity of the collection is reflected sufficiently strongly in the hyperlink structure that it would be identifiable in the Spearman rank correlation coefficient of one partition.

In contrast, the TF$\times$INF method described in this chapter does not treat the document content explicitly, but compares the global properties of the collection (the TF$\times$IDF distribution) with its local properties (the TF$\times$INF distribution). This removes the need to partition the dataset, while nonetheless incorporating the hyperlink structure.

Results

We carried out experiments on the same datasets as those used to evaluate the other homogeneity measures, but varied the number of clusters generated using the single-link method. This gave us a set of Spearman coefficient values for each dataset, which we then compared with the category entropy value using multiple regression as above. The results of the experiments are summarised in figure 6.1, with the result from the original non-clustered Spearman algorithm included for comparison purposes. The p-values are comparable to the non-clustering algorithm in general, with greater statistical significance reported when two or three clusters are generated. An improvement is also reported when more than sixteen clusters are generated. Between these extremes, the clustering method appears to give no significant improvement over the non-clustering Spearman method, and the significance levels reported at the extremes are of a lesser degree than those obtained using the TF$\times$INF-based method. We can therefore conclude that TF$\times$INF is a more effective technique than clustering for the inclusion of hypertextual information in a corpus homogeneity measure.

6.3.5 Tests based on Java Web corpora

A limited number of tests were carried out on collections based on the Java Web site. The collections were obtained using a Web spider program to extract the text and hyperlinks from a start page in the site. External links were discarded and the remaining links were used to locate and extract further pages. The following two tests were carried out on these corpora:

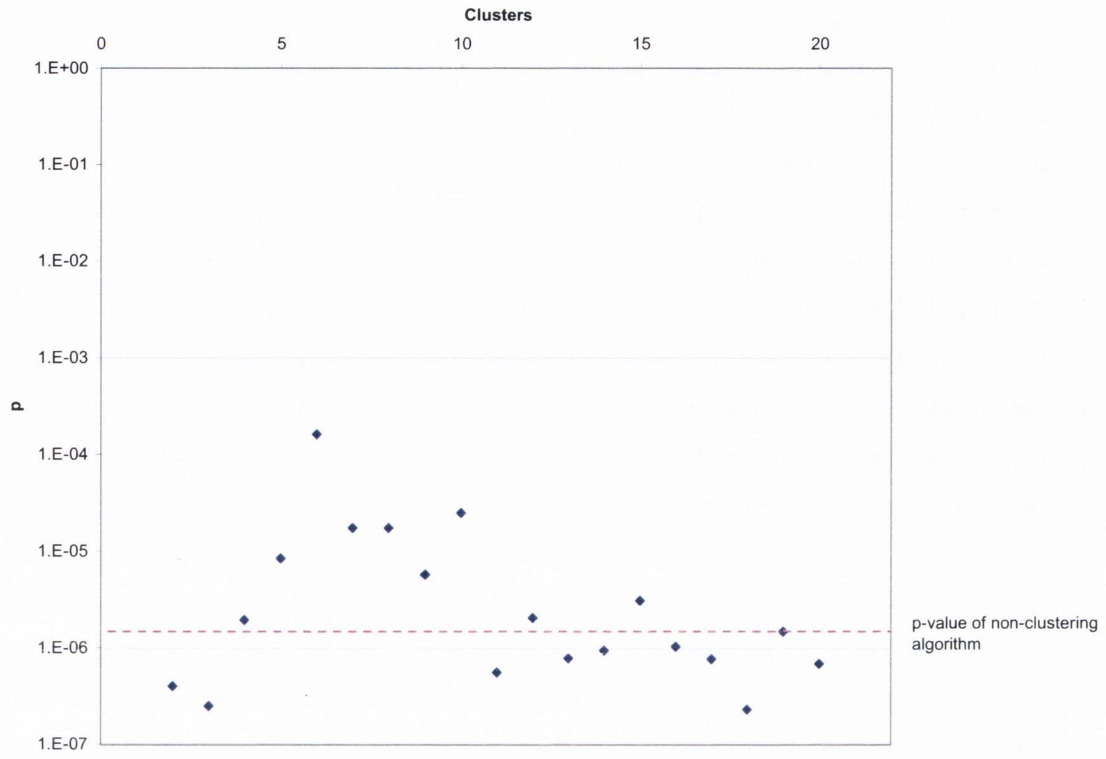


Figure 6.1: p-values of Category Entropy variable in regression analysis against Spearman rank correlation coefficient using clustering method

Directory restriction

When generating the corpora for this experiment, the links traversed by the Web spider were restricted to certain subdirectories of the Java Web site. Assuming the content of a subdirectory is more homogeneous than the content of the parent directory (including the subdirectory), the t values for the subdirectory should be lower. In this test, two sets of ten corpora were generated. The first set was generated by allowing the spider to extract pages from anywhere in the Java Web site (starting at `http://java.sun.com`) and the second set restricted the spider to documents in the Java tutorial[8] section of the Web site. The assumption therefore was that the corpora based on the tutorial Web site would be more homogeneous than those based on the general site. If this assumption is correct, the t value should be lower for the Java tutorial corpora than for the general corpora, and this is the result that was obtained. Figure 6.2 shows the comparison of the t values, for $n = 1$, with 95% confidence intervals shown.

Connectivity restriction

In this test, the spider was restricted to following only a proportion of the outgoing links from any given document. This required the spider to travel further from the start page in order to collect the same amount of documents, thus increasing the average path length between documents in

[8]http://java.sun.com/docs/books/tutorial/

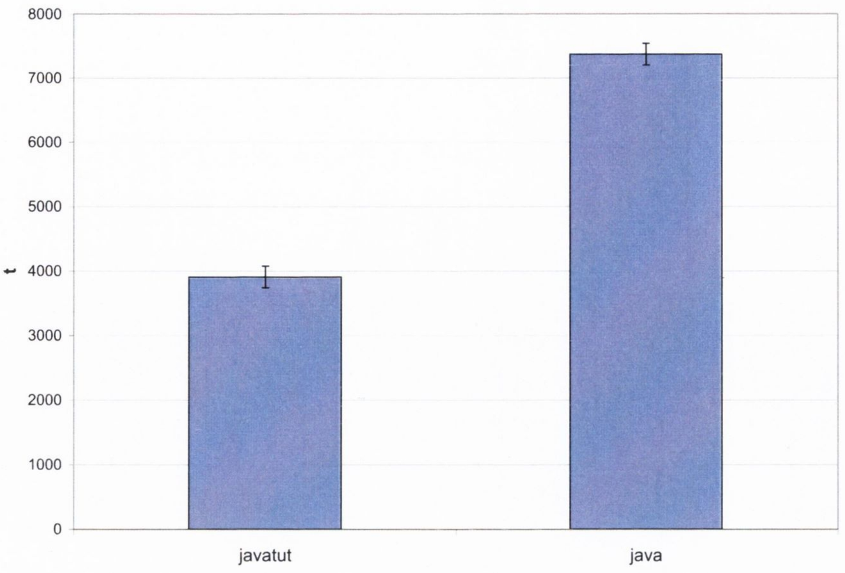


Figure 6.2: Comparison of t for Java and Java Tutorial corpora.

the resultant corpus. According to the assumption of relatedness of hyperlinked documents, this would result in a more heterogeneous corpus. Web spiders following 100%, 50% and 20% of the links from any site generated 10 corpora each. The tests were again applied to the Java Web site. The number of links in the resultant corpora were normalised (by randomly trimming links from highly connected corpora) in order to retain a constant level of connectivity across corpora. Figure 6.3 shows the results of the test, with 95% confidence intervals. As can be seen, the average t values were indeed lower for corpora with higher link proportions, i.e. lower average path length between the documents. However, as the link proportion decreases, the variation increases, as the effect of randomly choosing certain links over others becomes more prominent. This is shown by the large confidence intervals in the results. As a result, the outcome of this experiment is not as conclusive as others.

6.3.6 Tests based on Synthetic Corpora

In order to examine the effects of global properties of Web corpora on the t values and obtain a gold standard homogeneity value that can be compared with the value calculated by the χ^2 test, we have automatically generated several corpora. These "synthetic corpora" are hyperlinked structures parametrised in terms of *corpus size*, *connectivity*, *number of topics* and *probability of topic to topic links over topic-unrelated links*.

The corpora were designed to be an accurate simulation of a Web corpus, and to that end, the distribution of the in-degrees and out-degrees of the documents tend to follow power laws (Faloutsos et al., 1999; Barabasi and Albert, 1999).[9] The connectivity of a corpus is therefore

[9]This is a general rule based on the scale-free nature of the Web, but not all Web collections follow it. The same tests were carried out using uniform distributions of in and out-degrees with equivalent results. The methodology

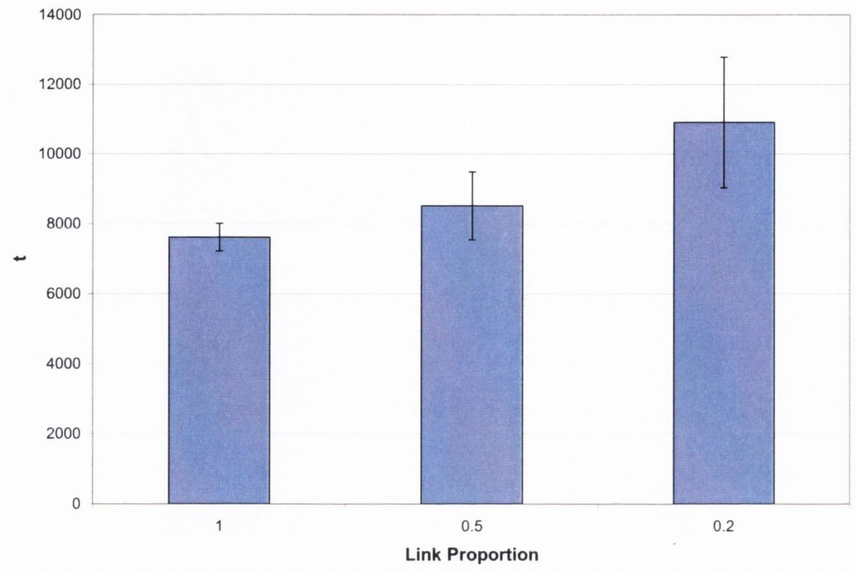


Figure 6.3: Comparison of t for Java corpora collected by traversing different proportions of links.

expressed as two exponents, one for the in-degree and one for the out-degree, as opposed to mean values. Each document in a corpus therefore contains one or more links to other documents, and one or more topic. The topics are selected from a finite set, and each document is assigned one or more topics with uniform probability. A finite vocabulary of artificial phrases is generated, and a 2D matrix associates each phrase-topic pair with a uniformly distributed probability value. The content of a document is then made up of phrases randomly chosen, according to Zipf's law,[10] from the vocabulary, weighted by the probabilities in the matrix for the topic of that document. The generated corpus then has the following properties:

- Each phrase can appear in each document, however, the content of documents tend to be related to its topic.
- A phrase can be associated with any number of topics to any degree.
- The homogeneity of the corpus can be controlled by changing the number of topics available to the documents. In this way, the number of topics acts as a gold standard homogeneity measure.

The hypothesis to be tested is that, for low values of n, the t value of corpora with one or few topics, or high probability of unrelated document linkage, will be lower than the t values of corpora that contain more topics, or have lower probability of unrelated document linkage.

This hypothesis predicts two distinct results:

- The number of topics has a positive effect on the value of t.

presented above does not rely on the power law distributions of hyperlinks

[10]Zipf's law states roughly that terms in a corpus are distributed such that a few words are the most frequent, while most are infrequent.

- The probability that two documents with different topics are linked has a negative effect on the value of t.

These two effects are tested and reported separately. Note that the first effect follows from the intuitive definition of content homogeneity. The second effect can be interpreted as random homogeneity. If this probability is high, the link structure of the Web is not related to the content of the documents as identified by the terms therein. In other words, the assumption mentioned above that linked documents are, for the most part, semantically related, is violated. This can occur in isolated cases, such as alphabetical or chronological directories of unrelated subjects, and it is sensible to consider such webs to be nominally homogeneous, with no topic, as opposed to heterogeneous corpora, in which the topics covered by a document are the factors involved in determining the documents to which it is linked. In the general case, Web pages are linked to topically-related documents, so this is not expected to be a factor. The model allows for such situations, however, by controlling the percentage of links that are assigned randomly, regardless of topic.

Choosing Parameters

Connectivity: In- and Out-degree

The literature describing the power law distributions of the Web offer the empirical values of the exponents of the in and out-degree distributions shown in Table 6.4 (Faloutsos et al., 1999; Barabasi and Albert, 1999; Donato et al., 2004; Litvak et al., 2006; Kogias and Anagnostopoulos, 2003; Kleinberg et al., 1999; Broder et al., 2000; Bharat et al., 2001; Barabasi et al., 2000; Adamic and Huberman, 2002). The chosen values, in Zipf form, were 1 for in-degree and 0.8 for out-degree as they were the approximate average of the reported values. The distributions were implemented in Zipf form (Adamic) in order to facilitate sampling of the distribution — the Zipf distribution is based on frequency rank rather than frequency itself. The out-degrees of the pages are calculated first, initialised by giving the page with the lowest rank exactly one outlink. After the out-degrees have been calculated, they are randomly assigned to pages according to the in-degree distribution.

Subnetwork size n

Early results showed that differences between the TF$\times$IDF and TF$\times$INF$_n$ distributions were most evident for values of n less than 5. The exact value depends on the connectivity of the graph. As expected, increasing n tends to reduce the difference between the TF$\times$IDF and TF$\times$INF$_n$ distributions, and for the vast majority of cases, a significant level of homogeneity was evident when $n = 1$. This allowed us to restrict our later tests to $n = 1$, avoiding the computational cost of calculating TF$\times$INF$_n$ for higher values of n. This is also in keeping with results in previous sections which show that TF$\times$INF is most effective on context sets defined by a path length of

Table 6.4: Exponent values for in and out-degree distributions reported in the literature

In-degree		Out-degree	
Power Law	Zipf	Power Law	Zipf
2.1	0.91	-	-
2.1	0.91	-	-
1.94		-	-
2.09	0.92	2.72	0.58
2.1	0.91	2.45	0.68
2.1	0.91	2.45	0.68
1.73	1.37	1.7	1.43
1.8	1.25	1.8	1.25
1.62	1.61	1.67	1.49
2	1	2	1
-	-	1.23	.81
-	-	2.21	.82
-	-	2.35	.74

$n = 1$.

Corpus size

The tests were carried out on corpora of up to 10,000 documents each. Each document contained up to fifty words.

Comparing t for corpora with 1 or 10 topics

Figure 6.4 shows the average values (with 95% confidence intervals) of t for 10 corpora of size 10,000, 5 with 1 topic (homogeneous) and 5 with 10 topics (heterogeneous). The chart shows a clear difference between the t values of homogeneous and heterogeneous corpora for $n = 1$.

Figure 6.5 shows the effect of changing n. The greatest difference between the homogeneous and heterogeneous averages are at $n = 1$ and $n = 2$. As n increases, the difference between the averages drop off, as expected, and at $n = 4$ the differences are negligible. These results validate the hypothesis above that the number of topics has a positive effect on the value of t. Similar results were observed for different corpus sizes that also validate the hypothesis.

Effect of relation of links to document content

The second part of the hypothesis above is that t will decrease as the probability that unrelated documents are linked increases. In order to validate this hypothesis, we conducted an experiment to gauge the average difference between t values of pairs of related corpora.

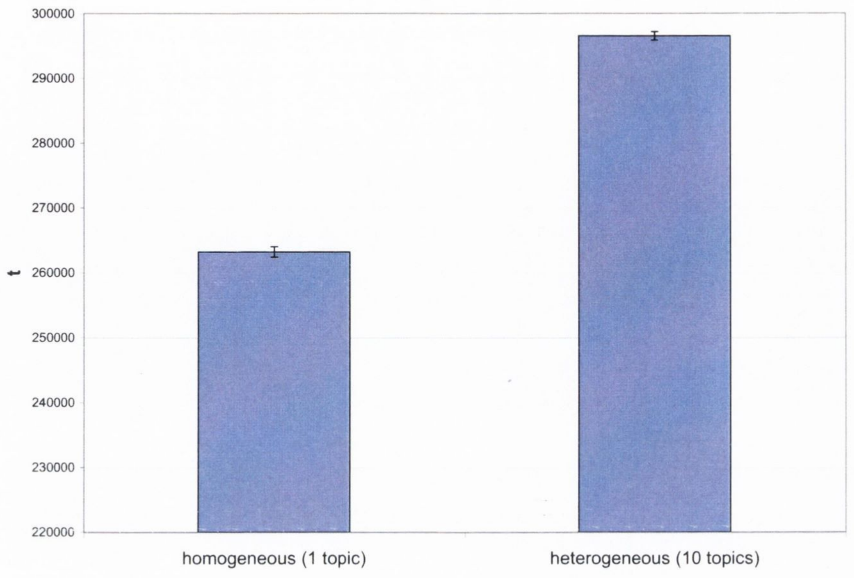


Figure 6.4: Comparison of t for heterogeneous and homogeneous Webs.

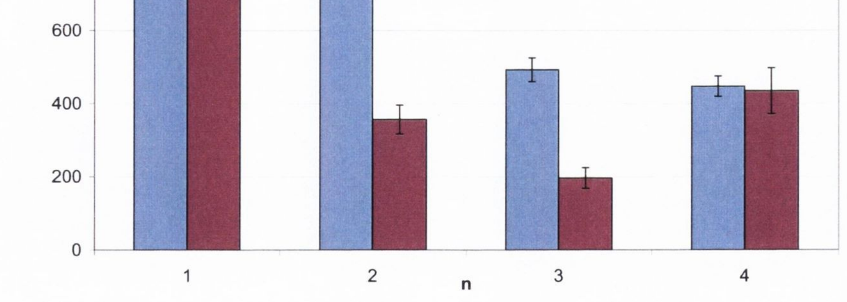


Figure 6.5: Effect of increasing n on t.

A corpus pair is made up of a corpus A and a corpus B. These corpora contain the same documents and vary only by their link structure.

In corpus A, 90% of all links between documents connect documents that share a topic. The remaining 10% of the links are generated irrespective of topic; that is, every document has an equal chance of being the target of a link.

In corpus B, the links are reassigned so that the topic of the documents are ignored; 100% of the links are topic-free, and every document has an equal chance of being the target of the link, in every case. The hypothesis is validated by figure 6.6, which charts the average difference between the t values of the corpus pairs. Figure 6.6 shows that the difference in t values is significant ($p <$

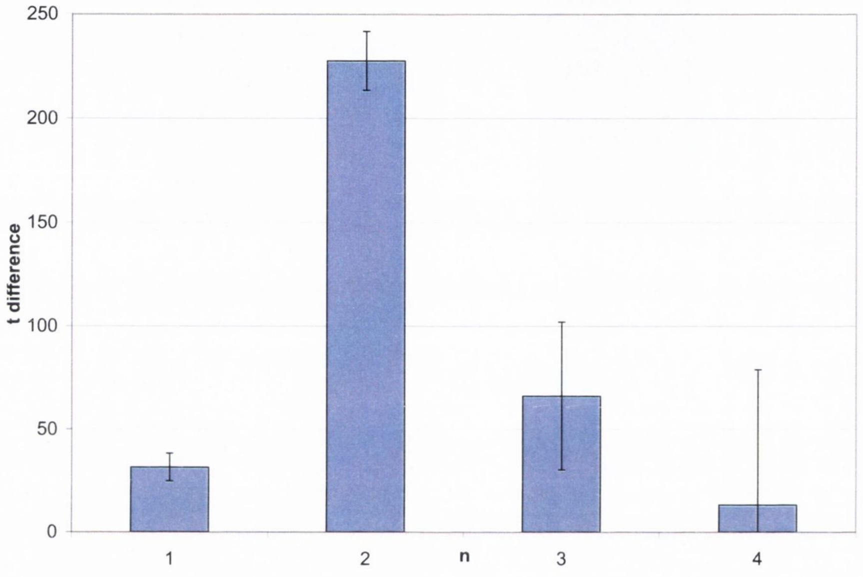


Figure 6.6: Differences between t values caused by link-topic relation.

0.05) for n = 1, 2 and 3. Again, similar results were observed for other corpus sizes.

Relationship of t to corpus size and connectivity

The t values were found to be proportional to the corpus size to the power of an exponent v:

$$t \propto s^v$$

where s is the corpus size. This is a result of the χ^2 operation employed to calculate the t value and was detected both on synthetic corpora and real Web corpora, although the exponent values vary. When comparing the homogeneity of corpora of different sizes, therefore, a normalising technique should be carried out, such as analysing the slope of the relationship between log t and log corpus size (assuming equivalent intercepts).

A weaker linear relationship exists between the connectivity of the graph and log t. These two relationships are shown in figure 6.7.

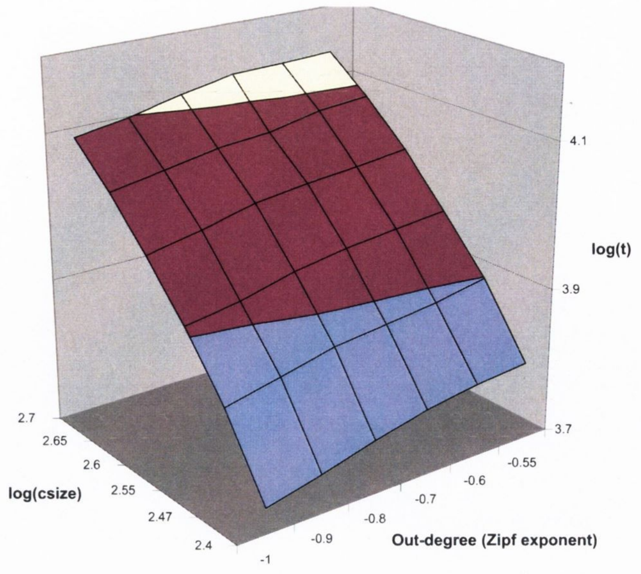


Figure 6.7: t against corpus size & connectivity. Connectivity is displayed as an exponent of the Zipf law used to calculate document out-degree

6.3.7 Summary of Results

In the absence of explicit homogeneity gold standards for Web corpora, we validated our technique using several different indicators of homogeneity :-

- A measurement of the entropy of manually annotated categories in Wikipedia articles
- The directory structure of the Java Web site
- The depth of a document harvester operating on the Java Web site, controlled by the proportion of links followed from any given site
- Direct control of document content using synthetic corpora

We found our measure to accurately identify corpus homogeneity according to each of these indicators.

Existing work on the subject of corpus homogeneity proposed the use of the Spearman rank correlation coefficient and CBDF (Chi By Degrees of Freedom) to measure nonlinked corpus homogeneity. We compared these measures with our technique using the category entropy measurement on Wikipedia, and evidence was found that our technique was more suitable for Web collections than both of these measures. Given the reported connection between content homogeneity and NLP application accuracy, the relationship between functional performance of these applications and this measure for homogeneity would be of interest. Future work will involve designing experiments to investigate this relationship.

6.4 Conclusions

We presented a method for applying TF$\times$INF to the identification of content homogeneity in Web corpora. The method uses a probability-binning χ^2 statistic calculation to compare the TF$\times$IDF distributions of the overall corpus with distributions of subsets of the corpus defined by topic. The link structure of the Web, along with the assumption that connected Web documents are likely to share a topic, are used to define these subsets, allowing us to define the TF$\times$IDF values of local subsets as their TF$\times$INF values. The technique was tested on two Web corpora as well as synthetic corpora. These tests enabled the investigation of alternative extrinsic homogeneity criteria: the Wikipedia corpus allowed evaluation against a category membership criterion, the Java corpus covered homogeneity as reflected by organisation of content, and the synthetic corpus allowed detailed study of a number of parameters. The technique was shown to be capable of identifying homogeneous or heterogeneous corpora, and favourable results were reported in a comparison between this technique and existing corpus homogeneity measures that do not use hyperlink information.

Chapter 7

Homogeneous Web Corpus Generation

In this section we present initial findings on the use of the Web-based homogeneity measure described in the previous section to generate homogeneous corpora from unstructured Web documents. We describe an iterative procedure that filters hyperlinks according to their impact on the homogeneity of the corpus when included. The procedure is tested on the Wikipedia Web site used in the previous section, and was found to significantly reduce the heterogeneity of the generated corpora, albeit at a substantial cost in terms of computation time.

7.1 Introduction - Corpus Generation on the Web

As described in the previous section, the homogeneity of a corpus is an important parameter for a large number of natural language processing applications, such as speech recognition and machine translation, that use corpora as training data. Depending on the application domain, linguists may require highly heterogeneous or homogeneous corpora as training data. Part-of-speech taggers and syntactic parsers, for example, require heterogeneous corpora (Biber, 1993), while domain-specific statistical parsing models and language models often require homogeneous training data (Gildea, 2001; Sekine, 1997; Rose et al., 1997). There is therefore a need for linguists to be able to control the homogeneity of a corpus that they are using, and, in the case of corpus generation, that is being generated.

In the past, corpus generation was difficult and expensive, due in part to the lack of easily available source material. The Web provides a valuable data resource for the creation of corpora; however, its semi-structured and unsupervised nature means that it is often difficult to ensure the content homogeneity of the contents of collections obtained from the Web. The Web is too large (and new documents are being added too regularly) to allow the comprehensive examination of all documents without incurring prohibitive costs.

Another problem associated with the generation of Web corpora is that the semantic roles of Web links are varied. Hyperlinks have a variety of semantic purposes, including subject specialisation or generalisation, or direction to more loosely or tightly related topics, as well as direction to semantically unrelated pages (such as the "this page is best viewed using Mozilla Firefox" variety of link). As a result, links cannot be relied upon to consistently point to related content to a constant degree. A supplemental restriction on the types of links traversed would allow greater control over the homogeneity of the generated corpora.

A sensible strategy would be to restrict the spreading to a specific URL prefix or domain. For example, a corpus containing governmental documents may be obtained by restricting the spread to URLs containing the .gov domain suffix. However, the Web is moving towards a paradigm of user-driven content and distributed Webs (see www.blogger.com, Wikipedia, MySpace) and Web hosts no longer necessarily determine the content of their sites. As a result, the URL directory or domain structure is less reliable as an indicator of content. A more robust method would be based solely on the content of the links, as opposed to the URL structure.

In the previous section, we provided and evaluated a Web-based homogeneity measure. In this chapter, we apply this measure to an automatic corpus generation system for the Web. This method does not use domain names or URL directory structures to choose documents for a corpus, and can therefore be applied to any domain or cross-domain resource, regardless of its structure. This chapter proposes a method of ranking hyperlinks at any stage in the corpus generation according to the degree to which they will affect the homogeneity of the overall corpus if added. A homogeneous corpus can therefore be obtained by adding only those links that have the lowest effect.

7.2 Method

In this section, we use the t-value measure described in the previous section to calculate the homogeneity of several prospective intermediary corpora, each one being the result of adding a different prospective link to an existing corpus. Each intermediary corpus is then ranked according to the value produced by this metric. The top r documents are then added to the corpus, and links from these documents, if not already present in the corpus or the prospective list, are added to the list of prospective documents. This process is repeated until the desired corpus size is attained.

The value of r above controls the finesse of the procedure. A value of $r = 1$ will mean that only the document that increases the heterogeneity of the corpus the least will be added at each iteration. At lower values of r, the resultant corpus is more likely to include documents that are further away from the start document, or documents (i.e. the hyperlink path is longer), since there may be closer documents that have a large effect on the homogeneity of the corpus and are therefore never added. The hypothesis tested in this chapter is that, despite this, doing so will result in a significantly more homogeneous corpus overall, than if documents are added indiscriminately (i.e. using a maximum value for r). The advantage of this must then be weighed against the fact that increasing r will result in fewer iterations of the procedure, greatly speeding up its operation.

To begin the procedure, an initial list of prospective documents is required. This is obtained by initialising the corpus with a seed of s documents. To ensure a reasonably homogeneous initial seed, these documents should be connected in the Web graph.

An initial list of prospective documents from which to begin the procedure can be obtained from a single seed document (assuming that this document contains at least one hyperlink), i.e. $s = 1$. However, the values obtained from the homogeneity measure are essentially meaningless when the corpus size is small, so the seed size can be increased without a significant loss of performance. This reduces slightly the operation time of the procedure. In the tests carried out here, the seed size was set to ten.

7.3 Validation

We test the method described above using the Wikipedia corpus described in chapter 6. The objective of the tests was to generate subcorpora from this corpus by spreading from a starting document, either by following all links indiscriminately from this start point, until the required subcorpus size is obtained, or by ranking links using the method described above. The expected results are that the link ranking method described here will produce lower t-values, on average, and therefore more homogeneous corpora, according to the correlation between t-values and the category entropy described in the previous chapter.

The corpus generation procedure was carried out by selecting an initial document at random from this corpus, and then spreading from this document across the internal links to collect an initial seed of ten documents. The links from this seed were then collected and set as the initial list of prospective documents to add to the corpus. Because the homogeneity measure is sensitive to document size, the document sizes were normalised by considering only the first 80 terms in each document when calculating the homogeneity values. Using this method, subcorpora of 500 documents were created at varying values of r. The purpose of the tests was to discover the degree to which varying r affected the homogeneity of the resultant corpus according to t.

7.4 Results

Figure 7.1 shows average homogeneity values for corpora generated under varying values of r as a proportion of the homogeneity values of corpora generated when all prospective documents are added to the corpus at each step. When all documents are added, no weighting according to homogeneity is taken into account, and this is taken to be the benchmark against which the

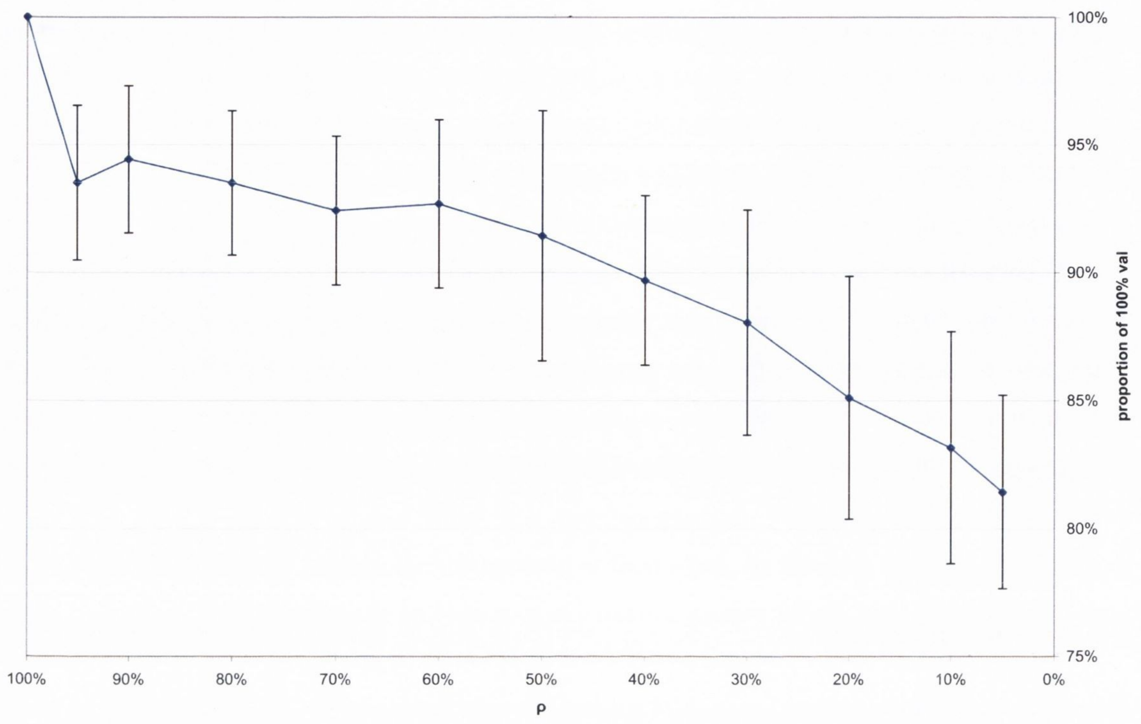


Figure 7.1: Average t-values of generated corpora as a proportion of those generated when r=100%

method can be compared. Here, the value of r for each test is calculated as follows:

$$r = \rho \cdot |L| \tag{7.1}$$

where L is the list of prospective documents, and ρ is a proportion value, between 0 and 1. Figure 7.1 shows the relationship between ρ, on the X axis, and the degree to which the procedure reduces t, or the heterogeneity of the resultant corpora (using the same seed), on the Y axis. When $\rho = 1$, the homogeneity measure was not used - all documents in the prospective list were added at each stage of the iteration. The tests were carried out on thirteen different seeds, and for 12 different values of ρ between 0 and 1. The vertical bars show significance levels with 95% confidence.

Performing this procedure for any value of $\rho < 0.95$ gives a significant improvement in homogeneity. This suggests that the method is capable of producing homogeneous corpora. As ρ gets smaller, the amount of documents added at each stage is reduced, and the t value decreases, indicating higher homogeneity. However, as shown in figure 7.2, the computation time for each stage of the algorithm also increases at a linear rate at each stage of the iteration. As more documents are added to the corpus, the list of the prospective documents grows, resulting in an increase in computation time to recalculate the homogeneity of the corpus for each prospective document. This results in a polynomial relationship between the intended corpus size and the time taken to generate that corpus. Reducing ρ produces, on average, more homogeneous corpora, however, the confidence intervals shown on the chart are wide, indicating that a using a low ρ value will not necessarily produce a more homogeneous corpus in all cases.

By increasing ρ (and therefore increasing r), the computation time is reduced. There is therefore a trade-off between the acceptable computation time, and the control of the homogeneity of the resultant corpus. However, it can be seen from figure 7.1 that the most dramatic reduction in heterogeneity (a drop of roughly 6%) can be achieved by setting ρ to 0.95. This essentially removes the most damaging 5% of links from the corpora, at a reasonably low increase in computation time. This may be acceptable for some corpus generation tasks, and is still a significant improvement from naïve spreading across the Web graph, following all links.

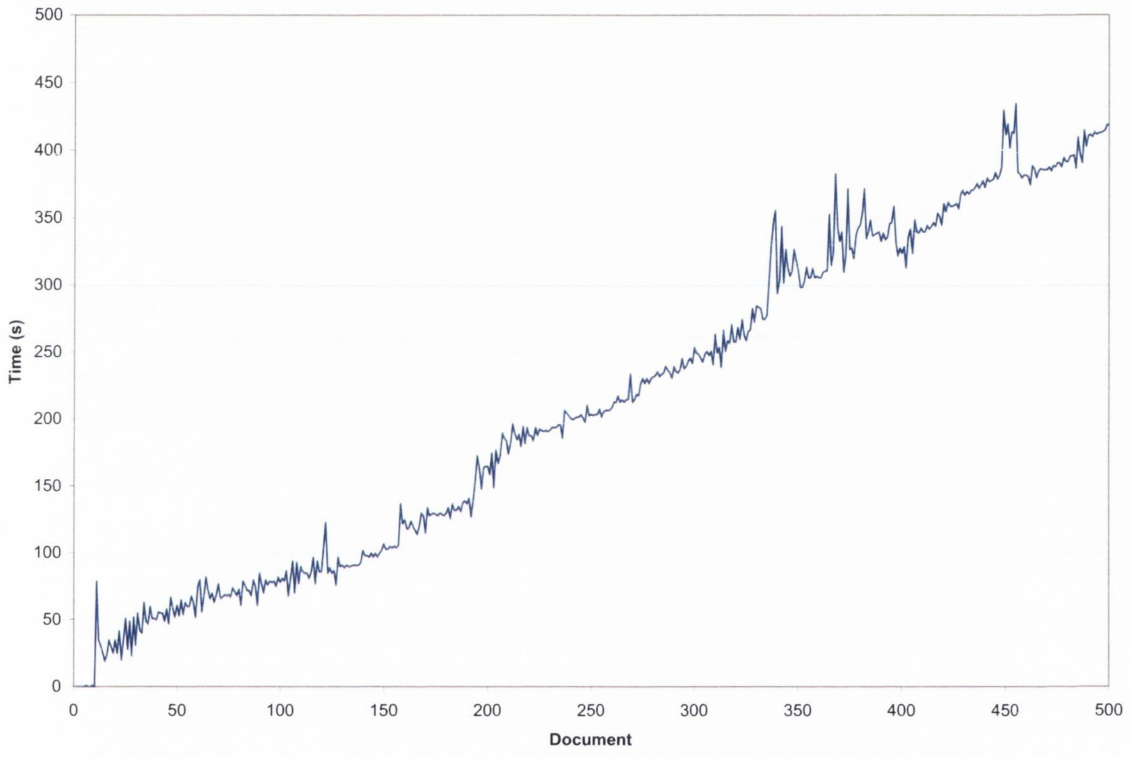


Figure 7.2: Time taken to add each document to the collection ($r = 1$)

7.5 Conclusion

We presented a procedure for the generation of homogeneous corpora from unstructured Web sites. The method is unrestrained by Web domain information and can therefore be applied to any section of the Web and across domains. This is important as the Web develops into a freer medium in which users have more control over content.

The procedure ranks prospective next steps in the corpus generation process (i.e. prospective documents to add at any stage) with the t-value measure, and adds those documents with the lowest score.

The procedure was tested on the Wikipedia corpus, and it was found to result in homogeneity scores up to 20% better than simple spreading from a starting position on the Web. However, it is a computationally expensive method, with the computation time increasing on a polynomial basis as the corpus size is increased. A compromise solution is therefore suggested that allows a trade-off between homogeneity and computation time, which may be acceptable for many applications.

Chapter 8

Conclusions

8.1 Overview

The Web is an invaluable global information resource, unprecedented in terms of scale and accessibility, and unique in terms of its democratic and unrestricted nature. One fundamental aspect of the Web that sets it apart from other large-scale information resources is the hypertext medium on which it is represented. The hyperlink structure of the Web offers an important source of information to applications that is supplemental to document content. However, initial approaches to Web document processing either ignored this information, or used it to tackle a specific practical function, such as improving document ranking or Web-space visualisation.

This work interprets hypertext links as providing information about document 'context', and presents a flexible method for the incorporation of this contextual information in a variety of Web-based document processing applications. Contextual information is measured using a variation of the TF$\times$IDF term-weighting measure, TF$\times$INF (Term Frequency by Inverse Neighbour Frequency), that calculates the frequency of a term in a document with respect to the frequency of the term in the document's 'context set' – the set of documents connected to the document by a path of hyperlinks up to a given length.

8.2 Summary of Results

The value of the TF$\times$INF measure was evaluated by implementing three Web-based document processing applications. The first two applications, automatic keyphrase extraction and document retrieval, used a Bayesian probability model trained on sample data to incorporate TF$\times$INF. The third, a measure of content homogeneity in Web corpora, interprets TF$\times$INF as a form of TF$\times$IDF localised on subsections of a corpus bound by hyperlinks and therefore likely to be related in content. The distributions of TF$\times$INF values and TF$\times$IDF values in the corpus are compared using a χ^2 measure.

8.2.1 Automatic Keyphrase Extraction

A non-Web-based automatic keyphrase extractor, KEA, was extended by including TF×INF as a feature in its Naïve Bayes classifier. The application was evaluated on four corpora obtained from the Web. Documents in these corpora contained annotated keywords that were used to train and evaluate the classifier. A rise in the number of correctly identified keyphrases was reported for all corpora, with a maximum improvement of 89% in the large corpus obtained from the Java Web site. This result is obtained when using a context set of path length 1, and using only out-links (links from a document to elements of its context set). TF×INF KEA was therefore found to outperform the state of the art in domain-independent automatic keyphrase extractors when applied to Web data. TF×INF KEA was found to perform slightly better when the training set and test set were drawn from the same corpus, but this domain-dependence did not inhibit its use in environments when domain-specific training data was unavailable.

8.2.2 Document Ranking

A document ranking post-processor was developed by adapting the probability model generator used in TF×INF KEA. A document relevance value (RSV – Retrieval Status Value) with respect to a query was defined as the probability that the query terms were keyphrases of the document. The resultant ranking algorithm was applied as a post-processor of search engine result lists ordered by a selection of different ranking algorithms: TF×IDF, TF×IDF including a relevance propagation component (see section 2.2), and variations of the PageRank and HITS link-topology-based ranking algorithms (see section 2.1). The test sets used in chapter 4 were used here.

The post-processor was found to improve the precision-recall curves of the result lists ranked by the link-topology algorithms, showing that on local Web document collections (i.e. those with no external in- or out-links – links to or from documents on a different domain), TF×INF provides information that is important and different to information about authority that is provided by these algorithms.

The ability of TF×INF to improve the results lists ordered by TF×IDF was found to be dependent on the behaviour of the annotated keyphrases used to evaluate the method. In one corpus the keyphrases are too general, and in another, they are too specific to be accurate representations of document relevance. In two other corpora, where the keyphrases had a more moderate distribution, TF×INF was able to improve the rankings of the result list to a greater degree than the competing network-based method, Relevance Propagation.

8.2.3 Homogeneity

The TF×INF homogeneity measure was evaluated on a corpus taken from the Wikipedia Web site, with manually-annotated category information. This category information was used to obtain extrinsic homogeneity values for subsets of the corpus, to which the TF×INF homogeneity measure was compared. The correlation between the two measurements was observed to be strong in comparison to other non-Web-based homogeneity techniques, and a variant of these techniques that uses clustering to incorporate Web information.

The method was also evaluated on synthetic corpora with controlled content homogeneity, and corpora obtained from the Java Web site. The homogeneity levels of these corpora were controlled using the URL directory structure of the site.

A corpus generation algorithm that uses the homogeneity measure was also developed and evaluated. This algorithm verified the hypothesis that corpus heterogeneity roughly increases as documents are added to the corpus, and attempted to mitigate this increase by selecting new linked documents according to the degree to which including them would change the homogeneity of the corpus. Using this method to generate homogeneous corpora was found to increase average homogeneity by up to 20%, according to the TF×INF-based homogeneity measure, but did so at an expensive time overhead.

8.3 Discussion

The above results demonstrate the flexibility of TF×INF as a measure of contextual information in a variety of document processing applications. Existing work on the use of hyperlinks has been focused on particular applications, such as information retrieval or Web visualisation. In this work, context is defined externally to these applications and contextual information is represented as a term weighting measure, allowing it to be used in a variety of unrelated applications.

Some works also use a restricted definition of context, weighting neighbours according to their similarity to a document or to a query. This ignores the fact that related, but textually dissimilar content is nonetheless potentially useful contextual information. This work enables this more flexible definition of context by treating TF×INF as a non-linear measure. In the keyphrase extraction and information retrieval chapters, this non-linear measure is interpreted by training a probabilistic model that estimates term importance, or document relevance to a query, according to TF×INF values.

Since document out-links are exhaustively defined in a document's HTML source, they are inexpensive to locate and traverse in comparison to in-links. TF×INF was found to require only document out-links. Therefore, TF×INF values for terms in a document can be calculated on the fly, without recourse to a large-scale internal representation of the Web's hyperlink structure, unlike HITS and PageRank. The complexity of the calculation of a TF×INF value (or a probability value based on TF×INF) is therefore constant after the contents of a document and its context set have been indexed. The indexing process is linear with respect to the document's out-degree and the size of the document and its neighbours. TF×INF can therefore be used online (e.g. at query-time in an IR situation) with low computational cost. As a result, TF×INF technology is computationally inexpensive to include in existing IR applications. The low computational cost of TF×INF is vital in Web IR, where the search space is of the order of several billion, and several million queries are carried out every day.

Another advantage of TF×INF is that, other than the model training stage required by some applications, it requires only information that is currently readily available on the Web: document content and out-links. Some works interpret hyperlinks according to unreliable data sources, such as a hyperlink's URL directory structure, or metadata attached to links or documents, or are intended to be applied to a finite document collection (e.g. TF×IDF). TF×INF does not require such information, and can therefore be applied to a large range of Web sites and documents without user input. Web query languages allow the user to constrain the set of potential search results according to the contents of their context sets, but often require the user to have explicit knowledge of the site structure of the Web environment in which they expect result documents to be located. TF×INF requires no such knowledge, and implicitly includes the contextual information without requiring a complex search query formulation. TF×INF can therefore be applied in the background of IR systems without requiring extra input from the user.

Ongoing work on the Semantic Web (Berners-Lee, 1999) aims to codify relationships between Web documents in a machine-understandable form, in order to allow applications to intelligently interpret and act upon Web information. The practical feasibility of this project has been called into question, due to problems associated with comprehensive ontology definitions and the scale of the Web. Since TF×INF represents contextual information on the Web, it may be able to act as a bridge from the current system to the Semantic Web by using this contextual information to suggest possible semantic relationships between documents.

8.4 Future Work

This work described and evaluated three applications that benefit from the incorporation of contextual information. These applications provide a rounded impression of the scope of TF×INF by demonstrating its use in the applied fields of information retrieval and document summarisation, and the field of corpus linguistics. Extensions to this work include the identification, implementation and evaluation of other document processing applications that can take advantage of contextual information in this way. Another direction for future work is the development of an application kit that uses and extends the applications described here to improve a Web user's browsing experience.

8.4.1 Potential Additional Applications

Text Categorisation & Clustering

Automatic text categorisation is the classification of documents into a predefined finite set of subjects or categories. A related problem is the unsupervised clustering of similar or related documents into undefined groups. The prevalence of content on the Web and the lack of a consistent classification scheme gives added importance to these tasks when applied to Web documents. Example applications of Web-based text categorisation are:

- Identification of *Web Spam* or *Spamdexing*: documents that attempt to undermine a search engine's ranking system in order to obtain an artificially inflated rank. The identification of such documents is an automatic classification task that has great importance for the future of Web search engines. Various Spamdexing methods are described in Gyongyi and Garcia-Molina (2005) and Metaxas and DeStefano (2005) and include text and link-based methods. Therefore, to fully solve this problem, a Web-specific classification method is required, that will take both textual content and hyperlinks into account.

- Automatic clustering of Web log content or news documents: A feature of current trends on the Web (often described as *Web 2.0*) is the re-use, combination and integration of data from multiple sources (*Mashups*). Such re-use is possible due to the separation of presentation and content on the Web, using technologies such as XML and RSS. The intention behind these methods is to provide an 'information push' delivery paradigm, where users are actively presented with information that they may be interested in, to complement the 'information pull' paradigm used in Web search engines. For example, instead of requiring the user to visit several news Web sites daily, services such as Google News[1] combine content from multiple sources and present them to the user on one site, with related stories from different sources clustered together, or ranked in terms of their information value. The challenge for such services is the inexpensive automatic collection and categorisation of stories. Since news stories often contain links to previous stories on the same subject that provide the context to the story in question, a link-based classification algorithm is suitable for this purpose.

Non-Web Hypertexts

Early hyperlink-based applications were applied to pre-Web hypertexts and hypertext-like structures induced from document collections with natural links, such as collections of academic articles with citation links between them. Other collections of manually-connected texts include e-mail and message board threads, in which previous messages are frequently referenced by newer messages. In such collections, a definition of context similar to that used in this work can be applied, and future work will entail the adaptation of these methods to such environments.

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

Non-textual Content

With the onset of inexpensive broadband Internet connections, video, sound and image content has become much more widespread on the Web. Web sites such as YouTube are examples of the use of video as a communications medium that is no longer restricted to large media companies. As with other information sources on the Web, videos have a context set of other videos made by the same author, user-annotated descriptions and meta-data, textual and video content found in hyperlinks from a document displaying the video etc. The interpretation of such content using the TF$\times$INF system described here would require a textual transcript of the video content, raising questions of successful signal-processing and speech recognition. Also, in a multi-media setting, information in the context set will be categorised according to its source (description text, hyperlinked video content, hyperlinked textual content etc.) allowing a potentially more detailed interpretation of context to that described here, as different information sources may provide different types of information about the video. Nevertheless, it is possible that a system similar to TF$\times$INF may have a part to play in the interpretation and indexing of such videos in the future.

Hypertext Generation

Previous works in the field of hypertext generation often employ a strategy of generating hyperlinks between textually similar documents (e.g. Kurland and Lee (2006)). However, this ignores the principal advantage of hyperlinks in document collections; that they allow the user to locate complementary or related sources of information, that may or may not be textually similar to each other. The probabilistic model used in the keyphrase extraction and document ranking algorithms described above may be used to identify possible candidate documents that belong in each other's context sets. The probability of a link existing between two document can be calculated by identifying the TF$\times$INF values of terms in the documents and determining whether those values are more likely to occur in hyperlinked documents or non-hyperlinked documents.

8.4.2 An Application Kit

The applications described in this work can be of practical benefit to Web users, and future work will involve the development of an application kit that implements these applications. For example, many people use browsers (e.g. Mozilla Firefox) for which extension software can be easily developed. A Firefox extension based on TF$\times$INF may contain the following functionality:

- Hovering the mouse pointer over a hyperlink gives an approximation of its information content. This information content is calculated by generating a keyphrase list for both the source and destination documents (in the background), and reporting those keyphrases that are in the destination document's keyphrase list but not the source document's. This would give information about the contents of the destination document that is not shared by the source document. Links could also be ranked or colour-coded according to the amount of keyphrases that the destination document shares with the source document. If the keyphrase lists are wildly different, this may be interpreted to indicate a hyperlink that points to a document with little semantic connection to the current one.

- Automatic generation of hyperlinks to dictionary or encyclopædia pages containing definitions of key phrases in the document. A Firefox extension that contains this functionality already exists[2]. No extra functionality would be added to this extension, but the existing keyphrase extractor (based on heuristics) would be augmented to use TF$\times$INF.

- Web Visualisation: A visualisation of the Web graph could be simplified by clustering documents that share a query term according to the TF$\times$INF value of the term. Documents that each contain the query term and form a connected subsection of the graph would each have the query term in their context sets, and would therefore have low TF$\times$INF values. Documents that contain the query term but do not have neighbours that contain the term would have higher TF$\times$INF values. Based on a user's information need, a user may prefer to retrieve a document of either type, and visualisation software based on TF$\times$INF could help a user to make such a decision. An alternative to clustering is simply colour-coding graphic representations of nodes in the Web graph according to the TF$\times$INF value of a query term in that document. A sample colour-coding Web visualisation tool has been developed (in Java) and is described in appendix C.

- Personalisation of ranking scores: An advantage of using a probability model in a document ranking algorithm is that the mechanism for ranking of a document is not inherent in the system, but is dependent on the distribution of the probability model. This allows the personalisation of a search engine by altering the probability distribution to favour terms that have particular TF$\times$INF values. This can be carried out automatically by training the probability model using relevance feedback. Since the only information required by the probability model is its TF$\times$INF score, a client-side ranking algorithm need only store the probability model in the form of a set of histograms (or parametric distributions) for each of the classes, by default – relevant and non-relevant. The server-side search engine can then return to the client a set of document titles and their TF$\times$INF scores with respect to a query, which are then ordered at client side by consulting the probability model. The model would be initialised to be equal to a generic model, following which, if a user adjudges a document to be relevant to a query, that information is fed back into the probability model. Likewise if a user adjudges a document irrelevant, that provides the model with negative information. In this way, the ranking algorithm will become more accurate in time, and will be capable of adapting to users' specific needs or habits. In addition to the default classes in the probability

[2]Wikiproxy: `http://userscripts.org/scripts/show/1632`

model (relevant and non-relevant) it may be possible to improve search engine accuracy by including other scores according to the search habits of the user. For example, the relevance of documents to an encyclopædic query may follow a different probability distribution to the relevance of documents to a shopping-based query. In this case, the user could specify at query time the type of query that they are making, and the probability model would contain the extra classes *encyclopædia* and *shopping*. Documents that achieve high probability scores for the class that matches the query would be given a stronger rank. The use of a probability model in a retrieval algorithm allows a large amount of personalisation in this way.

- Additional functionality, such as bookmark optimisation, is described in chapter 4.

8.5 Conclusion

In this thesis, the question of contextual information in Web documents is addressed. The principal contributions of this work are summarised below:

- A term-weighting scheme is proposed that incorporates contextual information from neighbouring Web documents in an unrestrictive form. This scheme is capable of capturing the fact that a term may be common in a subsection of the Web, or specific to a particular document within that subsection. The scheme is highly flexible, in that it can be applied to a number of Web-based document processing applications, and does not use information resources that may be unreliable in Web documents, such as document or link meta-data, URL directory structure or hierarchy, or collection-based data.

- A Web-based automatic keyphrase extractor is implemented based on a probabilistic classifier that uses this term-weighting scheme, and was capable of identifying a significantly larger amount of suitable keyphrases than the non-Web version of the algorithm.

- A framework for the inclusion of contextual information in a IR document ranking algorithm was proposed, and was found to provide complementary relevance information to TF$\times$IDF and link topology schemes such as PageRank.

- A Web-based content homogeneity measure was introduced. This measure compared Web document contextual information to the overall corpus in order to determine the homogeneity of the overall collection. The measure was compared with non-Web based homogeneity measures and was found to correlate more strongly with the variation of categories in collections of encyclopædia documents.

Bibliography

S. Abiteboul. Querying semi-structured data. In *ICDT '97: Proceedings of the 6th International Conference on Database Theory*, pages 1–18, London, UK, 1997. Springer-Verlag. ISBN 3-540-62222-5.

L. A. Adamic. Zipf, Power-laws, and Pareto - a ranking tutorial. URL `http://www.hpl.hp.com/research/idl/papers/ranking/ranking.html`.

L. A. Adamic and B. Huberman. Zipf's law and the internet. *Glottometrics*, (3):143–150, 2002.

F. Aguiar. Improving Web search by the identification of contextual Information. In *Intelligent exploration of the Web*, pages 197–224. Physica-Verlag GmbH, 2003.

J. Allan. *Automatic Hypertext Construction*. PhD thesis, Cornell University, 1996.

J. Allan, A. V. Leouski, and R. C. Swan. Interactive cluster visualization for information retrieval. Technical report, In Proceedings of ECDL'98, 1997.

T. C. Almind and P. Ingwersen. Informetric Analyses on the World Wide Web: Methodological Approaches to 'Webometrics'. *Journal of Documentation*, 53(4):404–426, 1997.

G. O. Arocena, A. O. Mendelzon, and G. A. Mihaila. Applications of a Web Query Language. *Comput. Netw. ISDN Syst.*, 29(8-13):1305–1316, 1997.

P. Atzeni, G. Mecca, and P. Merialdo. To Weave the Web. In *VLDB '97: Proceedings of the 23rd International Conference on Very Large Data Bases*, pages 206–215, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc. ISBN 1-55860-470-7.

A.-L. Barabasi and R. Albert. Emergence of scaling in random networks, 1999. URL `http://arxiv.org/abs/cond-mat/9910332`.

A.-L. Barabasi, R. Albert, and H. Jeong. Scale-free characteristics of random networks: the topology of the World-Wide Web. *Physica A: Statistical Mechanics and its Applications*, 281 (1-4):69–77, 2000.

T. Berners-Lee. The ENQUIRE System - Short Description, 1980. URL `http://infomesh.net/2001/enquire/manual`.

T. Berners-Lee. *Weaving the Web*. Harper San Francisco, 1999.

M. Bernstein, M. Joyce, and D. Levine. Contours of constructive hypertexts. In *ECHT '92: Proceedings of the ACM conference on Hypertext*, pages 161–170, New York, NY, USA, 1992. ACM. ISBN 0-89791-547-X. doi: http://doi.acm.org/10.1145/168466.168517.

K. Bharat, B. Chang, M. Henzinger, and M. Ruhl. Who links to whom: Mining linkage between Web sites. In *International Conference on Data Mining (ICDM)*. IEEE, 2001.

D. Biber. Using register-diversified corpora for general language studies. *Comput. Linguist.*, 19 (2):219–241, 1993.

R. A. Botafogo. Cluster analysis for hypertext systems. In *SIGIR '93: Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 116–125, New York, NY, USA, 1993. ACM. ISBN 0-89791-605-0. doi: http://doi.acm.org/10.1145/160688.160704.

R. A. Botafogo and B. Shneiderman. Identifying aggregates in hypertext structures. In *HYPERTEXT '91: Proceedings of the third annual ACM conference on Hypertext*, pages 63–74, New York, NY, USA, 1991. ACM. ISBN 0-89791-461-9. doi: http://doi.acm.org/10.1145/122974.122981.

M.-M. Bouamrane and S. Luz. Meeting Browsing: State-of-the-art Review. *Multimedia Systems*, 12(4-5):439–457, 2006.

S. Brin and L. Page. The Anatomy of a Large-Scale Hypertextual Web Search Engine. *Computer Networks and ISDN Systems*, 30(1-7):107–117, 1998.

A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener. Graph structure in the Web. *Comput. Networks*, 33(1-6):309–320, 2000.

V. Bush. As We May Think. *Library Computing*, 18(3):180, 1945.

F. Can and Y.-M. Lee. Hypir: a hypertext-based approach to information retrieval. In *SAC '93: Proceedings of the 1993 ACM/SIGAPP symposium on Applied computing*, pages 729–736, New York, NY, USA, 1993. ACM. ISBN 0-89791-567-4. doi: http://doi.acm.org/10.1145/162754.167177.

G. Cavaglia. Measuring corpus homogeneity using a range of measures for inter-document distance. In *Proceedings of LREC*, pages 426–431, 2002. URL `citeseer.ist.psu.edu/article/cavaglia02measuring.html`.

M. Ceglowski, A. Coburn, and J. Cuadrado. Semantic Search of Unstructured Data Using Contextual Network Graphs. Technical report, 2003.

O. C. L. Center. Web Characterization Project: Size & Growth Statistics, 2003. URL `www.oclc.org/research/projects/archive/wcp/stats/size.htm`.

S. Chakrabarti, B. Dom, and P. Indyk. Enhanced hypertext categorization using hyperlinks. In *SIGMOD '98: Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, pages 307–318, New York, NY, USA, 1998a. ACM. ISBN 0-89791-995-5. doi: http://doi.acm.org/10.1145/276304.276332.

S. Chakrabarti, B. Dom, P. Raghavan, S. Rajagopalan, D. Gibson, and J. Kleinberg. Automatic resource compilation by analyzing hyperlink structure and associated text. *Comput. Netw. ISDN Syst.*, 30(1-7):65–74, 1998b. ISSN 0169-7552. doi: http://dx.doi.org/10.1016/S0169-7552(98)00087-7.

J. Cho and S. Roy. Impact of search engines on page popularity. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 20–29, New York, NY, USA, 2004. ACM. ISBN 1-58113-844-X. doi: http://doi.acm.org/10.1145/988672.988676.

P. R. Cohen and R. Kjeldsen. Information Retrieval by constrained spreading activation in semantic networks. *Inf. Process. Manage.*, 23(4):255–268, 1987.

J. Conklin. Hypertext: An introduction and survey. *IEEE Computer*, 1987.

W. B. Croft and D. J. Harper. Using probabilistic models of document Retrieval without relevance Information. In *Readings in Information Retrieval.* 1997.

W. B. Croft and H. Turtle. A retrieval model incorporating hypertext links. In *HYPERTEXT '89: Proceedings of the second annual ACM conference on Hypertext*, pages 213–224, New York, NY, USA, 1989. ACM. ISBN 0-89791-339-6. doi: http://doi.acm.org/10.1145/74224.74242.

W. B. Croft and H. R. Turtle. Retrieval strategies for Hypertext. *Inf. Process. Manage.*, 29(3):313–324, 1993.

D. B. Crouch, C. J. Crouch, and G. Andreas. The use of Cluster Hierarchies in Hypertext Information Retrieval. In *Hypertext '89: Proceedings of the second annual ACM conference on Hypertext*, Pittsburgh, Pennsylvania, United States, 1989.

S. Deerwester, S. Dumais, T. Landauer, G. Furnas, and R. Harshman. Indexing by Latent Semantic Analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

L. Denoyer and P. Gallinari. INEX reports: The Wikipedia XML corpus. *ACM SIGIR Forum*, 40 (1), 2006.

P. Domingos and M. Pazzani. On the Optimality of the Simple Bayesian Classifier under Zero-One Loss. *Machine Learning*, 29(2-3):103–130, 1997.

D. Donato, L. Laura, S. Leonardi, and S. Millozzi. Large scale properties of the Webgraph. *European Physical Journal B*, 38:239–243, 2004.

D. Durand and P. Kahn. MAPA: a system for inducing and visualizing hierarchy in websites. In *HYPERTEXT '98: Proceedings of the ninth ACM conference on Hypertext and hypermedia : links, objects, time and space—structure in hypermedia systems*, pages 66–76, New York, NY, USA, 1998. ACM. ISBN 0-89791-972-6. doi: http://doi.acm.org/10.1145/276627.276635.

C. E. Dyreson. A jumping spider to index concepts that span pages, 1997. URL `http://citeseer.ist.psu.edu/166929.html`.

C. E. Dyreson. Content-based navigation in a Mini-World Web, 2002. URL `http://citeseer.ist.psu.edu/568433.html`.

N. Eiron and K. S. McCurley. Untangling compound documents on the web. In *HYPERTEXT '03: Proceedings of the fourteenth ACM conference on Hypertext and hypermedia*, pages 85–94, New York, NY, USA, 2003. ACM. ISBN 1-58113-704-4. doi: http://doi.acm.org/10.1145/900051.900070.

M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the Internet topology, 1999. 316229 251-262.

Fayyad and Irani. Multi-interval discretization of continuous-valued attributes for classification learning. pages 1022–1027, 1993. URL `http://www.cs.orst.edu/\~{}bulatov/papers/fayyad-discretization.pdf`.

G. W. Flake, S. Lawrence, C. L. Giles, and F. M. Coetzee. Self-Organization and Identification of Web Communities. *IEEE Computer*, 35(3):66–71, 2002.

S. Fortunato, M. Boguna, A. Flammini, and F. Menczer. How to make the top ten: Approximating Pagerank from in-degree, 2005a. URL `http://arxiv.org/abs/cs.IR/0511016`.

S. Fortunato, A. Flammini, F. Menczer, and A. Vespignani. The egalitarian effect of search engines, 2005b. URL `http://arxiv.org/abs/cs/0511005`.

S. Fortunato, A. Flammini, F. Menczer, and A. Vespignani. Topical Interests and the Mitigation of Search Engine Bias. *National Academy of Sciences of the United States of America*, 103(34), 2006.

E. Frank, G. W. Paynter, I. H. Witten, C. Gutwin, and C. G. Nevill-Manning. Domain-Specific Keyphrase Extraction. In *IJCAI '99: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 668–673, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc. ISBN 1-55860-613-0.

H. P. Frei and D. Stieger. Making use of hypertext links when retrieving information. In *ECHT '92: Proceedings of the ACM conference on Hypertext*, pages 102–111, New York, NY, USA, 1992. ACM. ISBN 0-89791-547-X. doi: http://doi.acm.org/10.1145/168466.168502.

H. P. Frei and D. Stieger. The use of semantic links in Hypertext Information Retrieval. *Inf. Process. Manage.*, 31(1):1–13, 1995.

D. Gildea. Corpus Variation and Parser Performance. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing.*, pages 167–202, 2001.

P. A. Gloor. CYBERMAP: yet another way of navigating in hyperspace. In *HYPERTEXT '91: Proceedings of the third annual ACM conference on Hypertext*, pages 107–121, New York, NY, USA, 1991. ACM. ISBN 0-89791-461-9. doi: http://doi.acm.org/10.1145/122974.122985.

S. Golder and B. A. Huberman. Usage Patterns of Collaborative Tagging Systems. *Information Science*, 32(2):198–208, 2006.

G. Golovchinsky. What the query told the link: the integration of Hypertext and Information Retrieval. In *Eighth ACM conference on Hypertext*, Southampton, UK, 1997.

A. Goodman. An End to Metatags, 2002. URL `www.traffick.com/article.asp?aID=102`.

D. Goodman. *The Complete Hypercard Handbook*. 1988.

Z. Gyongyi and H. Garcia-Molina. Web Spam Taxonomy, 2005.

K. M. Hammouda, D. N. Matute, and M. S. Kamel. CorePhrase: Keyphrase Extraction for Document Clustering. In *Lecture Notes in Computer Science*, volume 3587/2005. Springer Berlin / Heidelberg, 2005.

D. J. Hand and K. Yu. Idiot's Bayes - Not So Stupid After All? *International Statistical Review*, 69(3):385–398, 2001.

P. Harding. Automatic Indexing and Classification for Mechanised Information Retrieval. Technical Report 5723, 1982.

M. Henzinger. Link Analysis in Web Information Retrieval. *IEEE Data Engineering Bulletin*, 23 (3):3–8, 2000.

M. Henzinger. Hyperlink Analysis for the Web. *IEEE Internet Computing*, 5(1):45–50, 2001.

J. E. Hirsch. An index to quantify an individual's scientific research output. *PNAS*, 102(46): 16569–16572, 2005.

J. Hodgson. Do HTML Tags Flag Semantic Content? *IEEE Internet Computing*, 5(1):20–25, 2001.

G. H. John and P. Langley. Estimating Continuous Distributions in Bayesian Classifiers. In *Eleventh Conference on Uncertainty in Artificial Intelligence*, 1995.

S. Jones, S. Lundy, and G. W. Paynter. Interactive Document Summarisation Using Automatically Extracted Keyphrases. In *35th Hawaii International Conference on System Sciences*, 2002.

D. Kelleher and S. Luz. Automatic Hypertext Keyphrase Detection. In *IJCAI '05: Proceedings of the international joint conference on artificial intelligence*, 2005.

D. Kelleher and S. Luz. 'Context' in Web Information Retrieval. *Trinity College Dublin Journal of Postgraduate Research*, 6:149–159, 2007a.

D. Kelleher and S. Luz. Identifying Content Homogeneity in collections of Web pages (under review), 2007b.

A. Kilgarriff. Comparing Corpora. *International journal of corpus linguistics*, 6(1):97–133, 2001.

A. Kilgarriff. Using Word Frequency Lists to Measure Corpus Homogeneity and Similarity between Corpora. In *Language Engineering for Document Analysis and Recognition. Proceedings, AISB Workshop, Falmer*, pages 231–245, 1997. URL `citeseer.ist.psu.edu/kilgarriff97using.html`.

A. Kilgarriff and G. Grefenstette. Introduction to the special issue on the Web as corpus. *Comput. Linguist.*, 29(3):333–347, 2003.

A. Kilgarriff and T. Rose. Measures for corpus similarity and homogeneity. In *Proceedings of the 3 rd conference on Empirical Methods in Natural Language Processing, Granada, Spain.*

M. E. I. Kipp and D. G. Campbell. Patterns and Inconsistencies in Collaborative Tagging Systems: An Examination of Tagging Practices. In *Annual General Meeting of the American Society for Information Science and Technology*, Austin, Texas, US, 2006.

R. I. Kittredge. Variation and Homogeneity of Sublanguages. In R. I. Kittredge and J. Lehrberger, editors, *Sublanguage: Studies of Language in Restricted Semantic Domains*, pages 107–137. de Gruyter, Berlin, 1982.

R. I. Kittredge. The Significance of Sublanguage for Automatic Translation. In *Theoretical and Methodological Issues in Machine Translation of Natural Langugages*, Colgate University, Hamilton, New York, 1985.

J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, 1999. ISSN 0004-5411. doi: http://doi.acm.org/10.1145/324133.324140.

J. M. Kleinberg, R. Kumar, P. Raghavan, S. Rajagopalan, and A. S. Tomkins. The Web as a Graph: Measurements, Models, and Methods. *Lecture notes in computer science*, July

1999 1999. URL `http://citeseer.ist.psu.edu/cache/papers/cs/8049/http:zSzzSzwww.almaden.ibm.comzSzcszSzk53zSzcocoon.pdf/kleinberg99{W}eb.pdf`.

A. Kogias and D. Anagnostopoulos. A Simulation-Based Evaluation of an Exponential Growth Copying Model for the Web-Graph. In *World Wide Web Conference Series*, 2003.

D. Konopnicki and O. Shmueli. W3QS: A Query System for the World-Wide Web. In *VLDB '95: Proceedings of the 21th International Conference on Very Large Data Bases*, pages 54–65, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc. ISBN 1-55860-379-4.

B. Krulwich and C. Burkey. Learning User Information Interests through the Extraction of Semantically Significant Phrases. In *Working Notes 1996 AAAI Spring Symposium Machine Learning in Information Access*, pages 110–112. AAAI Press, 1996.

O. Kurland and L. Lee. Corpus structure, language models, and ad hoc information retrieval. In *Proceedings of SIGIR*, pages 194–201, 2004.

O. Kurland and L. Lee. PageRank without hyperlinks: Structural re-ranking using links induced by language models, 2006. URL `http://arxiv.org/abs/cs.IR/0601045`.

K. L. Kwok. The use of title and cited titles as document representation for automatic classification. *Inf. Process. Manage.*, 11(8.12):201–206, 1975.

K. L. Kwok. A document-document similarity measure based on cited titles and probability theory and its application to relevance feedback Retrieval. In C. Van Rijsbergen, editor, *Research and Development in Information Retrieval*, pages 221–232. Cambridge University Press, 1984.

K. L. Kwok. A probabilistic theory of indexing and similarity measure based on cited and citing documents. *Journal of the American Society of Information Science*, 36, 1985.

K. L. Kwok. On the Use of Bibliographically Related Titles for the Enhancement of Document Representations. *Inf. Process. Manage.*, 24(2):123–131, 1988.

Z. Lacroix, A. Sahuguet, R. Chandrasekar, and B. Srinivas. A novel approach to querying the Web: Integrating Retrieval and Browsing. In *Workshop on Conceptual Modeling of Multimedia Information Seeking, in Conjunction with ER'97*, Los Angeles, CA, 1997.

L. V. S. Lakshmanan, F. Sadri, and I. N. Subramanian. A declarative language for querying and restructuring the WEB. In *RIDE-NDS*, pages 12–21, 1996. URL `citeseer.ist.psu.edu/article/lakshmanan96declarative.html`.

A. Le Hors, D. Raggett, and I. Jacobs. HTML 4.01 Specification. Technical report, W3C, December 1999.

Y. Li. Beyond Relevance Ranking: Hyperlink Vector Voting. In *ACM-SIGIR97 Workshop on Networked Information Retrieval*, 1997.

N. Litvak, W. R. W. Scheinhardt, and Y. Volkovich. In-Degree and PageRank of Web pages: Why do they follow similar power laws?, 2006. URL `http://arxiv.org/abs/math.PR/0607507`.

B. Liu, K. Zhao, and L. Yi. Visualizing web site comparisons. In *WWW '02: Proceedings of the 11th international conference on World Wide Web*, pages 693–703, New York, NY, USA, 2002. ACM. ISBN 1-58113-449-5. doi: http://doi.acm.org/10.1145/511446.511536.

J. B. Lovins. Development of a Stemming Algorithm. *Mechanical Translation & Computational Linguistics*, 11(22-31), 1968.

M. Marchiori. The quest for correct information on the web: hyper search engines. In *Selected papers from the sixth international conference on World Wide Web*, pages 1225–1235, Essex, UK, 1997. Elsevier Science Publishers Ltd. doi: http://dx.doi.org/10.1016/S0169-7552(97)00036-6.

A. Mathes. Folksonomies - Cooperative Classification and Communication Through Shared Metadata, 2004. URL `http://www.adammathes.com/academic/computer-mediated-communication/folksonomies.html`.

O. A. Mcbryan. Genvl and wwww: Tools for taming the web. In *In Proceedings of the First International World Wide Web Conference*, pages 79–90, 1994.

A. O. Mendelzon, G. A. Mihaila, and T. Milo. Querying the World Wide Web. In *DIS '96: Proceedings of the fourth international conference on on Parallel and distributed information systems*, pages 80–91, Washington, DC, USA, 1996. IEEE Computer Society. ISBN 0-8186-7475-X.

A. O. Mendelzon, G. A. Mihaila, and T. Milo. Querying the World Wide Web. *International Journal on Digital Libraries*, 1(1):54–67, 1997.

P. T. Metaxas and J. DeStefano. Web Spam, Propaganda and Trust. In *Proceedings of the 1st International Workshop on Adversarial Information Retrieval on the Web (AIRWeb).*, 2005.

Y. Mizuuchi and K. Tajima. Finding context paths for web pages. In *HYPERTEXT '99: Proceedings of the tenth ACM Conference on Hypertext and hypermedia : returning to our diverse roots*, pages 13–22, New York, NY, USA, 1999. ACM. ISBN 1-58113-064-3. doi: http://doi.acm.org/10.1145/294469.294474.

T. H. Nelson. Embedded Markup Considered Harmful. *World Wide Web*, 2(4):129–134, 1997.

T. O'Reilly. What is Web 2.0, 2005. URL `http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-{W}eb-20.html`.

L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the Web. Technical report, Stanford University., 1998.

B. Pinkerton. Finding What People Want: Experiences with WebCrawler. In *WWW 94, Second International WWW Conference*, 1994.

P. Pirolli, J. Pitkow, and R. Rao. Silk from a sow's ear: extracting usable structures from the web. In *CHI '96: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 118–125, New York, NY, USA, 1996. ACM. ISBN 0-89791-777-4. doi: http://doi.acm.org/10.1145/238386.238450.

J. Pitkow and P. Pirolli. Life, Death, and Lawfulness on the Electronic Frontier. In *Proceedings of the Conference on Human Factors in Computing Systems CHI'97*, 1997. URL `citeseer.ist.psu.edu/pitkow97life.html`.

J. M. Ponte and W. B. Croft. A Language Modeling Approach to Information Retrieval. In *Research and Development in Information Retrieval*, pages 275–281. 1998.

J. R. Quinlan. *C4.5: programs for machine learning.* Morgan Kaufmann Publishers Inc., San Francisco, USA, 1993.

L. Rainie. 28% of Online Americans Have Used the Internet to Tag Content, September 2007 2007. URL `www.pewinternet.org/pdfs/PIP_Tagging.pdf`.

M. Roederer, W. Moore, A. Treister, and L. A. Herzenberg. Probability Binning Comparison: A Metric for Quantitating Univariate Distribution Differences. *Cytometry*, (45):3746, 2001.

T. Rose, N. J. Haddock, and R. C. Tucker. The Effects of Corpus Size and Homogeneity on Language Model Quality. In *Workshop On Very Large Corpora*, August 1997 1997. URL `http://citeseer.ist.psu.edu/rose97effects.html`.

G. Salton. On the use of term associations in automatic information retrieval. *COLING-86*, pages 380–386, 1986. URL `citeseer.ist.psu.edu/salton86use.html`.

G. Salton. On the Automatic Generation of Content Links in Hypertext. Technical report, Cornell University, April 1989 1989.

G. Salton and C. Buckley. Term Weighting Approaches in Automatic Text Retrieval. Technical Report TR87-881, Cornell University, 1987.

G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval.* McGraw-Hill, Inc, New York, 1983.

G. Salton and Y. Zhang. Enhancement of text representations using related document titles. *Inf. Process. Manage.*, 22(5):385–394, 1986.

G. Salton, A. Wong, and C. S. Yang. A Vector Space Model for Automatic Indexing. *Communications of the ACM*, 12(11):613–620, 1975.

G. Salton, E. A. Fox, and H. Wu. Extended Boolean Information Retrieval. *Communications of the ACM*, 26(11):1022–1036, 1983.

J. Savoy. A learning scheme for Information Retrieval in Hypertext. *Inf. Process. Manage.*, 30(4): 515–533, 1994.

J. Savoy. An extended vector-processing scheme for searching information in hypertext systems. *Inf. Process. Manage.*, 32(2):155–170, 1996.

S. Sekine. The domain dependence of parsing. In *Proceedings of the fifth conference on Applied natural language processing*, pages 96–102, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc. doi: http://dx.doi.org/10.3115/974557.974572.

A. Shakery and C. Zhai. Relevance Propagation for Topic Distillation: UIUC TREC-2003 Web Track Experiments, 2003.

A. Shakery and C. Zhai. A probabilistic relevance propagation model for hypertext retrieval. In *CIKM '06: Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 550–558, New York, NY, USA, 2006. ACM. ISBN 1-59593-433-2. doi: http://doi.acm.org/10.1145/1183614.1183693.

M. Song, I. Y. Song, R. B. Allen, and Z. Obradovic. Keyphrase extraction-based query expansion in digital libraries. In *JCDL '06: 6th ACM/IEEE-CS joint conference on Digital Libraries*. ACM Press, 2006.

K. Spärck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28, 1972.

E. Spertus. *ParaSite: Mining the Structural Information on the World-Wide Web*. PhD thesis, Massachusetts Institute of Technology, 1998.

K. Sugiyama, K. Hatano, M. Yoshikawa, and S. Uemura. Refinement of tf-idf schemes for web pages using their hyperlinked neighboring pages. In *HYPERTEXT '03: Proceedings of the fourteenth ACM conference on Hypertext and hypermedia*, pages 198–207, New York, NY, USA, 2003. ACM. ISBN 1-58113-704-4. doi: http://doi.acm.org/10.1145/900051.900096.

M. Sydow. Can link analysis tell us about web traffic? In *WWW '05: Special interest tracks and posters of the 14th international conference on World Wide Web*, pages 954–955, New York, NY, USA, 2005. ACM. ISBN 1-59593-051-5. doi: http://doi.acm.org/10.1145/1062745.1062815.

R. Trigg. A Taxonomy of Link Types. In *A Network-Bsaed Approach to Text Handling for the Online Scientific Communitiy*. 1983.

P. D. Turney. Learning Algorithms for Keyphrase Extraction. *Information Retrieval*, 2(4):303–336, 2000.

C. Van Rijsbergen. *Information Retrieval.* Dept. of Computer Science, University of Glasgow, 1979.

L. Vaughan and M. Thelwall. Search engine coverage bias: evidence and possible causes. *Inf. Process. Manage.*, 40(4):693–707, 2004.

M. Volk. Using the Web as Corpus for Linguistic Research. *Thendusepüüdja. Catcher of the Meaning. A Festschrift for Professor Haldur im.*, 2002.

S. Walker, S. E. Robertson, M. Boughanem, G. Jones, and K. Sparck Jones. Okapi at TREC-6: Automatic ad hoc, VLC, routing, filtering and QSDR. 1997.

R. Weiss, B. Vélez, and M. A. Sheldon. HyPursuit: a hierarchical network search engine that exploits content-link hypertext clustering. In *HYPERTEXT '96: Proceedings of the the seventh ACM conference on Hypertext*, pages 180–193, New York, NY, USA, 1996. ACM. ISBN 0-89791-778-2. doi: http://doi.acm.org/10.1145/234828.234846.

H. D. White. Author Cocitation Analysis and Pearson's r. *Journal of the American Society of Information Science and Technology*, 54(13):1250–1259, 2003.

I. H. Witten, G. W. Paynter, E. Frank, C. Gutwin, and C. G. Nevill-Manning. KEA: practical automatic keyphrase extraction. In *DL '99: Proceedings of the fourth ACM conference on Digital libraries*, pages 254–255, New York, NY, USA, 1999. ACM. ISBN 1-58113-145-3. doi: http://doi.acm.org/10.1145/313238.313437.

O. Zamir and O. Etzioni. Grouper: A Dynamic Clustering Interface to Web Search Results. *Computer Networks*, 31(11-16):1361–1374, 1999.

H. Zha. Generic summarization and keyphrase extraction using mutual reinforcement principle and sentence clustering. In *SIGIR '02: 25th annual international ACM SIGIR conference on research and development in Information Retrieval*, pages 113–120, Tampere, Finland, 2002. ACM Press.

H. Zha, O. Marques, and H. Simon. A Subspace-Based Model for Information Retrieval with Applications in Latent Semantic Indexing. In *Irregular '98*, 1998.

H. Zhang. The Optimality of Naive Bayes. In V. Barr and Z. Markov, editors, *FLAIRS Conference.* AAAI Press, 2004. URL `http://www.cs.unb.ca/profs/hzhang/publications/FLAIRS04ZhangH.pdf`.

J. Zhu, J. Hong, and J. G. Hughes. PageCluster: Mining conceptual link hierarchies from Web log files for adaptive Web site navigation. *ACM Trans. Inter. Tech.*, 4(2):185–208, 2004.

M. Zizi and M. Beaudouin-Lafon. Accessing hyperdocuments through interactive dynamic maps. In *ECHT '94: Proceedings of the 1994 ACM European conference on Hypermedia technology*, pages 126–135, New York, NY, USA, 1994. ACM. ISBN 0-89791-640-9. doi: http://doi.acm.org/10.1145/192757.192786.

Appendix A

Glossary

3-point Precision Recall Average	
χ^2 Test	A statistical test used to determine the independence of two categorical variables.
Chi Square By Degrees of Freedom (CBDF)	The χ^2 test statistic divided by the degrees of freedom (number of terms used in the comparison) allowing the direct comparison of χ^2 values calculated using a differing number of terms in a subcorpus comparison.
HITS	Hyperlink-Induced Topic Search (HITS) is an iterative algorithm used to identify authoritative Web pages by analysing the hyperlink structure of the Web graph surrounding them.
In-degree	The in-degree of a Web page is number of Web pages with hyperlinks that connect to it.
Information Retrieval (IR)	The identification of informative documents or passages of text in a set of documents.
KDE	Kernel Density Estimation is a method for estimating the PDF of a random variable by generating a normalised sum of a set of normal 'kernels' centred around each example in the training set.
KEA	A Java application based on the Weka machine learning toolkit that builds a Naïve Bayes model using training documents with annotated keyphrases and then uses the model to associate probabilities to candidate keyphrases in unseen documents.
Naïve Bayes classifier	A probabilistic classifier based on supervised learning using Bayes' theorem. The classifier assumes that feature probabilities are conditionally independent of each other.

Nearest Neighbour (NN)	The Nearest Neighbour classifier assigns to an unclassified example the class of the nearest classified example in a multidimensional space. A variant will assign the average class of n nearest classified neighbours.
NLP	Natural Language Processing is the study of the automatic generation and comprehension of natural human language.
Out-degree	The out-degree of a Web page is the number of distinct links on that Web page.
PageRank	PageRank is an iterative algorithm that identifies prestigious or popular pages on the Web by assigning each page a score based on the sum of the scores of pages that link to it. It is the basis for the Google search engine.
Precision	In IR, Precision is a measurement of the accuracy of a search result set. It is calculated as the number of relevant documents divided by the total number of retrieved documents.
Probability Mass Function (PMF)	A Probability Mass Function is a function that assigns a probability to each possible value of a discrete random variable. It is equivalent to a normalised histogram.
Recall	In IR, Recall is a measurement of the completeness of a search result set. It is calculated as the number of relevant retrieved documents divided by the total number of relevant documents.
Relevance Propagation (RP)	Relevance Propagation is a technique used in hypertext IR in which the retrieval status values (RSVs) of documents relative to a query are influenced by the RSVs of neighbouring pages.
Retrieval Status Value (RSV)	The Retrieval Status Value of a document is a score assigned to it based on its relevance to a particular query, as determined by some IR algorithm.
Scale-free Network	A scale-free network is a form of complex network in which the degree distribution of the nodes in the network follows a power-law. The exponent of the power law remains constant as the size of the network changes. This form of network structure is evident in many 'real-world' networks, including the Web.
Spearman Rank Correlation Coefficient	A non-parametric measure of correlation between two random variables based on the rankings of ordered data.

TF×IDF	A term-weighting measure in which the weight of a term is given by its normalised frequency in a document divided by the proportion of documents in a set that contains the term. This is frequently used in traditional IR applications.
TF×INF	A variation of TF×IDF in which the term frequency is divided by the terms frequency in a 'context set' of connected documents in a hypertext, rather than its frequency in an overall document set.
Vector Space Model (VSM)	The Vector Space Model is an IR model in which documents and queries are represented as vectors in a multidimensional space in which each dimension represents a term in the vocabulary. This model can then be used to determine distances between documents and queries (in the case of IR) or distances between documents (in the case of text classification).
Zipf law	A probability distribution in which the probability of an event (e.g. a term in corpus linguistics, or the in-degree of a page in a hypertext) is in a power-law relationship with the rank of its frequency in an ordered list.

Appendix B

Application Structures

B.1 Keyphrase Extractor

`public` KEA. **KEAModelBuilder**

Builds a keyphrase extraction model from a set of training documents. Generates a **KEAFilter** object and passes the textual content and keyphrases of each document in a directory to it. Saves the **KEAFilter** object as a Java object file when all documents have been passed.

`public abstract` KEA. **WebModelBuilder** Extends:**KEAModelBuilder**

Generic TF$\times$INF KEA model builder. Adapts the KEA model builder to Web documents by incorporating hyperlink information in the document representation passed to the **KEAFilter** object.

`public` KEA. **TFINFModelBuilder** Extends:**WebModelBuilder**

Creates a **TFINFFilter** object. Generates a context set by locating all documents connected by one out-link to a document. Sends their contents to the filter.

`public` KEA. **GeneralTFINFMB** Extends:**WebModelBuilder**

Creates a **GeneralTFINFFilter** object. A more general model builder that contains a *generality* field that determines the size of the context set by maximum path length. The filenames of the documents in the context set are sent to the filter.

`public` KEA. **KEAKeyphraseExtractor**

Automatically generates keyphrases for all documents in a directory and compares them to annotated keyphrases in the `.key` files. Loads a **KEAFilter** model from a given filename. Reads each document in turn and passes its contents to the model. The five most likely keyphrases are then retrieved from the model, and outputted. After all documents have been processed, the average number of correct keyphrases are calculated and reported.

`public abstract` KEA. **WebExtractor** Extends:**KEAKeyphraseExtractor**

Generic class that adapts **KEAKeyphraseExtractor** to incorporate hyperlink information. Loads a model of type **WebFilter**

`public` KEA. **TFINFExtractor** Extends:**WebExtractor**

Front-end keyphrase extraction class for TF×INF KEA. Loads a model of type **TFINFFilter**. Generates a context set by locating all documents connected by one out-link to a document. Sends their contents to the filter.

`public` KEA. **GeneralTFINFExt** Extends:**WebExtractor**

A more general TF×INF keyphrase extractor class for TF×INF KEA. Loads a model of type **GeneralTFINFFilter**. Contains a *generality* field that determines the size of the context set by maximum path length. The filenames of the documents in the context set are sent to the filter.

`public` KEA. **KEAFilter** Extends:**Filter**

Contains the **DistributionClassifier** object that calculates the keyphrase probablilities. This class converts a training document into a set of phrases (n-grams). During the model-building phase, these are stored in a dictionary table. Once all documents have been converted, the feature (TF×IDF and *First Occurrence*) values for each phrase are calculated, and the classifier object is built. During the extraction phase, the feature values of each phrase are calculated and sent to the classifier. The classifier returns a probability distribution for the classes <keyphrase, non-keyphrase>. The phrases are then ordered according to the <keyphrase> probability value, and the top five are returned to the keyphrase extractor.

`public abstract` KEA.**WebFilter** Extends:**KEAFilter**

Includes a TF×INF feature in the classifier, that is calculated by inheritors of this class.

`public abstract` KEA.**TFINFFilter** Extends:**WebFilter**

Calculates TF×INF as the ratio of TF to INF (context set $n = 1$). INF is calculated by inspecting the contents of the neighbouring documents, sent to the filter by the **TFINFModelBuilder** or **TFINFExtractor** objects.

`public abstract` KEA.**GeneralTFINFFilter** Extends:**WebFilter**

Calculates TF×INF as the ratio of TF to INF with a variable-sized context set. The filenames of the documents in the context set are obtained from the model builder or extractor classes. The contents of these documents are then read, and used to calculate INF.

B.2 Document Retrieval

B.2.1 ModNLP wrapper for the BerkeleyDB - IDX module

`public` modnlp.idx.database.**Dictionary**

Serves as an intermediate between the IR application and the database tables. Adds documents to the database and allows the lookup of phrases.

`public` modnlp.idx.database.**Dictionary**

Serves as an intermediate between the IR application and the database tables. Adds documents to the database and allows the lookup of phrases.

`public` search.idxWrapper.**WebDictionary** Extends:**Dictionary**

Adapts the **Dictionary** class for the Web, includes a **LinkTable** object `public abstract` `modnlp.idx.database.` **Table**

Encapsulates BerkeleyDB and acts as a template for all table classes.

`public` `modnlp.idx.database.` **CaseTable** Extends:**Table**

Stores a canonical (lowercase) form of a word along with all other forms in which it occurs.

`public` `modnlp.idx.database.` **FileTable** Extends:**Table**

Stores all files or URIs with an integer ID.

`public` `modnlp.idx.database.` **WordFileTable** Extends:**Table**

Stores words and the (integer) keys to files in which they occur.

`public` `modnlp.idx.database.` **WordPositionTable** Extends:**Table**

Stores words and the positions in which they occur in a document. One table is created for each document in the index.

`public` `search.idxWrapper.` **LinkTable** Extends:**Table**

Stores integer file keys with the keys of files to which they are linked.

B.2.2 Indexer

`public abstract` `search.makeIndex.` **Page** Extends:–

Implements:**Serializable**

Generic class to represent a Web document in the application. Includes a vector of out-links, and an optional vector of in-links.

`public` `search.indexSearch.` **IndexedPage** Extends:**Page**

Specification of the **Page** class to be used with an index. Stores a value containing the number of phrases in the document for quick reference.

`public` `search.idxWrapper.` **IDXPage** Extends:**IndexedPage**

Page class for use with the ModNLP IDX module. Stores the integer file key for quick lookup in the database.

`public` `search.makeIndex.` **IndexMaker**

Generates an index (stored in a Hashtable) by locating and downloading Web documents from a specified start URL. The hyperlinks from an indexed document are added to a **Queue** and are downloaded and indexed in turn until the index has reached a specified size or the maximum depth from the starting location has been reached.

`public` `search.makeIndex.` **ThreadedIndexMaker** Extends:**IndexMaker**

Optimises the basic index maker by generating a specified number of **IndexThread** threads that process different Web documents in parallel, avoiding some of the time cost of waiting for slow Web pages to download.

`public` `search.makeIndex.` **IndexThread** Extends:**Thread**

Reads a document from the top of the queue and downloads it. Sends the document to the **IndexMaker** object for inclusion in the index, and extracts all hyperlinks, for addition to the queue.

`public` `search.makeIndex.` **IDXIndexer** Extends:**ThreadedIndexMaker**

Specification of the **ThreadedIndexMaker**, incorporating the ModNLP IDX module. Contains a **WebDictonary** object that provides the interface to the Berkeley Database.

`public` `search.makeIndex.` **IDXIndexerFiles** Extends:**IDXIndexer**

Creates an index from a set of documents stored as `.txt` and `.links` files on a file system.

`public abstract` `search.` **GenericSearcher**

Provides an interface for a search application on a generic index.

`public` `search.idxWrapper.` **IDXSearch** Extends:**GenericSearcher**

Implements **GenericSearcher**. Contains a **WebDictonary** object that provides the interface to the Berkeley Database. A **SearchModel** object provides access to a **DistributionClassifier** object, that generates document retrieval status values (RSVs) according to the TF$\times$INF values of the query terms in the documents. When a query is received, the database is queried, and the results are ranked according to their RSVs.

`public` `search.idxWrapper.composite.` **CompositeSearch** Extends:**IDXSearch**

Extends **IDXSearch** to allow re-ranking of document result lists according to a series of ranking algorithms, stored as a vector of **SearchOrder** objects. This class encapsulates a linear composition of a number of ranking functions.

`public interface` `search.indexSearch.keaSearch.` **SearchModel**

Provides an interface for all ranking algorithms based on a **DistributionClassifier** object. Implementations of this interface implement the *featVals* method, generating the required feature values for the classifier.

`public interface` `search.idxWrapper.composite.` **SearchOrder**

This is the standard interface for all ranking algorithm implementations. The interface provides a *rearrangeList* method, that returns a result list that has been reordered according to the ranking algorithm.

`public abstract` `search.idxWrapper.composite.` **GenericSearchOrder** Extends:–

Implements:**SearchOrder**

Includes a proportion value (α) and a reference to the **WebDictionary**. When documents are re-ordered, their original input score (if present) is combined with their score from this ranking algorithm, according to α. All implementations of document ranking algorithms (TF$\times$IDF, TF$\times$INF, RP, PageRank, HITS) are subclasses of this class.

`public` `search.idxWrapper.` **IDXTrainer** Extends:**TFINFModelBuilder**

Subclass of **TFINFModelBuilder** that produces an **IDXFilter** model object.

`public` `search.idxWrapper.` **IDXFilter** Extends:**TFINFFilter**

Implements:**SearchModel**

Connects the document index with the probability model stored in the TF$\times$INF **Distribution-Classifier** object. This class allows feature values to be calculated for a phrase by inspecting the **WebDictionary**.

B.3 Corpus Homogeneity

`public` `search.idxWrapper.Homogeneity.generation.` **Controller**

This class is the manager class for the homogeneity calculator. It contains a **WebDictionary** object as an interface to an indexed collection of documents, and a **PBCalcRuntime** object, that carries out the homogeneity calculation.

`public` `search.idxWrapper.Homogeneity.` **PBCalcRuntime**

Calculates the t value for a corpus.

B.4 Description of Perl Scripts

B.4.1 fileconverter.pl

This script converts an HTML document into its text and link components. Each line in the HTML file is processed in turn as follows:

1 If the line is part of a script or style block then it is ignored

2 If the line contains a `<meta keyword...>` tag, then this tag is removed and the keywords are stored in a `.key` file.

3 If the line contains an `<a>` or `<link>` tag, these are removed, and the `href` values are added to the links file.

4 If the line contains a `<style>` or `<script>` tag then the rest of the line and any subsequent data is ignored until closing tags are found.

5 All other tags are removed

6 The remaining text in the line is added to the text file.

B.4.2 wikixmlconverter.pl

This script generates a subcorpus form the Wikipedia XML corpus, by converting XML documents into their text and link components. The Perl module `XML::Simple` is used to convert an XML document into a Perl data structure. XML elements marked “collectionlink” identify links to other XML documents in the corpus, and these are added to the document’s `.links` file. XML elements marked “content” contain text that is added to the `.txt` file.

The script is given the name of an XML file in the collection from which to start the subcorpus generation. If no file is given, a random document from the collection is chosen. Each document is processed, followed by all documents it links to, until a subcorpus of a certain size is attained.

B.4.3 setsplitter.pl

This script splits a hyperlinked document collection into a test set and a training set using ‘Dependency Splitting’, i.e. no document in the test set may link to one in the training set. A proportion value is provided to the script, specifying the size of the test set in relation to the overall corpus. The test set is built by iteratively adding to the test set those documents that have either no links, or just links to other documents in the test set. Once the test set reaches its desired size, or if no more documents can be added, the script ends.

B.5 Stop List

The following list of stop-words is used in the KEA keyphrase exraction application:

a	abaft	aboard	about	above	across
afore	aforesaid	after	again	against	agin
ago	aint	albeit	all	almost	alone
along	alongside	already	also	although	always
am	american	amid	amidst	among	amongst
an	and	anent	another	any	anybody
anyone	anything	are	aren't	around	as
aslant	astride	at	athwart	away	b
back	bar	barring	be	because	been
before	behind	being	below	beneath	beside
besides	best	better	between	betwixt	beyond
both	but	by	c	can	cannot
can't	certain	circa	close	concerning	considering
cos	could	couldn't	couldst	d	dare
dared	daren't	dares	daring	despite	did
didn't	different	directly	do	does	doesn't
doing	done	don't	dost	doth	down
during	durst	e	each	early	either
em	english	enough	ere	even	ever
every	everybody	everyone	everything	except	excepting
f	failing	far	few	first	five
following	for	four	from	g	gonna
gotta	h	had	hadn't	hard	has
hasn't	hast	hath	have	haven't	having
he	he'd	he'll	her	here	here's
hers	herself	he's	high	him	himself
his	home	how	howbeit	however	how's
i	id	if	ill	i'm	immediately
important	in	inside	instantly	into	is
isn't	it	it'll	it's	its	itself
i've	j	just	k	l	large
last	later	least	left	less	lest
let's	like	likewise	little	living	long
m	many	may	mayn't	me	mid

midst	might	mightn't	mine	minus	more
most	much	must	mustn't	my	myself
n	near	'neath	need	needed	needing
needn't	needs	neither	never	nevertheless	new
next	nigh	nigher	nighest	nisi	no
no-one	nobody	none	nor	not	nothing
notwithstanding	now	o	o'er	of	off
often	on	once	one	oneself	only
onto	open	or	other	otherwise	ought
oughtn't	our	ours	ourselves	out	outside
over	own	p	past	pending	per
perhaps	plus	possible	present	probably	provided
providing	public	q	qua	quite	r
rather	re	real	really	respecting	right
round	s	same	sans	save	saving
second	several	shall	shalt	shan't	she
shed	shell	she's	short	should	shouldn't
since	six	small	so	some	somebody
someone	something	sometimes	soon	special	still
such	summat	supposing	sure	t	than
that	that'd	that'll	that's	the	thee
their	theirs	their's	them	themselves	then
there	there's	these	they	they'd	they'll
they're	they've	thine	this	tho	those
thou	though	three	thro'	through	throughout
thru	thyself	till	to	today	together
too	touching	toward	towards	true	'twas
'tween	'twere	'twill	'twixt	two	'twould
u	under	underneath	unless	unlike	until
unto	up	upon	us	used	usually
v	versus	very	via	vice	vis-a-vis
w	wanna	wanting	was	wasn't	way
we	we'd	well	were	weren't	wert
we've	what	whatever	what'll	what's	when
whencesoever	whenever	when's	whereas	where's	whether
which	whichever	whichsoever	while	whilst	who
who'd	whoever	whole	who'll	whom	whore

who's	whose	whoso	whosoever	will	with
within	without	wont	would	wouldn't	wouldst
x	y	ye	yet	you	you'd
you'll	your	you're	yours	yourself	yourselves
you've	z				

Appendix C

Application Descriptions

C.1 Web Visualisation

C.1.1 Application Summary

The Web Visualisation program is a Java-based tool designed to aid analysis of a localised section of the Web. Using this program, the context set of a document can be identified and analysed. The program can also be included in the graphical interface of a Web search engine, in order to provide the user with an idea of their whereabouts within the Web.

C.1.2 Detailed Description

The Visualisation tool is a graphical Java interface, developed with the Swing graphical framework. The tool uses the JUNG[1] graph display module to present sections of a Web hypertext as a directed graph to the user. A hypertext is presented as a graph as follows:

- The documents become graph nodes,
- The hyperlinks (`<a>` or `<link>` tags in the HTML) become directed edges between nodes.

In the current version of the program, the graph edges are unweighted. In other words, the edges are all of equal strength, and their lengths in the display are determined only by the layout function.

The conversion of a section of hypertext to a directed graph uses a Perl module to generate a GraphML file from a directory structure. GraphML is a XML-based language that defines a graph using `node` tags to define the nodes in the graph and `edge` tags, with source and target attributes, to define edges between the nodes. The nodes contain `id` attributes that are referenced by the edge source and target attributes.

The directory structure contains a local copy of an online Web hypertext. The documents in the hypertext are downloaded using a Web indexer tool developed as part of the project. Each

[1] `jung.sourceforge.com`

HTML file is converted to a text file and links file by a Perl module (see appendix B). The Text file contains the unformatted textual information contained in the HTML document, and the Links file contains a list of all the hyperlinks contained within the document. These are used to determine the edges between the documents.

The Visualisation tool then reads the GraphML file, and generates a JUNG graph object from it. The graph object is not displayed (it would usually be too large to display effectively as a whole), but stored in memory, and subgraphs are extracted and displayed based on user requests.

C.1.3 Display of Subgraphs

A user can request a subgraph in the following two ways:

- A list of the document nodes is displayed on the right hand side of the current version of the Visualisation tool. If the user selects one or more of these document nodes from the list, a subgraph centred on these nodes is displayed in the display pane.
- The Visualisation tool contains a search field that allows the user to enter a query (in the form of a word or phrase) and receive a list of the documents that contain the query. Elements from this list can then be selected as with the list above.

C.1.4 Searching

Currently the search functionality included in the Visualisation tool is primitive, and is designed only to provide quick access to content-specific subgraphs, i.e. subgraphs centred on documents that contain specific keywords. The term to search for is entered into the text field, and the program finds every document that contains the term, and lists them in the second box on the right.

The *Use Keyphrases* checkbox, when selected, tells the visualisation tool to search the keyphrase files (files that contain keyphrases or keywords for each document), rather than the document text, when searching for occurrences of a term. This will usually give more accurate, but sparser search results. This can only be used when the Web contains annotated keyphrases (such as in the `<meta keywords>` HTML tags) or in conjunction with an automatic keyphrase extractor.

C.1.5 The Nature of the Displayed Subgraphs

The Visualisation tool includes fields that change the parameters of the subgraphs presented in the display pane.

The *depth* field allows the user to choose the number of nodes to display in the subgraph by giving the maximum path length from the selected document to its neighbours. For example, a subgraph of depth 3 will contain all nodes that can be reached from the selected document node

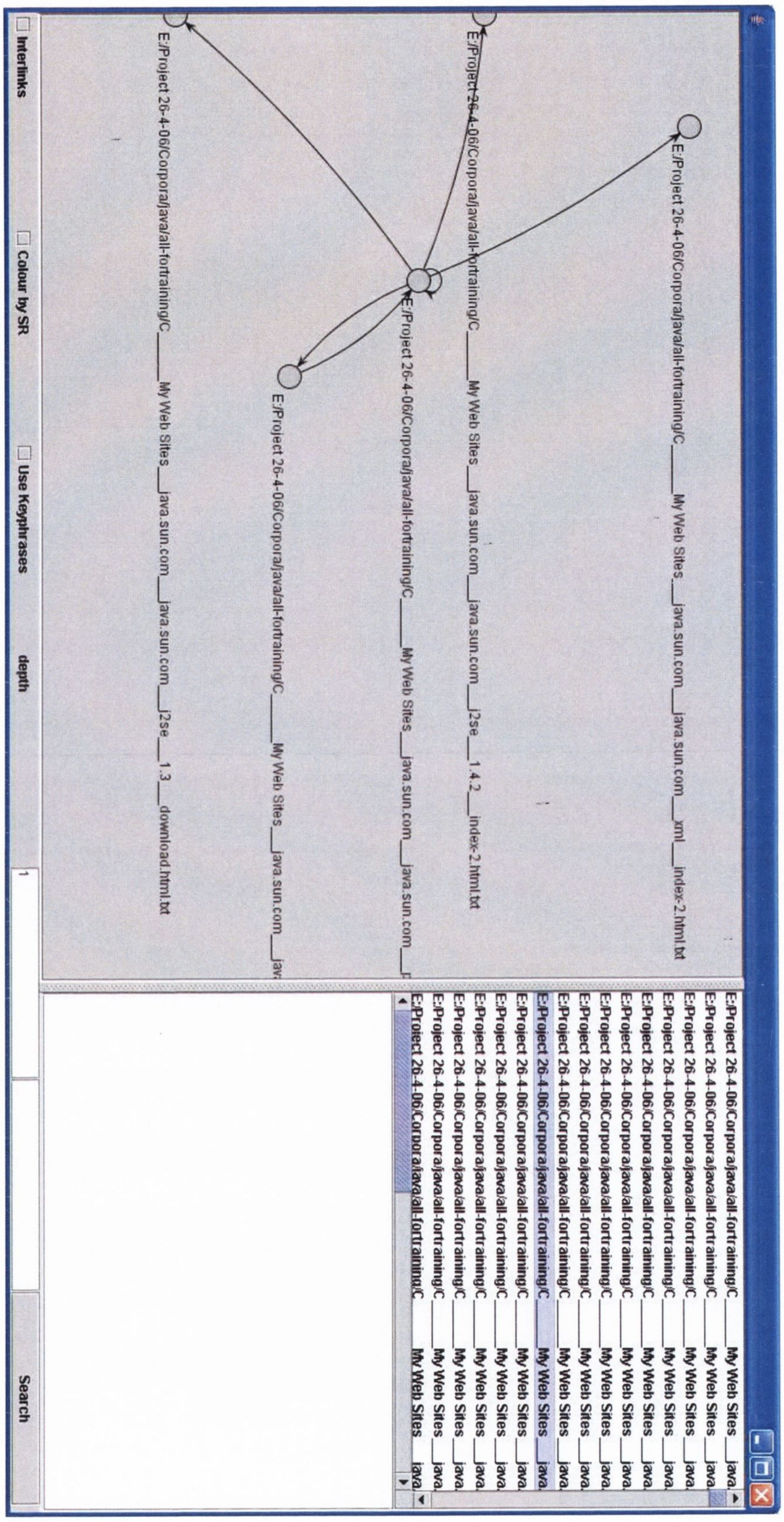

E:/Project 26-4-06/Corpora/java/all-fortraining/C____My Web Sites__java.sun.com__java.sun.com__xml__index-2.html.txt
E:/Project 26-4-06/Corpora/java/all-fortraining/C____My Web Sites__java.sun.com__java.sun.com__j2se__1.4.2__index-2.html.txt
E:/Project 26-4-06/Corpora/java/all-fortraining/C____My Web Sites__java.sun.com__java.sun.com__j
E:/Project 26-4-06/Corpora/java/all-fortraining/C____My Web Sites__java.sun.com__jav
E:/Project 26-4-06/Corpora/java/all-fortraining/C____My Web Sites__java.sun.com__java.sun.com__j2se__1.3__download.html.txt
Interlinks
Colour by SR
Use Keyphrases
depth
1
Search

Figure C.1: A simple subgraph - no query, depth 1

by up to three hyperlinks. A typical depth value is 1 or 2, depending on the connectivity (roughly equivalent to the hyperlink density) of the graph.

The Visualisation tool also includes an *interlinked* checkbox. When checked, the tool will display all the edges between nodes in the subgraph. When unchecked, only edges that form part of a path to the central node are included, and the paths must have length less than or equal to the designated depth. For example, if the designated depth is 2, and an edge exists between two nodes that are both minimum two hyperlinks away from the central node, then that edge would not be displayed, as it would be part of a path of length 3. If the designated depth is 1, only edges attached to the central node would be displayed. This functionality allows the user to have a clear view of the Web in relation to the selected node.

C.1.6 Node Colour Coding

One of the features of the visualisation tool is the ability to assign colours to the document nodes, relative to a search query, according to their contents and contexts. A low score is given a blue colour, and a higher score is given a red colour. Documents which don't contain the search term are coloured grey.

By default, the tool allows two coding schemes:

- Term Frequency (TF): When the "Colour by SR" checkbox is unselected, the nodes are coloured simply according to the number of occurrences of the search term in the document. TF is measured on a logarithmic scale, so the log value of the term frequency is used to assign a colour to the node. The minimum score is log(1), and the maximum is determined when a search query is processed.

- TF×INF: TF×INF measures the ratio of the frequency of the search term in the document and the frequency of the term in the document's context set (its neighbouring documents). For the purposes of TF×INF, only out edges (hyperlinks from the document to neighbours) are used to calculate the ratio. TF×INF is also measured on a logarithmic scale, so the log value of the ratio is taken to assign a colour. The minimum ratio value is currently set at 1/10, but it can be lower. Nodes with a lower TF×INF are coloured black. The maximum, as above, is determined when the query is entered.

C.1.7 Modules

JUNG

JUNG is a Java-based graph visualisation and layout tool that can load GraphML files. For more information, see: http://jung.sourceforge.net/

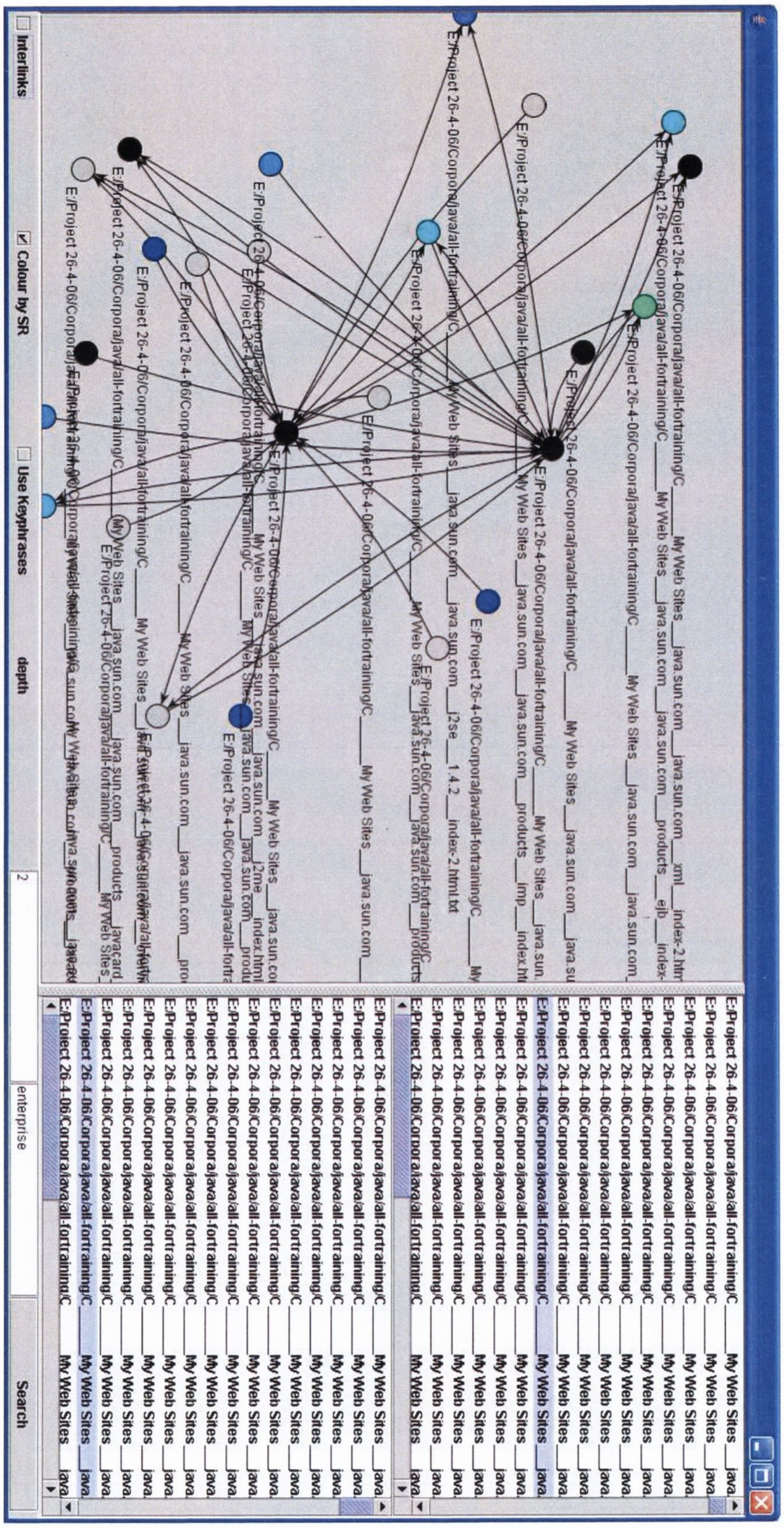
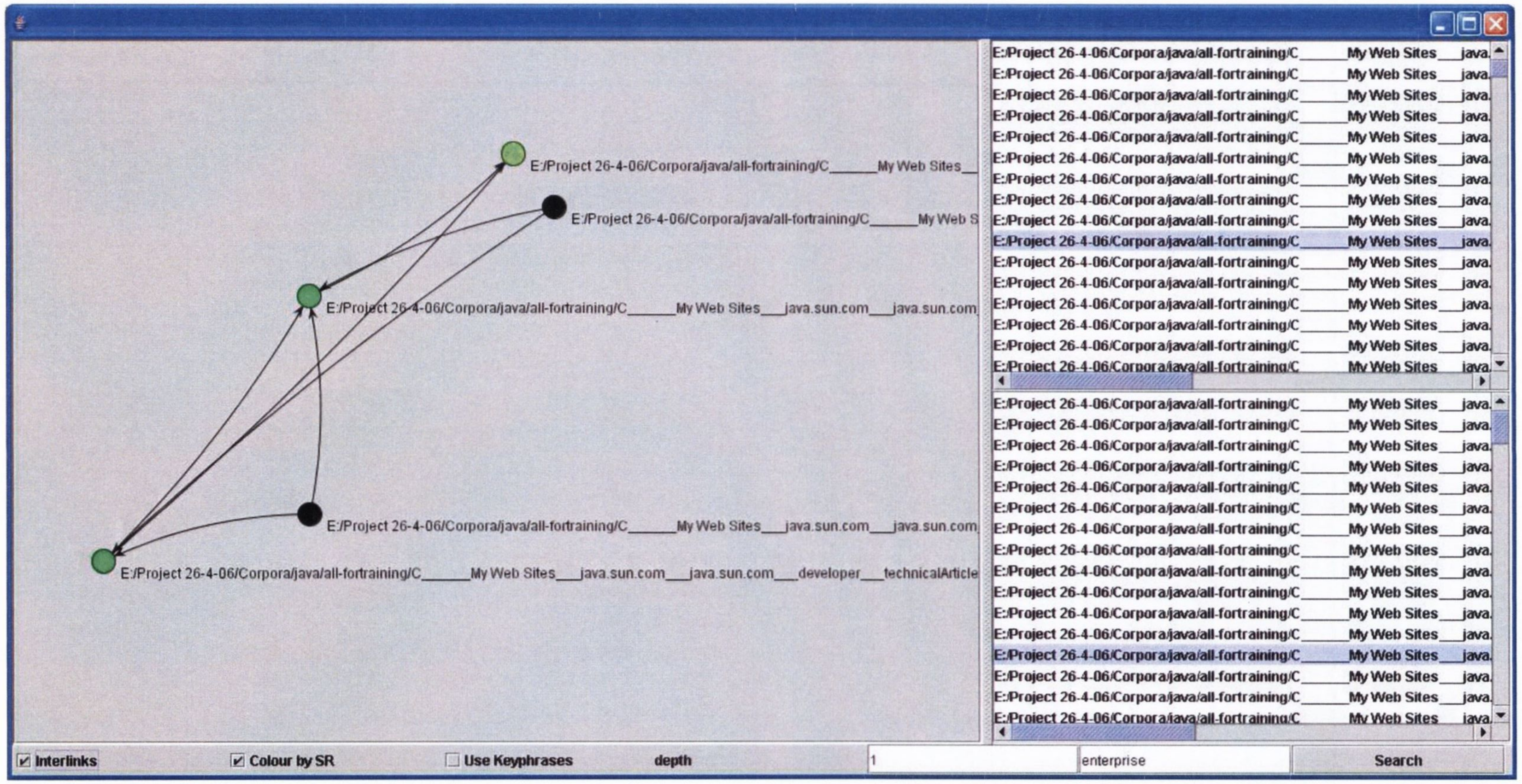


Figure C.2: An example of a subgraph with depth 2


E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites__
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web S
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.sun.com___java.sun.com
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.sun.com___java.sun.com
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.sun.com___java.sun.com___developer___technicalArticle
Interlinks
Colour by SR
Use Keyphrases
depth
Search

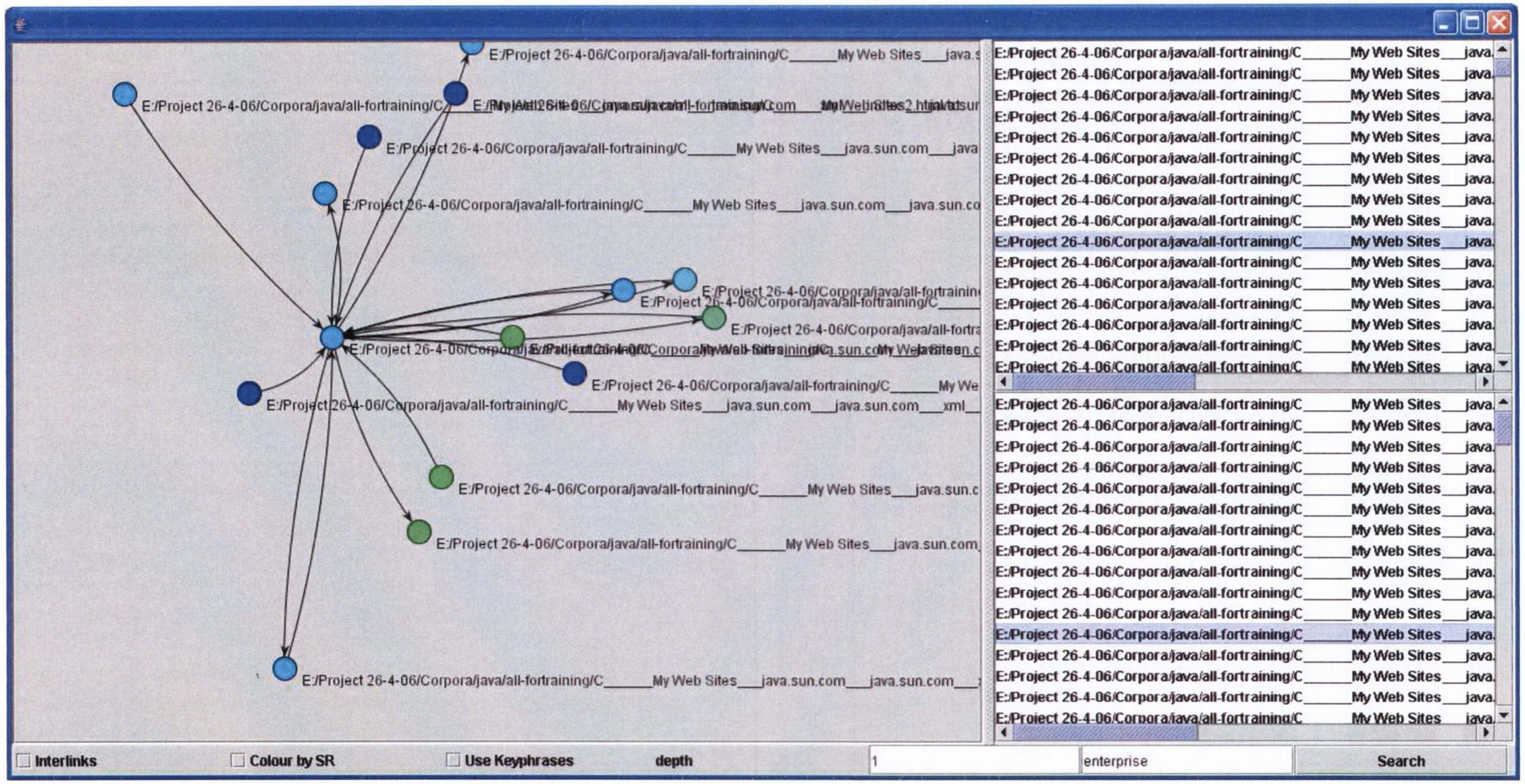
```
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
```

```
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
```

depth: 1

enterprise

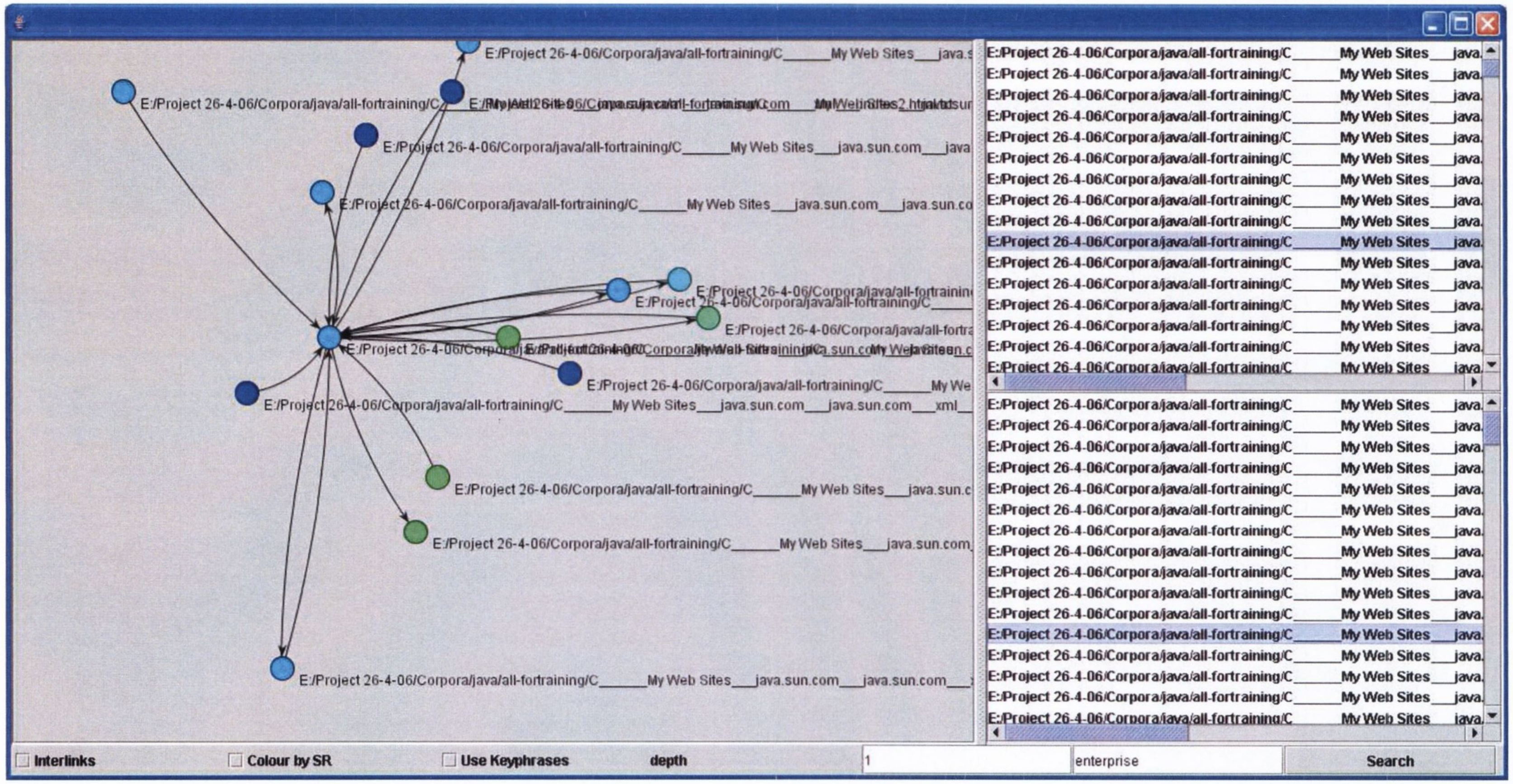
Figure C.3: A subgraph with all edges included


E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.s
E:/Project 26-4-06/Corpora/java/all-fortraining/C______
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.sun.com___java
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.sun.com___java.sun.co
E:/Project 26-4-06/Corpora/java/all-fortrainin
E:/Project 26-4-06/Corpora/java/all-fortraining/C
E:/Project 26-4-06/Corpora/java/all-fortra
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My We
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.sun.com___java.sun.com___xml
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.sun.c
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.sun.com
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.sun.com___java.sun.com
Interlinks
Colour by SR
Use Keyphrases
depth
1
enterprise
Search

```
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
```

```
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
```

Figure C.4: Query = “Enterprise”, coloured by TF


E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.s
E:/Project 26-4-06/Corpora/java/all-fortraining/C
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.sun.com___java
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.sun.com___java.sun.co
E:/Project 26-4-06/Corpora/java/all-fortraining/C
E:/Project 26-4-06/Corpora/java/all-fortraining/C______
E:/Project 26-4-06/Corpora/java/all-fortra
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My We
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.sun.com___java.sun.com___xml
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.sun.c
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.sun.com
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.sun.com___java.sun.com
Interlinks
Colour by SR
Use Keyphrases
depth
1
enterprise
Search

```
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
```

```
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
E:/Project 26-4-06/Corpora/java/all-fortraining/C______My Web Sites___java.
```

Figure C.5: Example of TF×INF colouring

fileconverter.pl

The fileConverter Perl script carries out the conversion from HTML page to text and link files. If the page contains a `<meta keyword...>` tag in its head then a .key file will be generated to store a list of the annotated keywords it contains.

keaToGraphMLRecursiveDirectories.pl

This module analyses a directory structure on the file system and generates a GraphML file that describes the downloaded section of the Web in the form of a graph of document nodes with hyperlink edges between them.

C.1.8 Future Work

The visualisation tool is currently in prototype stage, and can be improved and augmented in a number of ways:

- Allow a user to open a page simply by clicking on the node in the graph.
- Allow weighting of edges, i.e. certain edges will be stronger or weaker according to criteria such as a variation of TF$\times$INF or non-content-specific criteria such as URL similarity. Currently, the edge lengths are determined only by the JUNG layout manager.
- Change the way in which JUNG sets the layout of graph. Currently JUNG is set to use an iterative system to spread the graph to the outer reaches of the space. Other techniques can be used to allow the user to easily see any hierarchy within the Web, for example.
- Attach the tool to a search engine to create and rank the result set. The tool could be a useful addition to a search engine, and the use of a search engine would be an ideal replacement for the currently primitive inbuilt search functionality in the tool.
- Create a dynamic version of the tool that can be included in a Web browser, and performs analysis of the user's current position in the Web on the fly, in the background, to give the user a clue about the content of nearby documents.

C.1.9 Conclusion

This report describes a prototype Web visualisation tool that uses JUNG graph display software to present a section of the World Wide Web as a graph of document nodes and hypertext edges. Nodes containing specific terms can be retrieved using a primitive search tool, and nodes can also be colour-coded according to content and context-related measurements. Context is defined as the content of neighbouring documents in the Web, where document neighbours are defined as documents connected to a document by hyperlinks. The tool is intended to be included as part of a search engine or as a Web browser add-on to give a visual idea of the content of nearby documents in the Web.

C.2 Threaded ModNLP Indexer

C.2.1 Purpose of Application

The ModNLP Web Indexer harvests HTML documents from a localised subset of the World Wide Web and stores their contents in an index for future reference or analysis. The indexer was designed to be used in conjunction with a search engine so the following functionality is included:

- Web documents, once processed, need not be kept on the server, if disk space is scarce.
- Hyperlinks in the documents are retained and stored by the indexer.
- Word positions are stored in the index, as these are required to allow the search engine to provide contextual information for each search result.
- The search engine is cross-platform compatible, so the indexer is also required to be cross-platform compatible, and is programmed using Java and Perl.

C.2.2 Description of Application

The ModNLP Indexer is a multi-threaded, top-down Web harvester and indexer. The default parameters for the program are a start page, the maximum path length from the root to documents to be indexed (the depth) and the number of concurrent threads.

Each thread is a form of spider agent that downloads a page from the Web, extracts the links from that page and adds the links to the queue of pages to download.

The majority of the time cost of the application is taken up with downloading the required document from the Web. The speed of this operation is often limited by the speed of the Web server that hosts the document, as opposed to the bandwidth available to the indexer, so increasing the number of threads running concurrently allows multiple document requests to be carried out at the same time, thus reducing this time cost.

C.2.3 Application Operation

The actions of each spider can be summarised into three distinct steps:

1 Downloading

The page is downloaded to temporary storage by an external Java component (LinkGetter).

2 Conversion

The downloaded version of the page is split into text and link files by a Perl program. The text file contains the textual content of the HTML page (all markup information and links are removed). The links file contains a list of the targets of all the href links (contained in `<link>` and `<a>` tags) contained in the document.

3 Processing and Indexing
The text file is tokenised (converted to a list of phrases and words), and the information is added to the index. The link files used to add new pages to the queue of pages to be processed, and are stored in an internal copy of the Web structure, so that they can be added to the index during the post-processing stage. Finally, the page URL is stored in a list of processed pages, to ensure that it is not passed over again by a different spider, or the same one following a loop.

C.2.4 Application termination and post-processing

The Web spiders will terminate when all the documents up to the desired depth from the start page have been added to the index. The final step is to add the link structure, stored in the internal copy of the Web, to the index. This is required in order to obtain the contextual information required by the model builder and the search engine.

C.2.5 Application Versions

There are two versions of the ModNLP Indexer, the Web-based version and the command-line version.

Version 1: Command Line

The command-line version of the application is in the form of a collection of Jar files and Perl and Shell scripts. The advantage of the command-line version is that it is easier to install than the Web version, and it has fewer system requirements. However, this version of the program can only be operated and supervised locally, i.e. by one user logged in to one machine, as opposed to the distributed Web version. Another disadvantage of the command-line version is that it currently has no user interface associated with it, so threads cannot be added or removed, and the user cannot easily supervise the operation of the indexer.

Version 2: Web-based Version

The Web version uses J2EE servlet technology to allow the user to remotely run and supervise the indexer. This version of the indexer is packaged as a WAR file, separate Jar files and Perl and Shell scripts. After installation on the Web server, the indexer can be initialised and run remotely from any machine. This is helpful in the context of the search engine experiment, as the training system and the search systems are also accessible from any machine. The Web version contains a simple user interface, operated from a Web browser, so it is easier to use than the command-line version.

C.2.6 Components

- linkgetter.jar

  The LinkGetter Jar is an independent module that downloads a Web page and stores it in a specified location. Only the HTML text of a Web page is downloaded. Embedded images or other media are not considered . The LinkGetter module is compatible with http proxy servers and any proxy information, including authentication information is stored in XML form in `params.txt`.

- fileConverter.pl

  The fileConverter Perl script carries out the conversion from HTML page to text and link files described in step b above. If the page contains a `<meta keyword...>` tag in its head then a .key file will be generated to store a list of the annotated keywords it contains.

- search.war

  This is the Web archive that contains the servlets for the indexer (as well as those for the search engine), and includes the Jars below.

- srsearch.jar

  This is the main Jar for the program that contains the indexer classes themselves.

- configHandler.jar

  This Jar contains utility classes for use with XML configuration files.

- je.jar, idx.jar, gnu-regexp-1.1.4.jar

  These Jars contain classes that make up the ModNLP index application. The ModNLP index is an adaptation of the Berkeley Database for use with natural language processing applications.

C.2.7 Installation & Use

Web Application

1 Extract the zip file in the root directory of the servlet application.

2 Open config.xml in a text editor and edit the following:

  - Proxy information
  - Search and feedback URLs (change hostname and port number)

3 Deploy the WAR file on the web server.

4 Go to `/search/IDXIndexMaker?report`

5 Call the following commands on the UI:

- Create (creates the index object)
- Depth (sets the depth parameter of the indexer)
- Domain (sets the domain of the indexer. Spiders can index pages that have URLs beginning with this domain)
- Index (gives the start page of the indexer and starts the first spider)

6 Add more threads. (This will speed up the indexing process)

Command-line

1 Extract the zip file

2 Open config.xml in a text editor and edit the following:

(a) Proxy information

(b) Search and feedback URLs (change hostname and port number)

3 Call the command:

```
java -classpath lib/idx.jar:lib/je.jar/gnu-regexp-1.1.4.jar srsearch.jar <start>
<depth> -n <numThreads> -v <verbosity level 0-4> -l filelist.txt
```

C.2.8 System Requirements

General

- Java v1.4.2
- Perl v5.8.8
- 256mb RAM

C.2.9 Web Version

- J2EE enabled server, e.g. Tomcat 4

C.2.10 Current Limitations & Future Work

- MIME types from HTTP headers are not yet read by the indexer. The indexer therefore attempts to download all files regardless of their file type, other than explicitly hard-coded exclusions determined by Windows file extensions. Analysis of the HTTP headers would prevent the indexer from attempting to analyse unsuitable files.

- The cut-off criterion for the indexer is depth only at the moment. Depending on the connectivity and density of the web being analysed, this can give an unpredictable number of pages to be indexed, which results in the indexing procedure taking an unpredictable amount of disk space and time. An alternative would be to set a limit on the maximum number of pages or dictionary size in megabytes.
- There is no security protection on web user interface. This would be required if the indexer were to be made widely available in its current form.
- The LinkGetter module is currently not compatible with the secure HTTP protocol. Therefore the indexer cannot process any URLs with the `https:` prefix.

C.2.11 Workflow

Initialisation

1 The following settings are initialised from the config.xml file.

   (a) Proxy information

   (b) File to store the list of indexed pages

2 The IDX Dictionary object is created.

3 A Queue object to contain the list of pages to be indexed, is initialised. This is a first-in, first-out queue, with newly found pages added to the end of the queue (breadth-first search).

4 A Vector object is created to contain the list of indexed pages.

5 The first spider is created to index the start page.

Indexing Spider

Each spider thread follows the following control flow

Algorithm C.1: Indexing

```
1  Index:
2    loop
3      if the spider has received an instruction to stop
4        tell the indexer that the spider has stopped
5        exit loop
6      else pick the next page from the queue
7      if this is equal to the first of a series of pages that were
8      skipped for being over the maximum depth
9        tell the indexer that the spider has stopped
10       exit loop
11     else if the page has already been indexed
12       ignore this page
13     else if the page exceeds the maximum depth
14       if the last page did not exceed the maximum depth
15         set the current page as the first in a series of skipped pages
16       add this page to the end of the queue
17     else
18       download the page
19       split it into its text and link components
20       add the textual component to the index
21       for each line in the links file loop
22         create a new page object from the URL
23         add a reference to the new page in the link vector of
24         the initial page
25         if the new page is already in the queue, or in the vector of
26         indexed pages
27           if the depth of the page in the queue is greater than the
28           new depth, which is the current depth of the spider + 1
29             reduce its depth
30             recursively carry out 28 and 29 for each of
31             the links of this page, with the new depth incremented
32             by one each time
33         else, add the page to the queue, with a depth of the current
34         depth of the spider + 1
35       end
36   end
```

Algorithm C.2: Termination

```
On termination:
  The queue of non-indexed pages is serialised and saved to disk
  A list of indexed pages is saved as a text file,
  The links between documents (stored in the link
  vectors of a page object)
  are stored in the links database,
  The databases are closed, all data in memory is flushed to disk
```

Appendix D

Corpus Analyses

D.1 Comparison of Document Size to TF×INF KEA effectiveness

In this section, the effectiveness of the simple TF×INF formula:

$$\text{TF}\times\text{INF} = \frac{tf}{nf}$$

is investigated in relation to the size of the documents it affects. Table D.1 shows the mean and standard deviation values of log document sizes (in bytes) grouped according to whether TF×INF KEA returns a greater or smaller number of correct keyphrases than standard KEA for each document[1]. An analysis of variance was carried out on these data in order to decide whether the differences in the means are statistically significant, and the results are shown in table D.2. For two out of the four corpora (No10 and BigJava), the means were found to be statistically different. This is consistent with the confidence intervals of each mean shown in figure D.1. In the other two corpora (Small Java and Oireachtas) the means are not significantly different, allowing us to conclude that while there is some evidence that TF×INF performs slightly better on smaller documents, this effect is not consistent. Normalising the TF component of TF×INF by dividing by document length (in phrases) results in poorer KEA scores as demonstrated by figure D.2. The effect of document size on TF×INF is therefore ignored in this analyis.

D.2 Comparison of link frequency to TF×INF KEA effectiveness

In this section, the relationship between TF×INF KEA scores and link frequency is discussed. The link frequency measure used here is the reciprocal of Link Density (Almind and Ingwersen,

[1]50% sample taken from each corpus and used as test set. The remaining 50% is used to train the KEA model

Table D.1: Means (and standard deviations) of log (base 10) document sizes in each corpus, grouped according to the effect that TF×INF has on keyphrase extraction of the document. +: TF×INF KEA returns more correct keyphrases than standard KEA. −: TF×INF KEA returns fewer correct keyphrases than standard KEA.

Corpus	TF×INF effect	mean (sd)	count
Small Java	All	3.887738 (0.3485326)	126
	+	3.898032 (0.3547606)	37
	−	3.996027 (0.4062680)	12
No10	All	3.663664 (0.2758740)	166
	+	3.545242 (0.2906066)	39
	−	3.761992 (0.1778436)	15
Oireachtas	All	3.623494 (0.3658239)	452
	+	3.610348 (0.2601024)	92
	−	3.715244 (0.3016441)	35
Big Java	All	3.811893 (0.3901865)	871
	+	3.665901 (0.2850054)	358
	−	4.016178 (0.4329574)	25

Table D.2: One-way ANOVA of document size against TF×INF KEA effect (statistical significance denoted by *)

Source	DF	SS	MS	F	P
			Small Java		
Treatment	0.128504075	2	0.064252037	0.513282492	0.599441542
Error	21.53073716	172	0.125178704		
Total	21.65924124	174			
			No10		
Treatment	0.647031562	2	0.323515781	4.330962845	***0.014310561**
Error	16.20954207	217	0.074698351		
Total	16.85657363	219			
			Oireachtas		
Treatment	0.284160957	2	0.142080479	1.144314162	0.319224769
Error	66.05425074	572	0.124162125		
Total	66.3384117	578			
			Big Java		
Treatment	6.90013742	2	3.45006871	26.00792016	***8.57853E-12**
Error	165.9508307	1251	0.132654541		
Total	172.8509682	1253			

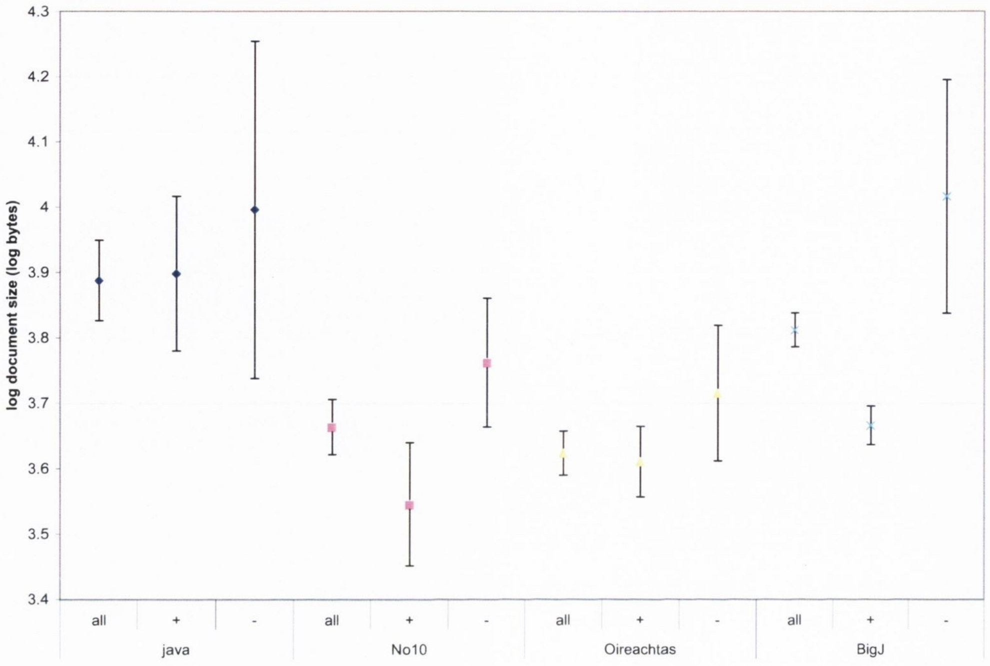


Figure D.1: Confidence intervals of means reported in D.1

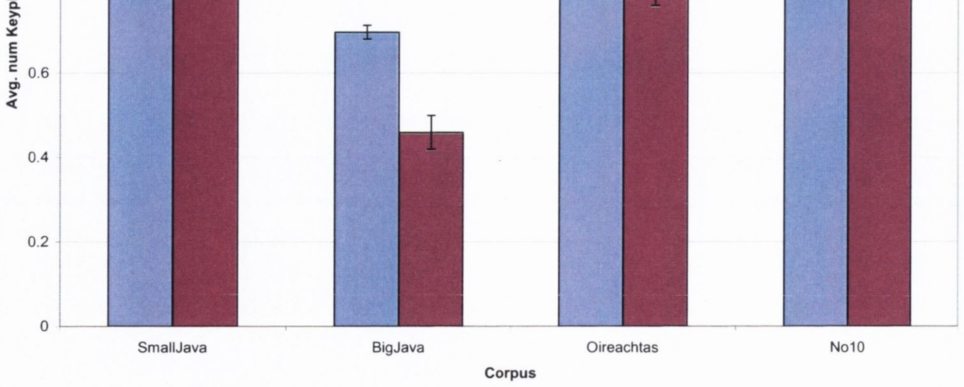


Figure D.2: Results of KEA using normalised TF×INF

1997), and is defined as the number of out-links a document contains, divided by its size. Figures D.3–D.14 show the link frequencies of each of the corpora used in the KEA experiments, grouped as in section D.1.

The data are not normally distributed, but a notable difference can be seen between the variances of the "positive" charts and the smaller variances of the "negative" charts. This suggests that the performance improvement due to TF$\times$INF is most evident at times when the link frequency deviates from the mean. In particular, when the link frequencies are high, TF$\times$INF is, predictably, more likely to find correct keyphrases. However, at low link frequencies in the Small Java and Oireachtas corpora, TF$\times$INF also performs well. TF$\times$INF is therefore effective in keyphrase extraction even when a small number of links are available.

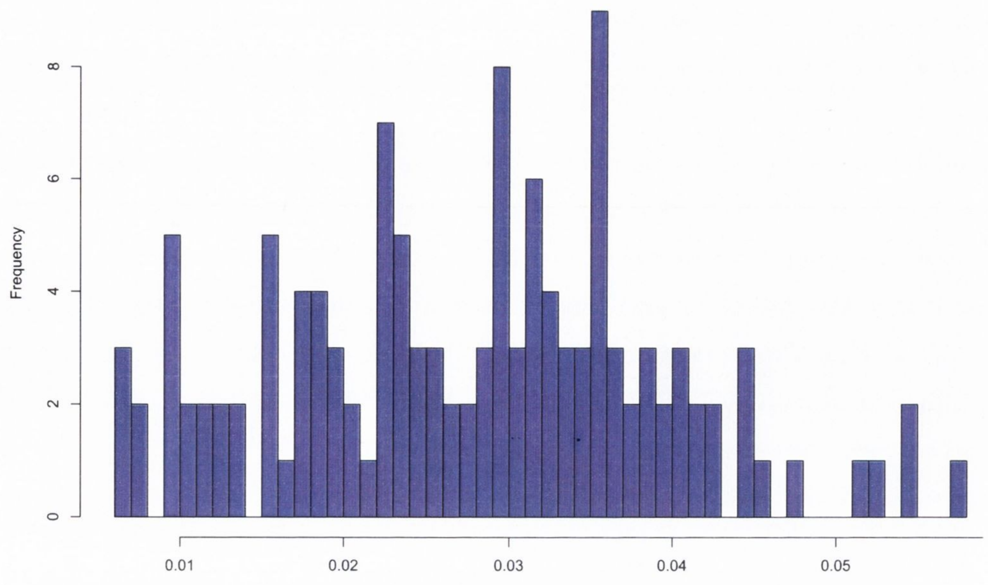


Figure D.3: Small Java [all]: Histogram of link frequencies of all documents

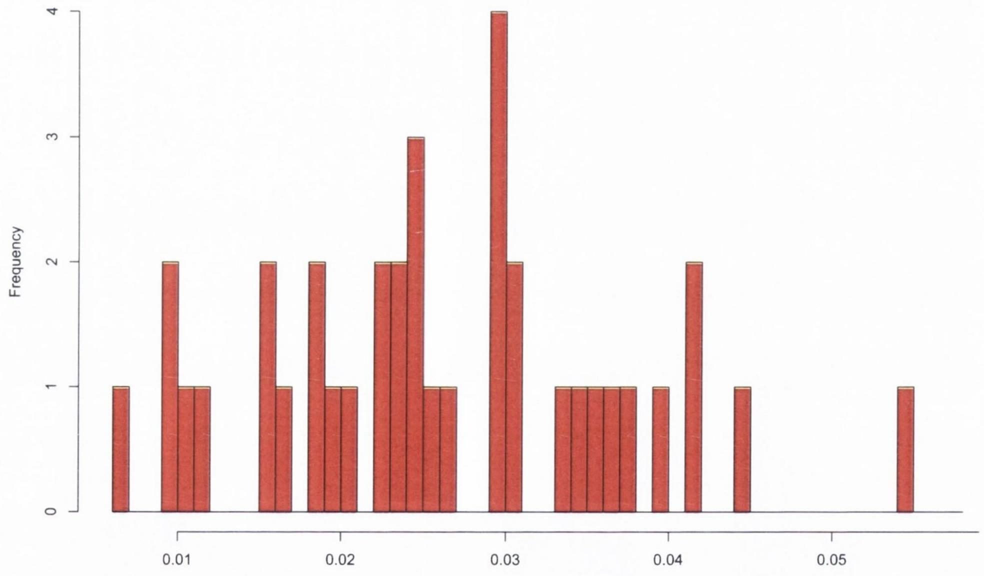


Figure D.4: Small Java [positive]: Histogram of link frequencies of documents for which TF × INF KEA finds more correct keyphrases than standard KEA

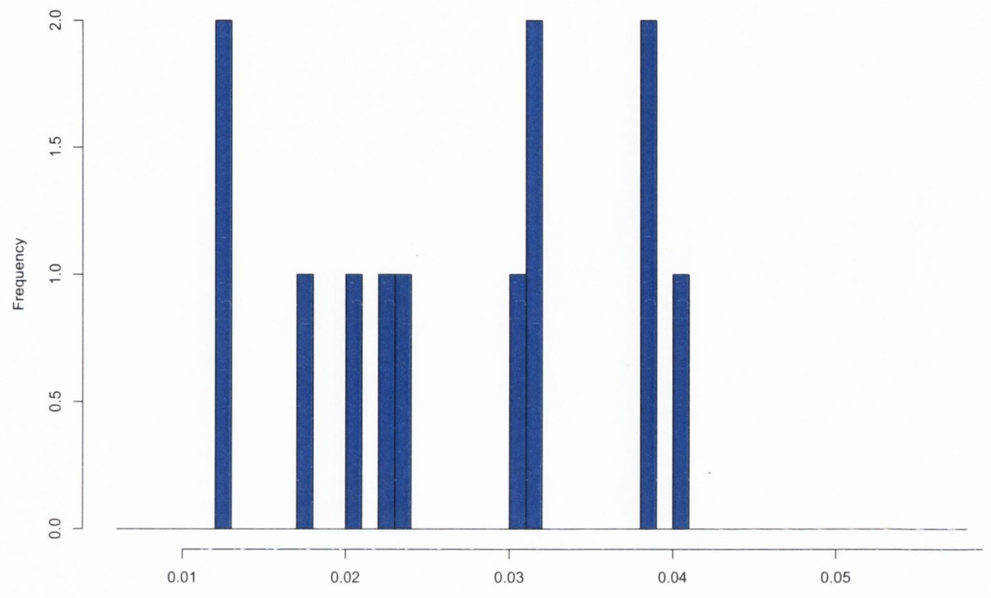


Figure D.5: Small Java [negative]: Histogram of link frequencies of documents for which TF×INF KEA finds fewer correct keyphrases than standard KEA

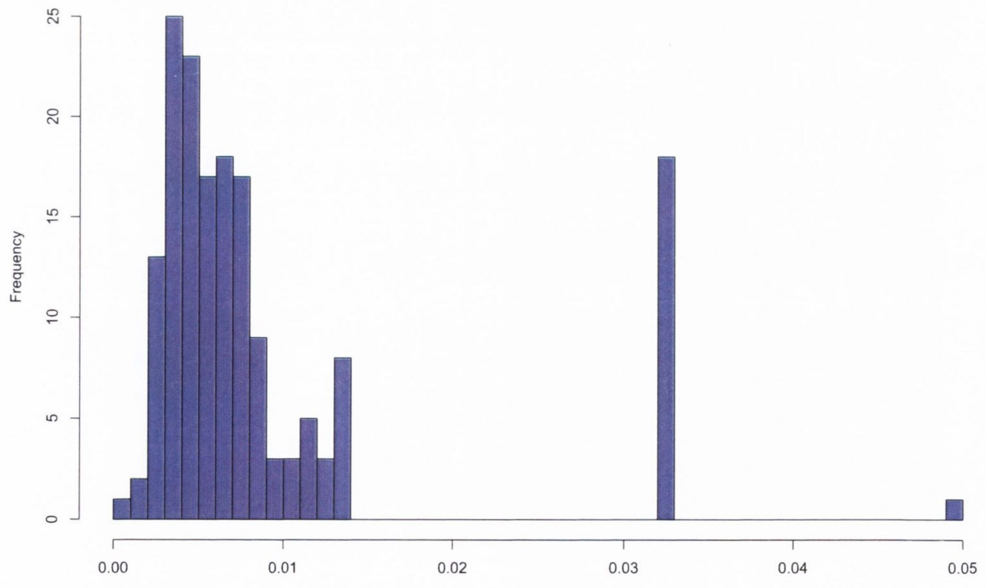


Figure D.6: No10 [all]: Histogram of link frequencies of all documents

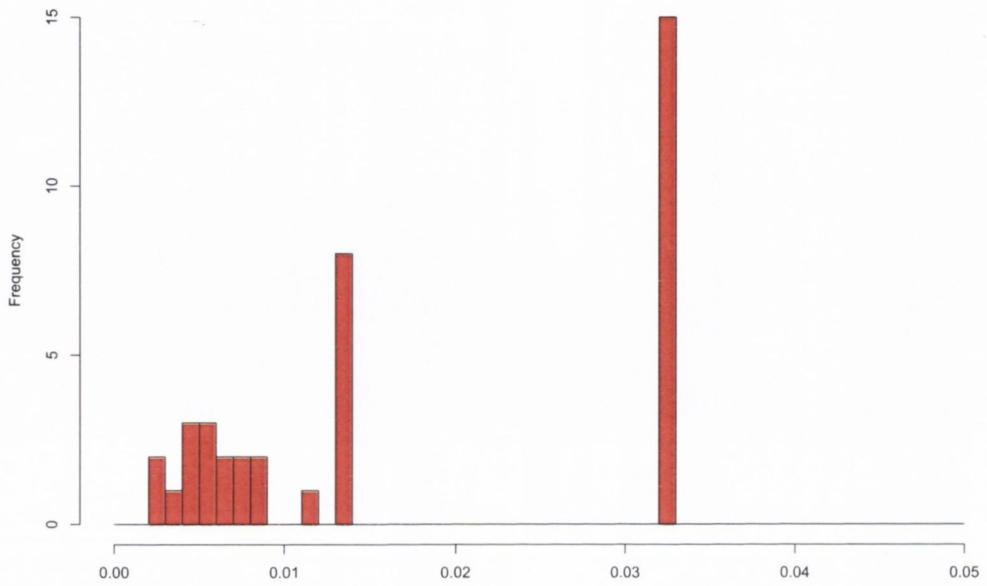


Figure D.7: No10 [positive]: Histogram of link frequencies of documents for which TF$\times$INF KEA finds more correct keyphrases than standard KEA

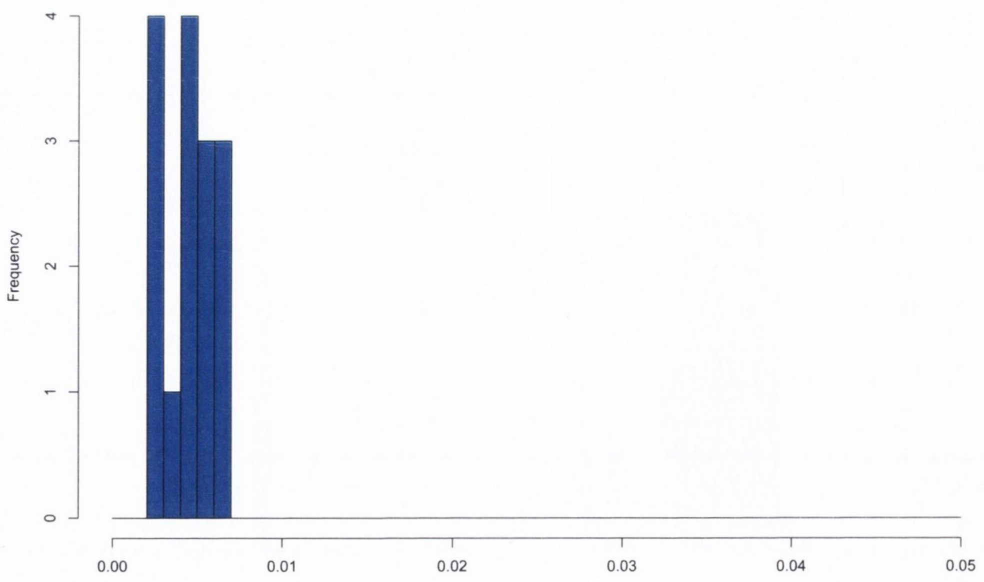


Figure D.8: No10 [negative]: Histogram of link frequencies of documents for which TF$\times$INF KEA finds fewer correct keyphrases than standard KEA

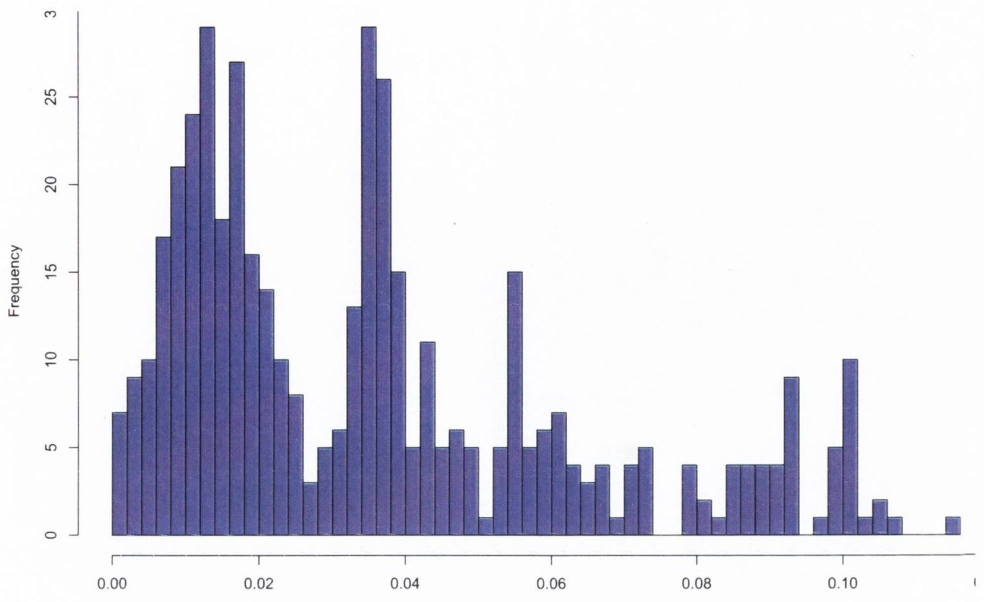


Figure D.9: Oireachtas [all]: Histogram of link frequencies of all documents

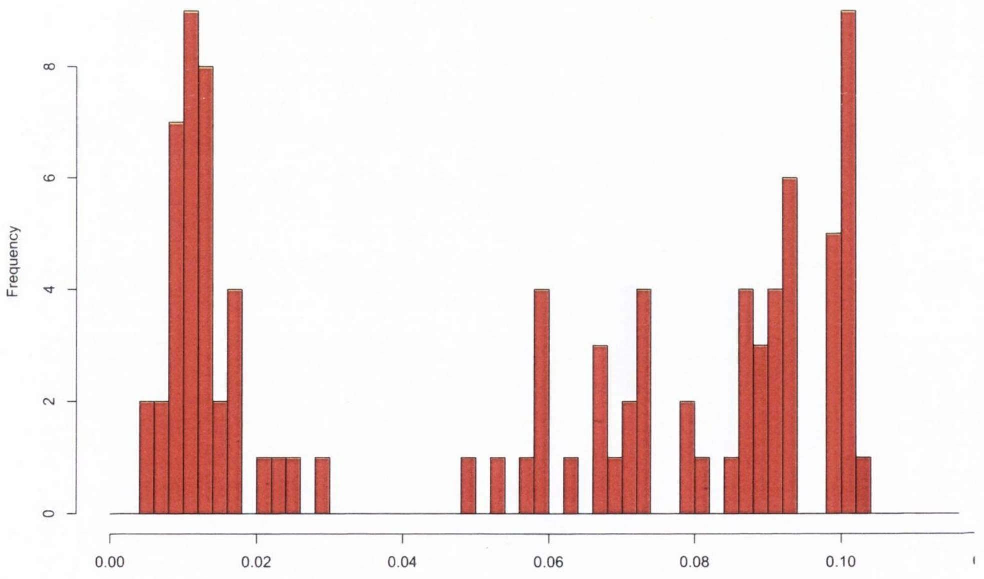


Figure D.10: Oireachtas [positive]: Histogram of link frequencies of documents for which TF×INF KEA finds more correct keyphrases than standard KEA

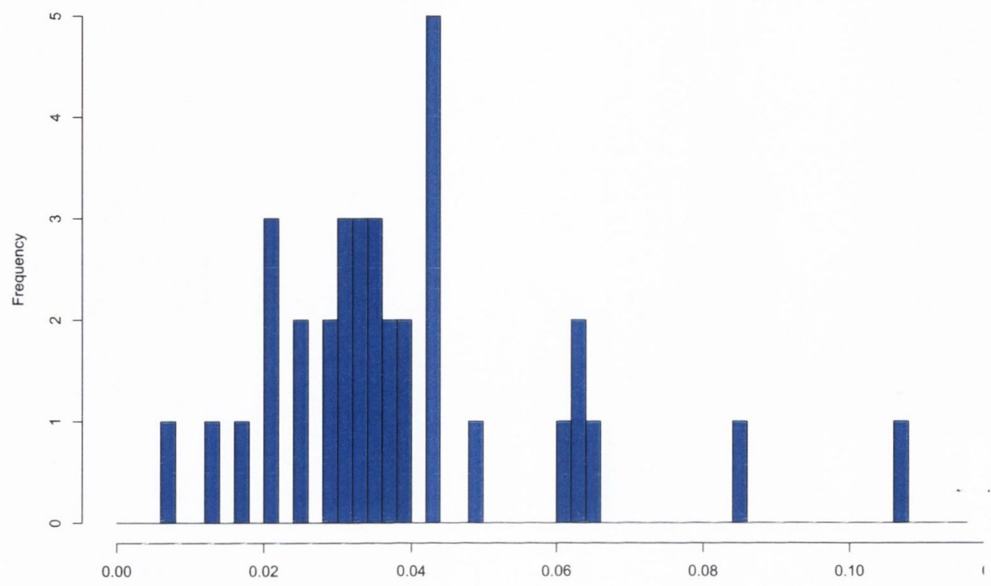


Figure D.11: Oireachtas [negative]: Histogram of link frequencies of documents for which TF×INF KEA finds fewer correct keyphrases than standard KEA

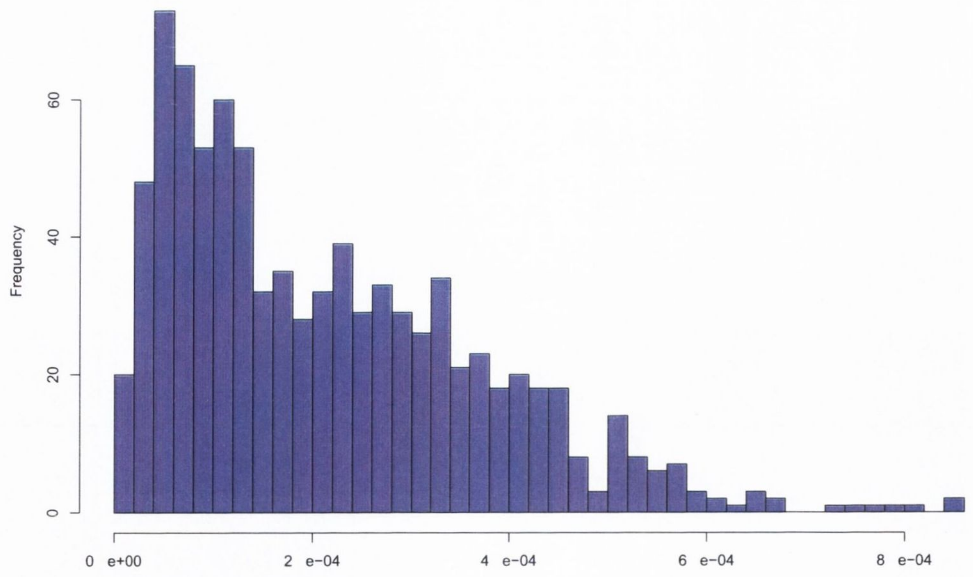


Figure D.12: Big Java [all]: Histogram of link frequencies of all documents

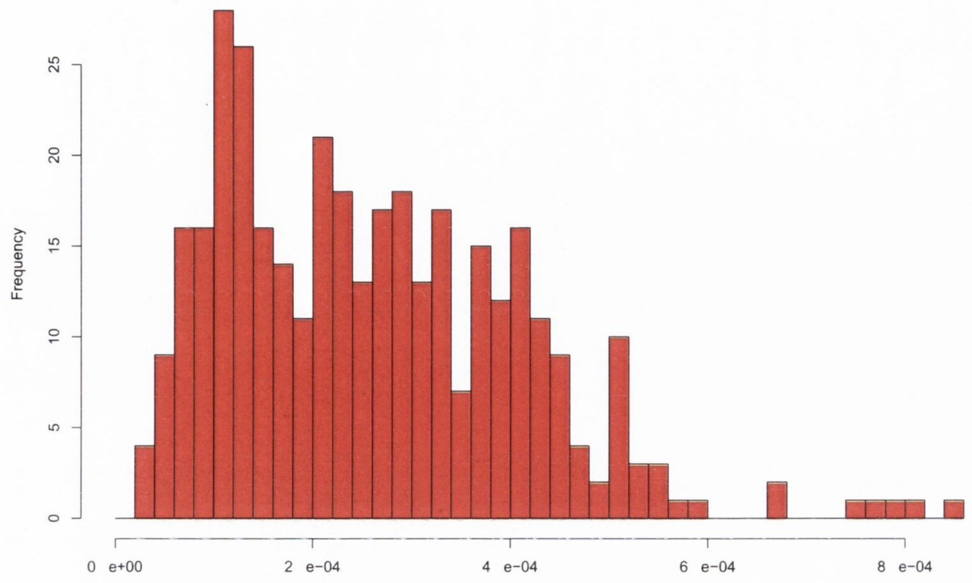


Figure D.13: Big Java [positive]: Histogram of link frequencies of documents for which TF×INF KEA finds more correct keyphrases than standard KEA

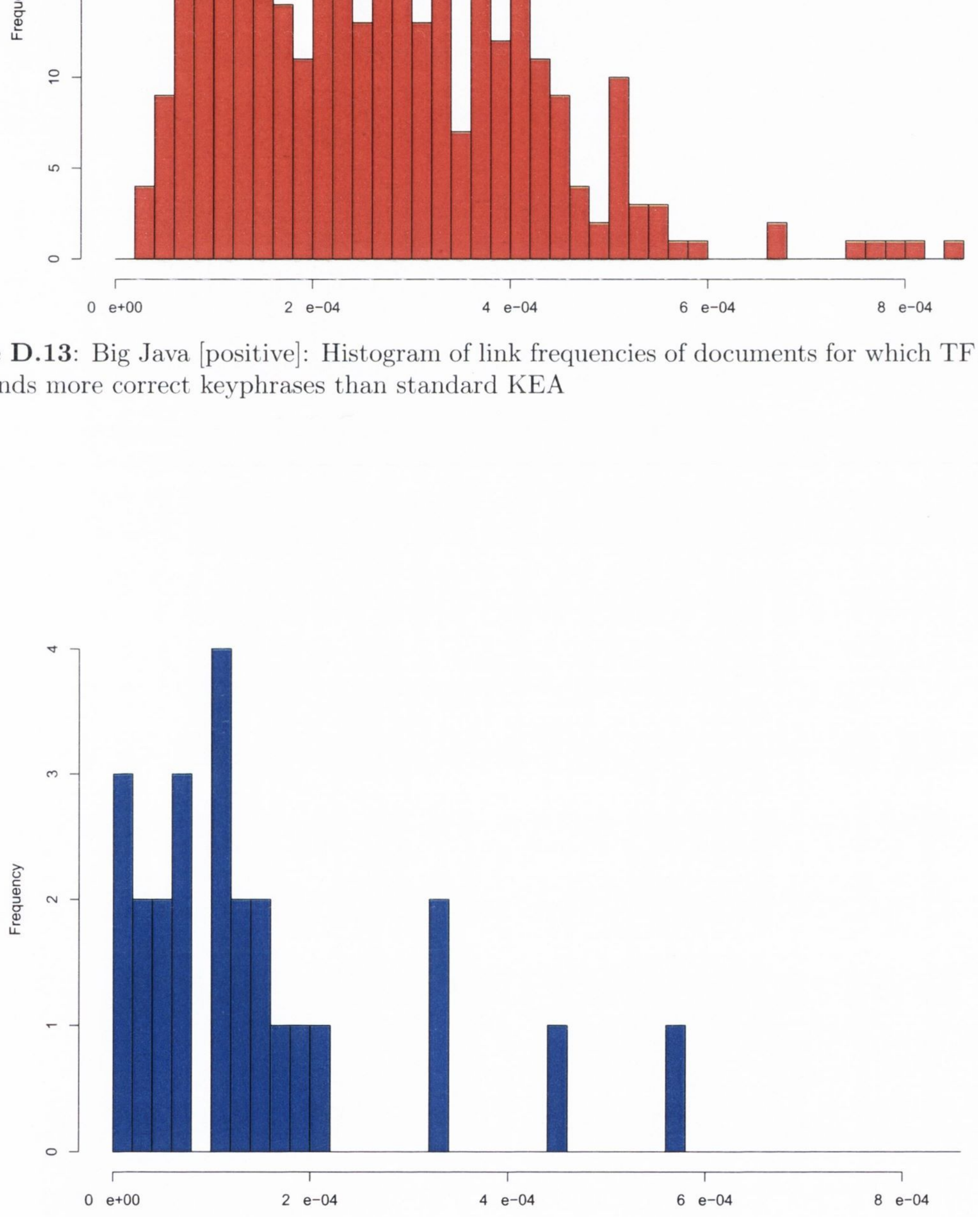


Figure D.14: Big Java [negative]: Histogram of link frequencies of documents for which TF×INF KEA finds fewer correct keyphrases than standard KEA