EVALUATION OF LINKAGE-BASED WEB DISCOVERY SYSTEMS

by

Cathal G. Gurrin

A dissertation submitted in partial fulfillment
of the requirements for the Ph.D. degree

Supervisor : Prof. Alan F. Smeaton

School of Computer Applications
Dublin City University

September 2002

DECLARATION

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Doctor of Philosophy in Computer Applications, is entirely my own work and has not been taken from the work of others save where referenced.

Signed: 

Cathal Gurrin

Date: 26th September 2002

ACKNOWLEDGEMENTS

Firstly, I'd like to thank my supervisor, Alan Smeaton for his assistance and guidance over the course of my four years working on this dissertation. I was fortunate to have worked with Alan and had him as my supervisor and would likely not have got this far without his guidance and unfailing enthusiasm.

Secondly, I'd like to thank friends in the postgrad lab, although people come and go there are a number that I must thank; Jer, Paul, Jiamin, Hyowon, Derek, Aidan but especially Tom and Kieran for discussion and advice. I would also like to thank the other postgrads, some of whom are here no longer, who, if nothing else helped me to keep a grasp on reality; Jer, Chríostaí, Stephen, Qamir, Saba, and the girls: Aoife, Mairead, Mary, Michelle and Nano.

Of course, I must thank both Martin and Amit whom I had the pleasure of working with at AT&T. My time spent there instilled me with confidence for the rest of my research. I do appreciate the opportunity I was given. Thanks guys!

I would also like to my deepest appreciation to my family, especially my mother who was always there when I needed especially when things were going wrong. Mother, at last you can rest-assured that I have finished college!

Finally, I must acknowledge the significant input of Jean, my girlfriend. I would like to thank her for her patience and understanding when I spent all my time either in the university or reflecting about my work at home. She is the only person that can truly appreciate the effort required to compete this dissertation.

Of course, I could never have completed this work without funding and I would like, in particular, to acknowledge the following:

- Enterprise Ireland Informatics Directorate for part-funding my studentship.
- School of Computer Applications for part-funding my studentship and for employment for the last year.
- Hewlett-Packard Europe for support through a gratefully received equipment donation. In retrospect, the equipment was essential for the completion of the work in this dissertation.
- BCS for twice funding my attendance at IRSG and CEPIS-IR for funding my attendance at ESSIR 2000.

Abstract

In recent years, the widespread use of the WWW has brought information retrieval systems into the homes of many millions people. Today, we have access to many billions of documents (web pages) and have (free-of-charge) access to powerful, fast and highly efficient search facilities over these documents provided by search engines such as Google. The "first generation" of web search engines addressed the engineering problems of web spidering and efficient searching for large numbers of both users and documents, but they did not innovate much in the approaches taken to searching.

Recently, however, linkage analysis has been incorporated into search engine ranking strategies. Anecdotally, linkage analysis appears to have improved retrieval effectiveness of web search, yet there is little scientific evidence in support of the claims for better quality retrieval, which is surprising. Participants in the three most recent TREC conferences (1999, 2000 and 2001) have been invited to perform benchmarking of information retrieval systems on web data and have had the option of using linkage information as part of their retrieval strategies. The general consensus from the experiments of these participants is that linkage information has not yet been successfully incorporated into conventional retrieval strategies.

In this thesis, we present our research into the field of linkage-based retrieval of web documents. We illustrate that (moderate) improvements in retrieval performance is possible if the underlying test collection contains a higher link density than the test collections used in the three most recent TREC conferences. We examine the linkage structure of live data from the WWW and coupled with our findings from crawling sections of the WWW we present a list of five requirements for a test collection which is to faithfully support experiments into linkage-based retrieval of documents from the WWW. We also present some of our own, new, variants on linkage-based web retrieval and evaluate their performance in comparison to the approaches of others.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCING INFORMATION RETRIEVAL

This chapter will define the context within which we are working by briefly describing Information Retrieval and to a lesser extent DataBase Management Systems. We then discuss common term weighting strategies before we describe the architecture of a simple information retrieval system. We distinguish between the two broad types of retrieval system found on the WWW and finally, we describe the objective of the research undertaken and presented in this thesis.

1.1 INTRODUCTION TO INFORMATION RETRIEVAL

Information Retrieval (IR) has been receiving increasing levels of attention since the end of the Second World War. In the immediate aftermath of the war, the US government began to pump huge amounts of money into research and development with a corresponding increase in the volume of scientific literature being produced [Garfield, 01]. The need to have search and retrieval facilities provided over this literature, along with a growing dissatisfaction with the then current manual processes and the hope that automation might hold the answers led to the development of a new field of research, which we now call Information Retrieval.

In principle, the problem of information storage and retrieval is simple [van Rijsbergen, 79]. If a person has an information need that can be fulfilled from reading each document in a given set of documents, retaining documents containing relevant material and discarding all others, this is called manual information retrieval. Manual information retrieval is clearly impractical in the majority of cases. In a library scenario, an individual may be seeking information contained between the covers of only a handful of books contained within a vast library. A person has neither the time nor the inclination to read a whole document collection to fulfill an information requirement. Therefore, it follows that the advent of computer technology from the mid-40s onwards posed possibilities for automatic information retrieval as opposed to manual information retrieval. Over half a century onwards, we are still grappling with the problems of effective information retrieval, although issues of scale have been all but solved for reasonable (in the billions) sized document collections.

1.2 What is Information Retrieval?

Information Retrieval is the name given to a process that stores, retrieves and provides maintenance functions over some body of information. Information in this context can be composed of text, images, audio, video and other multimedia objects. As mentioned, IR may be either manual or automatic, with automatic IR being central to this dissertation.

Although it seems a common-sense notion, it is important to distinguish between data retrieval and information retrieval. Data retrieval is concerned with looking for an exact match between queries and documents while information retrieval mostly seeks a partial match and then from this partial match, a small (manageable) number of the best documents are selected (best match) [van Rijsbergen, 79]. Data retrieval is best exemplified by a user's interaction with a DBMS. Here there is no ambiguity in the query, which will generally be expressed using some artificial query language such as SQL, and each query will only generate one possible (complete and exact-matching) set of results based on the underlying data. Ranking of this set of results is not possible (unless defined using 'sort by' or 'order by' commands within the query) as all results are equally valid. For example, the following query to a DBMS:

```
SELECT author FROM books WHERE title = 'Information Retrieval'
```

can only have one possible answer, which is a listing of all the authors who have written books called 'Information Retrieval'. We will briefly discuss DBMSs later in this chapter.

Information retrieval, on the other hand, is best exemplified by a user's search engine query. Table 1.1 shows ten randomly chosen queries, extracted from a query log comprised of queries that were submitted to the Excite[1] Search Engine [EXCITE, 02] in 1999.

horoscope	Who is Jimi Hendrix
seven kingdoms hints	Beard
United States Post Office	Electrical engineering
time zone map us	gift wrap wrapping paper
Personals	Shampoo making

Table 1.1 : Information Retrieval example queries.

[1] Excite has since ceased to provide search facilities over it's own index. Rather it is now a metasearch engine.

Automatic information retrieval removes the human from the relevance ranking process, leaving the human only to compose a query and examine the results automatically produced by the retrieval system.

A further distinction can be made with respect to information retrieval, which, although it does not apply to this thesis directly, is still important to make. This distinction is between information retrieval and document retrieval. In the strictest sense, information retrieval could refer to the retrieval of minimal information (such as Question Answering systems), which satisfies a user's query. In most information retrieval systems, however, the unit of retrieval is the document (in a ranked list) and not some smaller unit of information, resulting in the need for a user to scan a document in order to locate the required information. Techniques such as highlighting query terms in the result document to aid the user in locating the relevant information are commonplace, but the underlying unit of retrieval is still the document. It is our belief that a true IR system will attempt to provide the user with precisely the information (answers) that the user is requesting. A good example of such a system is the IONAUT Question Answering system [IONAUT, 02] which attempts to answer a user's query with specific content, extracted automatically from a collection of web documents. An example of IONAUT is shown in Figure 1.1. The query presented to the systems was "*Who were the members of the group Queen?*" and the top ranked (four) names returned completely answered the query by identifying the four members of the rock group.

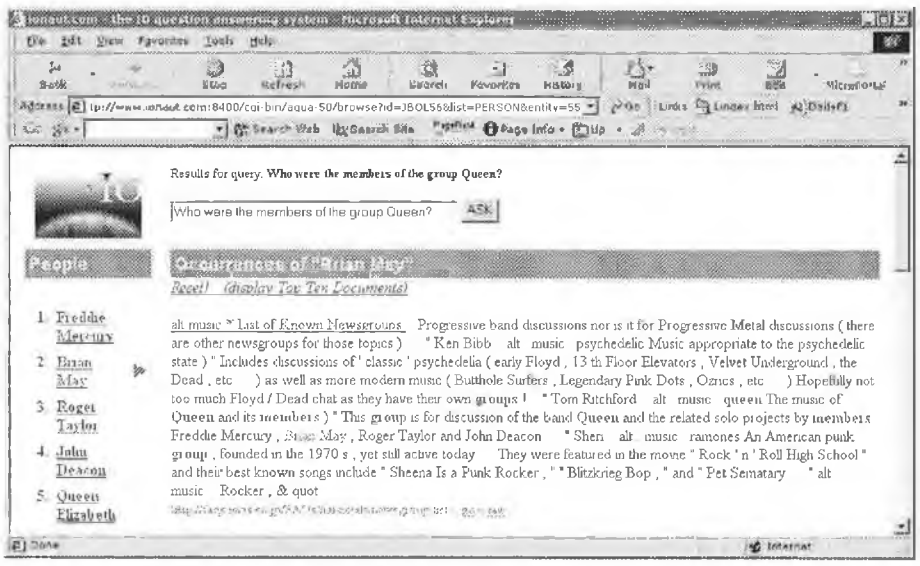


Figure 1.1 : IONAUT, a question answering system

It is clearly visible that IONAUT has extracted the required information from documents and presented just this information to the user, thus removing the requirement for the user to follow up on a query with a short phase of document browsing to locate the desired information. Based on this observation, the Figure 1.2 outlines a simple breakdown of retrieval techniques.

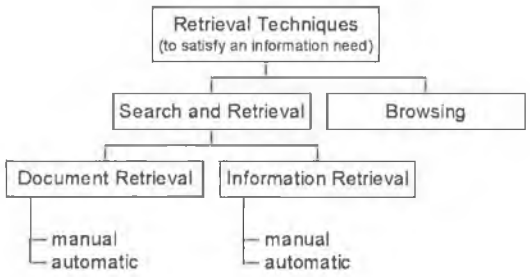


Figure 1.2 : A basic breakdown of retrieval techniques

It would be possible to include Question Answering systems as a third form of Search and Retrieval in Figure 1.2 as it is a large field if research in its own right, however this was not the option we have taken in describing Information Retrieval.

The term information retrieval (or IR as it will often be written) as used in this dissertation will primarily refer to the retrieval of unstructured data (in the form of relevant documents) of a textual nature. Of course, IR may refer to the retrieval of other forms of information such as image, video or audio, but these are not the focus of this dissertation. The textual data that we refer to is comprised of units, which are commonly referred to as documents. These documents will be grouped together into a collection, or a corpus of documents, which is referred to as a dataset. This collection may be static, as is the case with snapshot datasets such as those used by TREC, or dynamic as would be the case with a current collection of news articles or the WWW in general.

Before the 90s, IR systems were used to provide retrieval services over mostly static datasets and most of the experiment supporting datasets were small scale and static in nature. Since the emergence of the Internet and particularly the WWW, people are demanding retrieval facilities over huge numbers of WWW documents. IR systems that provide these services on the WWW are known as search engines and the largest is currently Google [GOOGLE, 02], which indexes in the region of 2 billion (web page) documents. Due to the nature of the WWW, these documents are in a constant state of change and there are new documents being added constantly.

1.3 COMPONENTS OF IR SYSTEMS

Figure 1.3 illustrates, in a very simplistic manner, the overall construction of a typical IR system. As can be seen, the system consists of three components: input, processor and output.

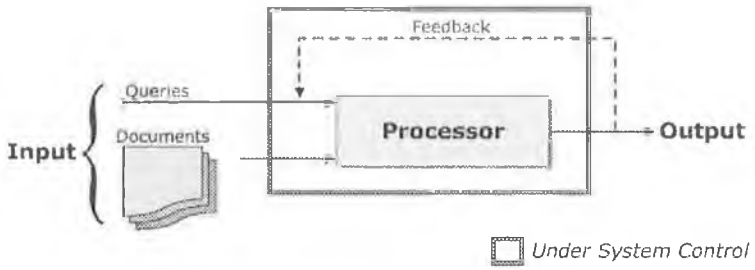


Figure 1.3 : Structure of a typical IR System, based on [van Rijsbergen, 79]

Looking at the inputs into an IR system, the primary task is to convert each input (both queries and documents) into an internal representation for a computer to use. The vast majority of IR systems will only store a representation of their inputs, as opposed to the full documents and queries. This is a one-way process, in that it is not possible to convert the internal representation of a document back into the original document. This internal representation of each document will (in most cases) take the form of a document vector[2] of significant words. The method of selecting and (perhaps) weighting these significant words will be discussed later in this chapter. A second phase of input could allow a user to modify the articulation of the information need in light of previous output of the system. This process is called '(relevance) feedback' and may be done both automatically (without the user even knowing that the feedback process is taking place) and manually.

The next component of a typical IR system is the processor. The processor is concerned with generating the memory and data structures in such a way as to achieve speedy, efficient and effective results in response to a user's query input. The processor will accept the internal representation of the query and calculate the documents that best match the user's information need as articulated by the query (input).

Finally we examine the output component, which is primarily composed of a set of documents which are returned (from the processor) in fulfillment of a user's information requirement.

---

[2] Document Vector : a basic description of a document vector is that it consists of a list of unique words extracted from a document that are considered to be significant and useful for the IR process.

This output may consist of a set of unranked document identifiers or the identifiers may be ranked in decreasing order of relevance.

We are now in a position to outline the steps an IR system must carry out in order to operate effectively.

1.3.1 Steps in Performing Information Retrieval

We can identify four distinct steps that a typical IR system must follow in order to be able to fulfill its task [Agosti, 00]. These are:

1.3.1.1 Document Gathering

This is the process of gathering the documents that are to form the core content of the IR system. If working with a fixed and readily available set of documents, then this is simply a process of knowing the location of each file on disk and gathering them before converting them into a searchable internal representation (*document indexing*), but it may be necessary to actively seek out content for the indexing stage, as is the case with search engines. In this stage also, some parsing of unnecessary content may take place. For example:

- Unnecessary mark-up of the text may be removed.
- Many frequently occurring words that are of no benefit to the automatic retrieval process may be removed. These words are called stopwords and we will discuss stopwords in greater detail later in this chapter.
- Terms within documents may be truncated to term stems (stemming).

1.3.1.2 Document Indexing

The documents gathered in the document gathering phase are converted into a fast searchable internal representation This will usually be implemented using some programming language dependent data structures which provide fast searching facilities such as arraylists, vectors, sets, multi-sets, maps or multi-maps.

1.3.1.3 Searching

This process involves accepting a query, processing it, finding possibly relevant documents, calculating a the degree of similarity between each document and the query for each possibly relevant document, sorting the set of relevant documents and returning these to the user in groups (usually) of

10. All this has to be done as efficiently and quickly as possible. For example, the IR system that operates as the Google search engine accepts and processes (as of July 02) [GOOGLE, 02] :

- 150 million queries per day.
- 6.25 million per hour.
- 105,000 per minute.
- 1,700 per second.

1.3.1.4 Document Management

In the previous three steps we have gathered documents, indexed them and are now allowing users to search their content. However, in many scenarios such as web searching, the documents that have been indexed will be unstable and constantly changing. Consequently, we must validate that:

- The documents that comprise the internal representation of the document collection are as up-to-date as possible.
- The documents included in the internal representation are actually still in existence.

This will involve re-gathering documents, at periodic intervals, or even completely rebuilding the internal representation of the document collection, which necessitates returning back to the document gathering phase and starting again.

1.4 Approaches to Automatic IR

In order for an automatic IR system to operate effectively, documents must be stored in some efficient internal representation within a computer system. In general, the internal representation of documents will differ from the original form of the documents, although in the early days of IR this was not the case. The reason for the different internal representation is that it would be too inefficient and slow to expect a retrieval system to engage in a process of full-text-scanning of all the documents in its index each time a query is processed.

Consequently, we will pre-process each document, often removing formatting and unnecessary terms that do not aid the indexing and retrieval process. However, the question arises,

how do we know what terms are unnecessary?. The solution lies in the work of Luhn, upon which much of the work on automatic text analysis has been based. Luhn [Luhn, 58] states that "It is proposed that the frequency of word occurrence in an article furnished a useful measurement of word significance". His assumption is that we can use term frequency information to extract words (and sentences) to represent a document. His addition of an upper bound and lower bound on the frequency of acceptable terms allows us to extract only the significant terms from a document to be included in an index. Letting *f* be the frequency of occurrence of terms in a document and *r* be their rank order (the order of their frequency of occurrence) and plotting this on the graph in Figure 1.4 we get the following hyperbolic curve.

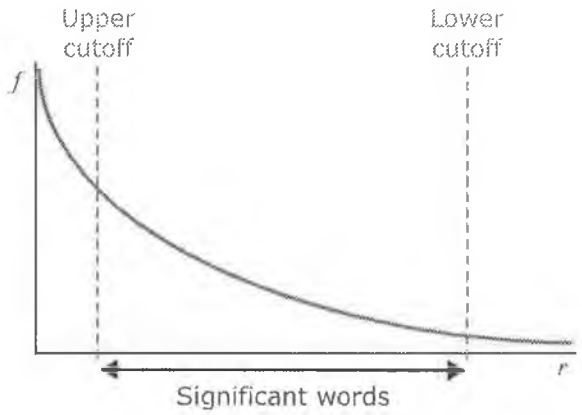


Figure 1.4 : Hyperbolic curve relating term occurrence frequency with rank order

This curve demonstrates Zipf's Law [Zipf, 32], [Miller, 96], which states that:

$$frequency(f) \times rank(r) = constant \qquad (1.1)$$

Based on Luhn's findings we can make two observations, that:

- Terms below the lower bound are considered too rare to be of benefit to the retrieval process.
- Terms above the upper bound are considered to occur too frequently to be of benefit.

This second observation leads to the process of stopword[3] removal, which, when converting a document from it's input format into the internal representation required by the processor, automatically removes high frequency words from the document and thus these words are never represented in the document vector. Table 1.2 shows a portion of such a stopword list, and demonstrates the kinds of words that are involved. The advantages of this approach are twofold; firstly the non-significant words are removed and will thus not interfere with retrieval, and secondly, since stopwords are the most frequently occurring words, the computer memory requirement for the internal representation for each document will be reduced by 30-50 percent for conventional texts.

a	across	against	alone
about	after	all	along
above	again	almost	also

Table 1.2 : Example of high-frequency stopwords.

Therefore examining the occurrence frequency characteristics of terms in text allows us to state that:

- The most frequent words are function words.
- The least frequent words are obscure.
- The mid-range frequency words are the content-bearing ones.

Therefore we should index text by the words with mid-range frequencies throughout the entire document collection (dataset).

Some additional processing may be done on the document terms prior to indexing, such as suffix removal (stemming). It should be noted that the process through which a document is converted into its internal representation is also applied to each query that is executed using the retrieval system. This is necessary in order to achieve proper matches between the internal representations of the queries and the documents.

[3] Stopwords : high frequency words which occur so often that they are of no benefit to the retrieval process.

1.5 INDEX TERM WEIGHTING TECHNIQUES

The method that we employ to index the internal document representations can be seen as consisting of a number of alternative approaches. Broadly these can be divided into the Classical models, the Structured models and the Browsing models. Figure 1.5 illustrates a taxonomy of IR models, with the highlighted models being of interest to us.

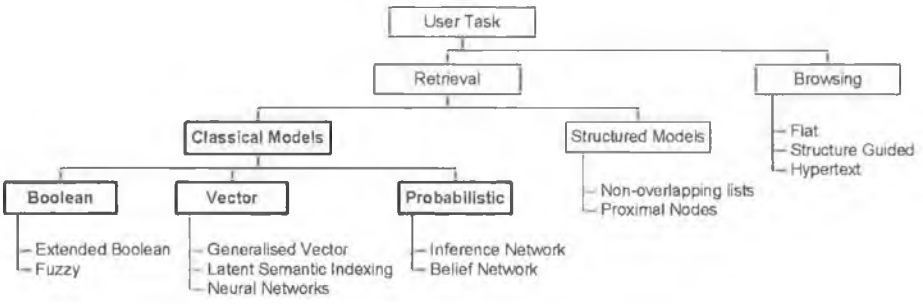


Figure 1.5 : A taxonomy of IR models [Baeza-Yates & Ribeiro-Neto, 99]

It is notable that the user task is divided into being either one of retrieval or browsing. This is especially the case on the WWW where a user often uses a retrieval system as a means of identifying relevant documents and then begins a browsing task to locate the required information. In effect, both tasks are being used interchangeably and in harmony with each other on the WWW.

In this dissertation, we are interested in the classical models of IR, which are the Boolean, Vector Space and Probabilistic models. In the boolean model documents are represented as sets of index terms and thus the model can be said to be set-theoretic. In the vector space model documents and queries are represented as vectors in a t-dimensional space (t being the number of unique indexed terms in the collection) and thus the model can be described as being algebraic in nature. Finally, in the probabilistic model the framework for the modeling of both documents and queries is based on probability theory and thus is described as being probabilistic in nature.

Recall that each document is processed resulting in a transformation from its original form into some internal representation for use by the retrieval engine. The structure of this internal representation will differ somewhat depending on the requirements of the model of IR that the system implements. We will now discuss the three classical models of IR.

1.5.1 THE BOOLEAN MODEL OF IR

The Boolean model is a simple retrieval model based on set theory and Boolean algebra and involves the query being formulated as a Boolean combination of terms. This allows for the use of the classical Boolean operators AND, NOT and OR. For example a query could be formulated thus : "java AND programming NOT coffee". In this case the documents that satisfy the query will contain the word java and programming, while any document that contains the word coffee will not be retrieved. From a developers perspective one could visualise the approach by separating the query terms into *n* atomic units, executing searches based on each individual unit and returning *n* sets of documents as one result set. At this point, the Boolean operator constraints within the query could be applied to produce the final set of relevant documents. For example, in Figure 1.6, the result for a query (A AND B) AND NOT C is shown:

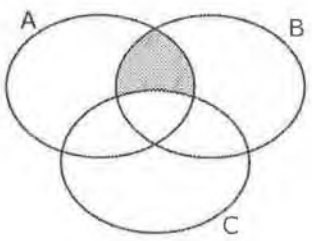


Figure 1.6 : The Boolean model of IR

One big disadvantage of the Boolean model is that its retrieval strategy is based on binary decision-making, that is, a document is either relevant or not relevant and this does not adequately support ranked output, leading to retrieval performance degradation. Thus, the Boolean model has many of the characteristics of a data retrieval model as opposed to an IR model. In addition the Boolean model does not make use of term weights, each term is given a binary weighting within a document (present or absent) [Cooper, 88]. Term weighting allows an importance value to be applied to each term within a document reflecting its importance within that document (and within the collection as a whole) which can bring with it a substantial improvement in retrieval performance.

1.5.2 THE VECTOR SPACE MODEL

The vector model[4] [Salton et al., 75], which was first introduced by Salton in the late 60s, assigns non-binary weighting to each term in both queries and documents. These non-binary term

[4] Vector model : often referred to as the Vector Space model.

weights are ultimately used to calculate the degree of similarity between a user's query and the documents indexed by the processor, thus producing ranked output of results. The goal of ranked output is that the most-relevant documents will be ranked highest and hopefully satisfy a user's information requirement without necessitating a user browsing through a large set of relevant documents in an attempt to find relevant content.

In the vector space model, a document d_j and a query q are represented as t-dimensional vectors (t being the number of terms in the index and the consequent dimensionality of the vector space) as shown in Figure 1.7. The degree of similarity of the document d_j with regard to the query q is calculated as the correlation (cosine of the angle θ) between the two vectors. It should be obvious that d_j is more relevant to q than d_i due to the relative size of the angles between the vectors.

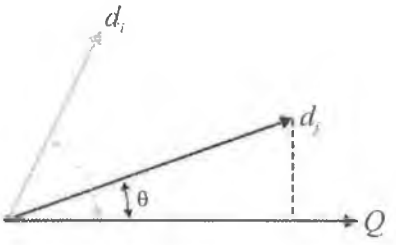


Figure 1.7 : The vector model; documents and queries in a vector space

So, instead of attempting to predict if a document is relevant or not, the vector space model simply ranks the documents according to their degree of similarity to the query. Time constraints in a real-world implementation may not support this for large document collections so it is probable that a subset of possibly relevant documents would be chosen to rank and this subset could consist of the documents that contain at least one of, or possibly all of, the query terms.

We have not yet specified how index terms are weighted, but there are many approaches that can be successfully applied. The most common approach views the retrieval problem as one of clustering (relevant and not relevant) and integrates two factors into the weighting process; the *tf* factor and the *df* factor.

The *tf* (term frequency) factor used to measure the raw frequency of a term inside a document, is simply a count of the number of occurrences of that term in the document and provides one measure of how well that term describes the document's contents.

The *df* (document frequency) factor is used to calculate how rarely a term occurs across a document collection and it is the inverse of the *df* factor (called *idf)* that is used in the calculation. The motivation for incorporating an *idf* measure is that terms that occur in many documents will not be very useful for distinguishing between relevant and non-relevant documents and it is beneficial for retrieval performance to take account of this evidence in the ranking process.

Perhaps the best known example (of many) of a term weighting scheme based on the vector model is called tf-idf and is calculated using the following formula where:

w_{ij} represents the weight assigned to a term T_j in a document D_i.
tf_{ij} = frequency of term T_j in document D_i
N = number of documents in collection.
df_j = number of documents where term T_j occurs at least once.

$$w_{ij} = tf_{ij} \cdot log\left(\frac{N}{df_j}\right) \qquad (1.2)$$

However, this approach is rather simplistic and does not take into account the length of the document. If we compare the *tf* scores of terms in long and in short documents, they will differ and rank longer documents above shorter documents, due to higher *tf* values in the longer documents. Therefore, the *tf* values should be normalized with respect to the length of the documents. There are many ways of doing this from simply using *1+ln(tf)* as in formula (1.3) below, or normalizing *tf* based on the max *tf* in the document (1.4), or more complex techniques such as Singhal's pivoted document length normalization [Singhal et al., 96].

$$w_{ij} = 1 + ln(tf_{ij}) \cdot log\left(\frac{N}{df_j}\right) \qquad (1.3)$$

$$w_{ij} = \left(\frac{tf_{ij}}{max_i\ tf_{ij}}\right) \cdot log\left(\frac{N}{df_j}\right) \qquad (1.4)$$

Despite its simplicity, the vector space model improves retrieval performance over Boolean IR by providing ranked output, sorted by the degree of similarity of document to query, which is difficult to improve on without some form of relevance feedback or query expansion such as Rocchio's approach which we will describe later in this chapter. Versions of the vector space model are used in the majority of Search Engines on the WWW today, as the tf-idf is well suited to retrieval of WWW documents although recent problems of keyword spamming have highlighted some shortcomings.

1.5.3 The Probabilistic Model

The Probabilistic model[Robertson & Sparck Jones, 76], [Robertson, 77], introduced in the 70's, attempts to capture the problem of content retrieval within a probabilistic framework. Given a query q, the probabilistic model assigns to each document d_j a measure of its similarity to the query based on the ratio *P(d_j relevant to q) / P (d_j not relevant to q)* which computes the odds (probability) of document d_j being relevant to q. Many formulae can be used to implement the probabilistic model, but one of the most commonly used formulae is known as BM25[5] [Walker et al., 97] and it uses different formulae to index both documents and queries.

To index documents:

W(ij) assigned to a term in a document is given by :

$$W_{ij} = \frac{(k_1 + 1)\, tf_{ij}}{K + tf_{ij}} \qquad where \quad K = k_1 \left[(1 - b) + b \bullet \frac{l_i}{avdl} \right] \tag{1.5}$$

With tf_{ij} indicating the within-document frequency of term j in document i and b, k_1 are parameters. K represents the ratio between the length of document i measured by l_i (sum of tf_{ij}) and the collection mean, denoted by *avdl*.

To index a query:

W(ij) assigned to a query term is given by :

$$w_{qj} = \frac{tf_{qk}}{k_3 + tf_{qk}} \quad . \quad ln\left[(N - df_j) / df_j \right] \tag{1.6}$$

[5] In BM25, BM stands for Best Match.

where tf_{qi} indicates search term frequency, df_i indicates collection-wide term frequency, N is the number of documents in the collection and k_3 is another parameter. Best result parameters have been determined by experimentation for different collection sizes and statistical distributions.

1.6 Other Issues in Information Retrieval

There are a large number of other issues in IR that we have not looked at, but two of the most important are the concepts of relevance feedback and query expansion.

1.6.1 Relevance Feedback

Relevance feedback is the concept of feeding back into a system, some relevance judgments from previous results that the system can then use to reformulate the search in an attempt to improve retrieval performance. These relevance judgments will typically have been made by the user who will have received the top ten ranked documents in response to a search. The user is encouraged to identify to the system which documents are relevant to the information need so that these documents may be used in the process of relevance feedback.

In its simplest form, relevance feedback can be used to re-compute the weights of query terms based on their frequencies of occurrence in relevant documents. In more complex forms, we can select the most representative terms from the documents that the user has identified as being relevant and add them to the query to produce a new query and using the new query run the search again and produce new documents for the user, and so on until the user is happy. A form of *tf-idf* is often used to weight the query terms.

Typically, during a search, a user may find no more than a handful of relevant documents at all, so using this scant information on say, 3 or 5 or 10 known relevant documents could be termed statistically unreliable, but implementations show it does improve effectiveness overall. Relevance feedback is tied in closely with the concept of query expansion.

1.6.2 Query Expansion

Query expansion is the name given to the process of expanding a query to incorporate more terms, either manually or automatically. As a search proceeds, users' information needs shift or are refined or evolve, depending on the task, but they do change. Allowing the user to expand a query during a search session helps reflect this. As relevant documents are discovered, these can be used as a source of new query terms to add to the original query with the overall goal of improving retrieval performance.

Probably the most famous method of automatic query expansion was developed by J.J. Rocchio Jnr, back in the mid 60s [Rocchio, 65]. It is simply known as Rocchio and it works like this:

1. Process a user's query in the normal manner and return the top twenty documents.
2. Convert the original query into a weighted query.
3. Add the top twenty weighted terms from the top twenty documents (calculated using a formula similar to that below) including their weights to the query giving us a new expanded, weighted query.

The weights for the document terms (Wt'_{ij}) are calculated using a formula similar to [Singhal et al, 98], where 0.2, 0.8 are parameters and *dl*, *adl* refer to the document length and average document length (in bytes) respectively:

$$Wt'_{ij} = tf - idf\ Wt_{ij} \times \left(\frac{1}{0.8 + 0.2 \times \frac{dl_i}{adl}} \right) \qquad (1.7)$$

This new query is passed back to the retrieval processor and the new results passed back to the user in the normal manner as the result-set of ranked documents of the search.

1.7 THE ARCHITECTURE OF A SIMPLE INFORMATION RETRIEVAL SYSTEM

An issue that remains to be discussed in this chapter is the nature of this internal representation into which documents and queries are converted in order to produce a ranked list of relevant documents. Figure 1.8 outlines a basic overview of the architecture of a simple IR system, which operates using an inverted index structure.

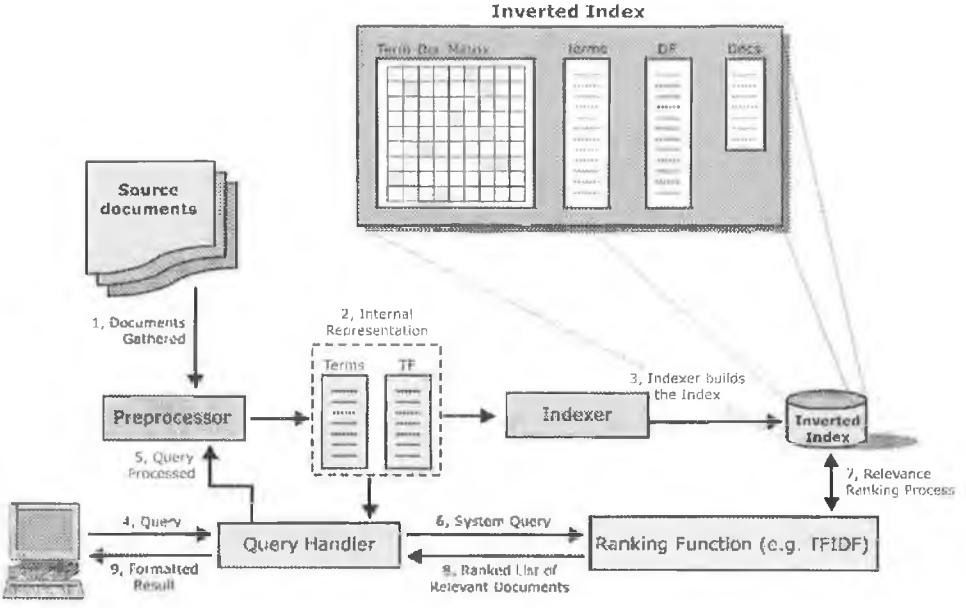


Figure 1.8 : The architecture of a basic IR System

1.7.1 THE PROCESS

Generally we can divide the IR process into two distinct phases with phase 1 being the building of the index and this must be carried out before the second phase, processing the query is supported. Steps 1-3 in Figure 1.8 comprise phase 1. Step 1 in the process is to input the document collection over which retrieval should operate into the Preprocessor (one document at a time). The Preprocessor may remove stopwords and stem the text to produce the internal representation of each document (step 2). This internal representation is (minimally) going to consist of a sorted list of unique terms and an indication of the document *tf* value for each term. Step 3 involves the Indexer taking the internal document representation and adding it into an inverted index, which is the internal representation of the whole document collection that supports fast search and retrieval. The structure of a typical inverted index is described presently.

Once all the documents have been added into the inverted index the system is ready to accept queries (phase 2). A Query Handler is the process that the user of a search engine will interact with. It is responsible for accepting queries and returning ranked formatted output to the user. Step 4 illustrates the system accepting a query and in step 5 the query is passed to the Preprocessor and is processed in the same way as a document which produces the system query which in step 6 is passed from the Query Handler to the Ranking Function. The ranking function (step 7) calculates the similarity of documents to the query (using the chosen ranking algorithm such as tf-idf or BM25) before the ranked list of documents is passed back to the Query Handler in step 8. The Query Handler then formats the ranked output (perhaps wrapping it in HTML) and passes it back to the user (step 9) thus completing the process.

1.7.2 The Inverted index

Inverted index structures are central to most IR systems and track which documents contain which index terms. An inverted index will have to map terms to documents and may store a weighting for each term in each document. In mapping terms to documents, a critical shortcut is provided which avoids searching the entire document database in response to a query. Instead of viewing the document as a pointer to a list of terms comprising the document, the inverted index views terms as the atomic unit which act as pointers to documents that contain the terms as in figure 1.9. A term may have associated pointers to a large number of documents if it's *df* value is high and obviously a document from the collection will be the target of many document pointers.

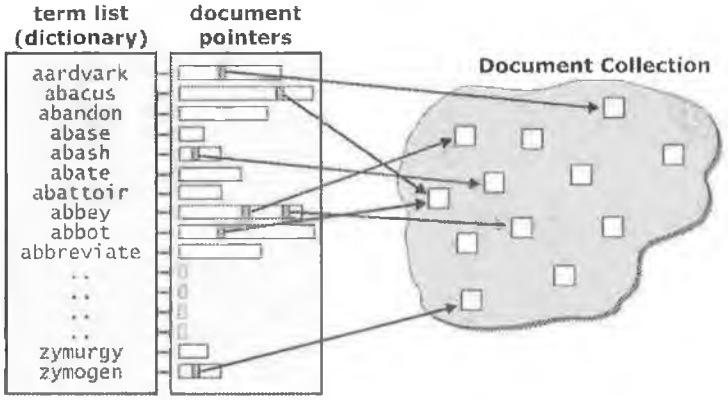


Figure 1.9 : Illustrating inversion as used in an inverted index

There are three components of an inverted index structure:

- The Document List, which is a listing of all the documents in the index, each having being given a unique identifier.
- The Term List or Dictionary, which is a sorted list of all the unique terms in the document collection. This list is sorted to support fast searching over its contents.
- The Term-Document Matrix, which allows for the encoding of what terms are contained in what documents. If the system is based around Boolean IR then no weight is stored at the term document intersection, however in all other approaches the term weight is stored and this is a number representing the importance of the term to the document and to the document collection as a whole. If the retrieval system is based upon a tf-idf ranking scheme, the inverted index will store a numerical representation of the weighting of each term in each document, implemented as a term-document matrix, but this poses one difficulty in that the matrix quickly becomes too large to be held in the memory of a computer. A close examination of this matrix shows it to be sparse in nature, meaning that there are a relatively small number of non-zero values in the matrix compared to the number of zero values. In order to avoid both the storage and processing of zero elements, a variety of sparse matrix formats have been developed, such as Compressed Row Storage and Compressed Column Storage [Berry & Browne, 99], which require three arrays as opposed to a matrix to store a term-document matrix. This reduces the storage requirements from $m \times n$ array locations to $2nnz + m + 1$ array locations.

While the inverted index is a common method for internal representation of a document collection, there do exist alternatives such as signature files. Signature files are based on dividing input text into logical blocks of fixed length and for each block, generating a signature by hashing words in the block and using Boolean logic to combine the hashed words together [Faloutsos & Christodoulakis, 87]. For retrieval, a signature is generated for the query and a sequential search is performed through signature file looking for signatures that subsume or exactly match the query signature. Comparison of inverted indices and signature files [Zobel et al., 98] has shown unequivocally that for typical indexing applications, signature file techniques do not perform well when compared to inverted indices. Signature files are larger, more expensive to build and update, require pre-indexing analysis of the collection and are slower during retrieval than inverted files.

1.8 Searching the Web.

Web search is without doubt the point of contact that most people have with IR systems. The manual discovery of information on the WWW is next to impossible due to the volume of documents in existence, currently in the (single figure) billions (just counting static documents). Although the hyperlink structure of the WWW supports browsing for information, research has shown that, on average, there are 19 degrees of separation between web pages [Albert et al., 99]. In addition, not all pages are accessible by following hyperlinks from any one given starting point and in any case, reading all web pages would require many lifetimes, thus making it next to impossible to find information by judiciously following links. Therefore, some content-based retrieval system is required to aid a user finding information on the WWW, and such services have been available since the early nineties.

The World Wide Web Worm (WWWW) [McBryan, 94] was one of the initial crawler-based search engines (the common name given to an IR system used to index and provide retrieval facilities over WWW documents) developed. Many more search engines have followed the early ones, some of which such as Lycos [LYCOS, 02] are still in existence today.

If we view the problem of searching for information on the WWW as analogous to searching for information between the covers of a large book we can identify the two types of

retrieval systems present on the WWW. Within a book (mostly non-fictional) the available search aids are the Table of Contents and the Index. An individual requiring a retrieval system on the WWW can choose one of two types of search aids, Web Directories or Search Engines. Each retrieval system can be broadly classified as belonging to either of these two types, although the dividing line between both types has all but disappeared in some cases.

1.8.1 WEB DIRECTORIES

Web Directories are comprised of a structured hierarchy of pages, each of which contains many links to other web pages based on the content of these pages. These (usually) have been painstakingly handcrafted by humans, which make them very expensive to maintain and grow in-line with the ever-expanding WWW. However, they do act as excellent starting points for a user to browse the web. If one views the web as a book then the web directory is like the table of contents, with a high-level overview of the contents of the WWW. If you are just browsing a non-fictional book, using the table of contents is a great way to quickly locate a desired section. It will get you near to what you are looking for, but you will have to do some additional reading to find the exact information required. Similarly, a web directory acts as a starting point for additional exploration, and any website within would be a good candidate to act as a starting point. This process of exploring the web is usually referred to as browsing for information.

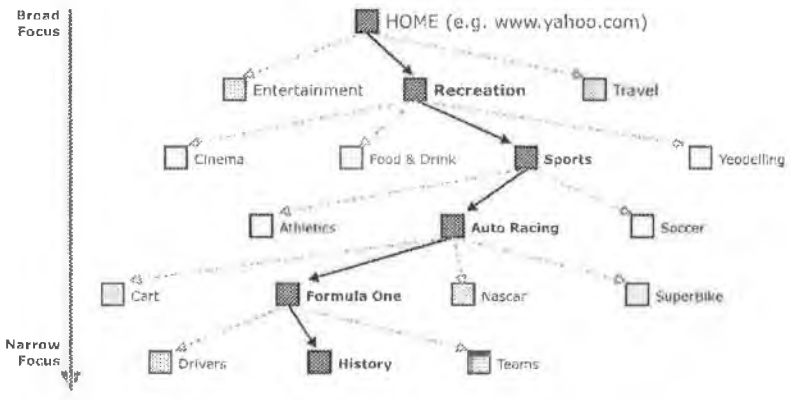


Figure 1.10 : The Hierarchical Structure of a Web Directory

The web directory will be laid out in a hierarchical fashion with numerous levels of menus that can be viewed as a tree structure. The farther down the directory tree one travels, the more

focused the information becomes. For example, in the Yahoo web directory, documents[6] regarding the history of Formula 1 motor racing are in this directory path:

Home > Recreation > Sports > Auto Racing > Formula One > History

Web directories, in the majority of cases, offer a text search facility over the directory. One can search all the pages contained in the directory (from HOME in Figure 1.10), or one can search a subset of the directory tree from any point downwards. For example, a search could be executed on all documents related to 'Formula 1'. This is very useful as one can search within a set of high quality pages, all of which are focused on particular clearly defined topics. The two best examples of web directories are Yahoo [YAHOO, 02] and the Open Directory [OPENDIRECTORY, 02] project (as shown in Figure 1.11).

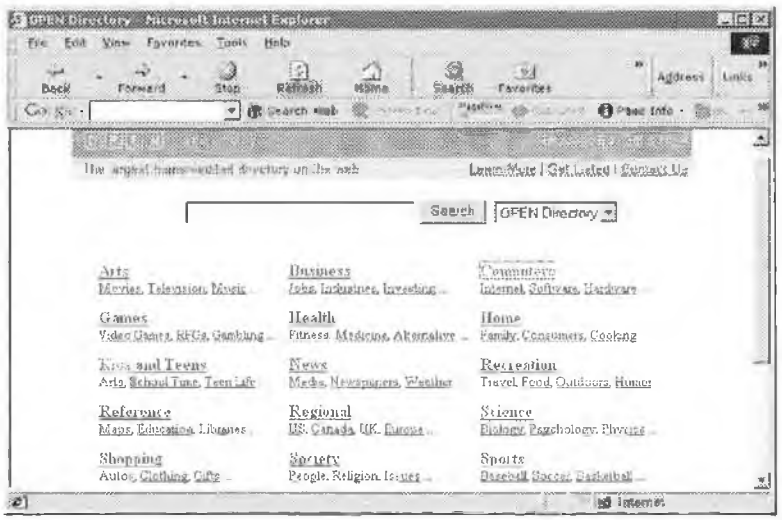


Figure 1.11 : The OpenDirectory Web Directory

In general, hierarchical approaches to search are considered to be the among the best guidance tools to minimize the risk of user disorientation [Botafogo et al., 92] when navigating a hypertext. The major problem with the directory approach is that, as mentioned, it requires huge levels of human intervention and consequently is limited in terms of size. There are, however, methods of automatically clustering documents into groups based on their content, probabilistic

[6] There are only two of these documents found, which perhaps indicates the problem of lack-of-coverage when relying on manually generated directories.

methods [Goldszmidt & Sahami, 98], clustering based on short segments of text [Zamir & Etzioni, 98] and WebCluster [Mechkour et al., 98] and also their linkage [Chakrabarti et al., 98], however this is not something that we have focused on in the research presented in this thesis.

1.8.2 Search Engines

The other broad type of retrieval system used on the WWW is the 'search engine'. Search engine is a generic term, which is used to describe the different types of complex software tools that together comprise a content-based IR system on the WWW. If we employ the book metaphor again, a search engine is similar to the index at the back of a book, which lists all the important words, and the pages upon which those words appear (a book constructed using modern publishing software can have these lists generated automatically). If one is looking for information from a web search engine we enter relevant terms and the search engine, generally using one of the term-weighting approaches mentioned earlier, will generate a ranked list of pages that contain these terms and (hopefully) will satisfy the information need.

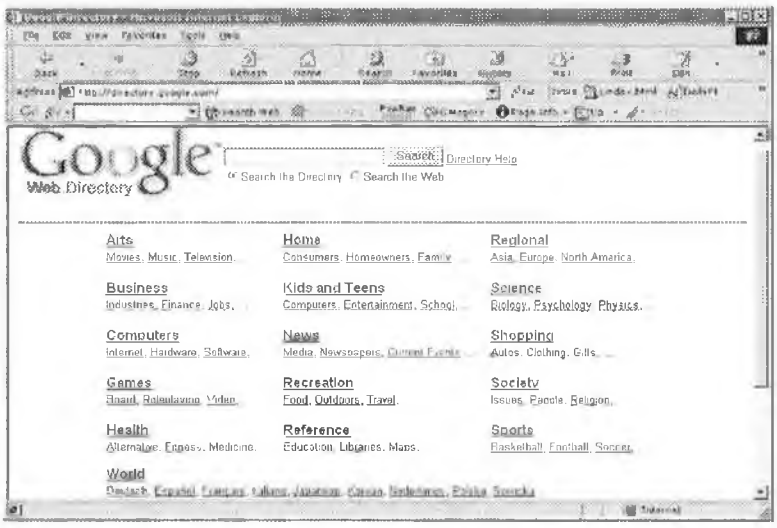


Figure 1.12 : The Google implementation of the Open Directory

However, this broad separation between the two search services is not actually the case in reality anymore. The majority, if not all, large web directories also provide a text search facility over

their directories while search engines are starting to provide access to directory listings also. Many of these are supplied from a source such as the open directory project [OPENDIRECTORY, 02]. If we examine Figure 1.12 we will see that Google's web directory service is just a version of the Open Directory web directory as shown in Figure 1.11. However, the point is important that there are two distinct search services, each of which poses it's own problems, regardless of how they may be combined in reality on the WWW. For this dissertation, we are primarily interested in search engines and will rarely refer to search directories. Examples of search engines in common use on the WWW of today are Google [GOOGLE, 02], AltaVista [ALTAVISTA 02] and AllTheWeb [ALLTHEWEB, 02]. The query screen of Google search engine is shown in Figure 1.13.

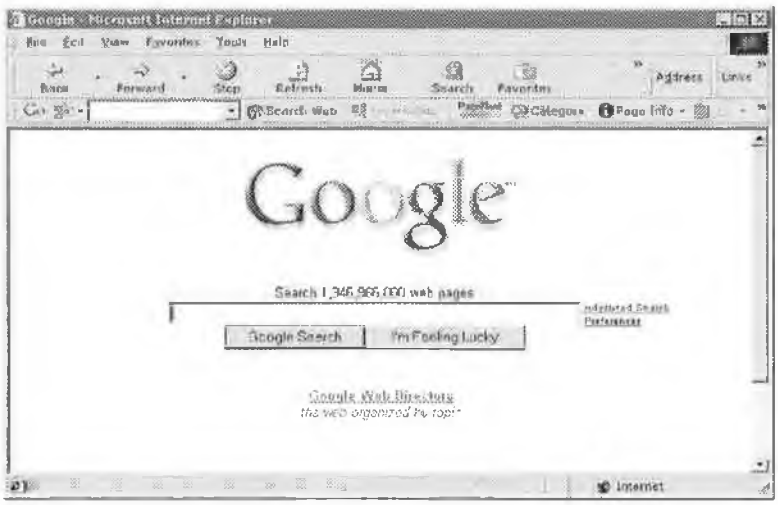


Figure 1.13 : The Google search engine

1.9 Database Management Systems (DBMSs)

There are two major categories of systems available to process information data: IR systems and Database Management Systems. We have already mentioned that data retrieval is concerned with looking for an exact match while IR mostly seeks a partial match for a request for information and this has been outlined in the previous sections. As mentioned previously, data retrieval is best exemplified by a user's interaction with a DBMS, so let us briefly examine the nature of DBMSs, not just with respect to data retrieval, but also because they do play a important role in supporting the experiments into web structure that underlie this dissertation.

DBMS's provide storage and retrieval services over structured data. Structured data is well defined and is typically represented by tables in the relational model [Codd, 70]. When querying data a language such as SQL is used to generate a query like the example SQL query on page 2. A DBMS table consists of zero or more rows, each of which consists of one or more columns, each of which can hold exactly one piece of data of a specific type, such as an integer, float or character array. Figure 1.14 shows a table from a database used in this research to store information about links between web documents.

idx	source	target	sourceHost	targetHost	type	totalLinks	docExists	docLinkCount
0	http://www.theage	http://ads.fairfax	www.theage.com.au	ads.fairfax.com.au	F	6	y	0
1	http://www.theage	http://www.theag	www.theage.com.au	www.theage.com.au	S	6	y	1
2	http://www.theage	http://www.theag	www.theage.com.au	www.theage.com.au	S	6	y	2
3	http://www.theage	http://www.theag	www.theage.com.au	www.theage.com.au	S	6	y	3
4	http://www.theage	http://melbourne.	www.theage.com.au	melbourne.citysearch.c	F	6	y	4
5	http://www.theage	http://ads.fairfax	www.theage.com.au	ads.fairfax.com.au	F	6	y	5
6	http://www.azalea	http://www.biblel	www.azaleacity.org	www.biblelessons.com	F	1	y	0

Figure 1.14 : Table showing 5 rows of linkage data.

Each row in this table consists of nine columns each of which stores information on some aspect of a web link. These columns are called "idx", "source", "target", "sourceHost", "targetHost", "type", "totalLinks", "docExists" and "docLinkCount" and are considered to be accurate descriptions of the data stores therein. Such a database that stores linkage information would allow access to this data by means of a structured query language such as SQL where the query to extract the urls of all the documents that link into a url such as www.apple.com would be:

```
SELECT sourceURL
FROM links_table
WHERE targetURL = 'http://www.apple.com'
```

In effect the query is precise and unambiguous with only one correct answer (from today's WWW) which happens to be of size 87,700 URLs[7].

[7] The source of this figure 87,700 is a Google query "link:www.apple.com" which returns the URLs that link into the query URL. The query was sent in July 2002.

1.10 Objectives of the Research being Undertaken

Today, users on the WWW demand high quality responses to their information requirements and demand this information promptly. This dissertation is centered on developing and evaluating improved methods for ranking documents on the WWW by exploiting the latent human judgments encoded in the hyperlink structure of the web.

Anecdotally the WWW IR research community believe that linkage-based retrieval of web documents is preferable over conventional content-only retrieval and that search engines such as Google that are known to incorporate linkage-based retrieval are believed to gain an increase in retrieval performance as a result. However, the research community, using accepted evaluation processes (TREC) have been unable to actually confirm or quantify the benefits of linkage-based retrieval. In this thesis, we present some of our own techniques for both generating linkage (or qualitative) weights for web documents and for combining linkage evidence with content evidence to produce one final ranking for documents in response to a user's information need. Finally, we examine the nature of the currently accepted evaluation process and identify where improvements should be made to support faithful evaluation of linkage-based retrieval techniques.

1.11 Summary

At this stage, the reader should have a clear understanding of what information retrieval (IR) is what the context is in which we are operating. As we have seen, data retrieval is concerned with finding an exact match between data and queries and is typified by a DBMS while information retrieval is more concerned with best match type retrieval where exact matches are rare occurrences. We have identified the four steps in performing information retrieval, namely:

- Document Gathering
- Document Indexing
- Searching
- Document Management

In addition to this, we have discussed various term weighting strategies based on the Boolean, Vector and Probabilistic models of IR along with conventional IR concepts such as relevance feedback and query expansion. Based on these term weighting strategies we have described the architecture of a basic retrieval system focusing on aspects such as the inverted index. Finally we categorized WWW based information retrieval systems into either of the following two categories:

- Web Directories such as Yahoo or the Open Directory, which are mostly human constructed retrieval systems.

- Search Engines, such as Google or Teoma [TEOMA, 02] which are automatic tools for providing content retrieval facilities for WWW data and whose architecture would be loosely based on the architecture of a basic retrieval system that we have discussed in this chapter.

Finally, we briefly discussed DMBS and their role in providing data retrieval before discussing the objectives of this thesis which are to develop new linkage-based retrieval techniques and identify where improvements should be made to the currently accepted evaluation methodology in order to support faithful evaluation of linkage-based retrieval techniques.

Chapter 2

INCORPORATING LINKAGE ANALYSIS INTO WEB SEARCH

In this chapter, we examine the nature of searching the WWW, identifying the inherent difficulties and benefits thereof before we develop the architecture of a sample WWW search engine. We then examine the topic of Linkage Analysis (one of the potential benefits of WWW IR over conventional IR) and discuss common techniques such as citation ranking, HVV, PageRank and Kleinberg's algorithm. Finally, we examine the architecture of a search engine that incorporates a linkage analysis component and discuss the requirements for a connectivity server to serve accurate and timely linkage information.

2.1 INFORMATION RETRIEVAL ON THE WEB

In the previous chapter, we have discussed IR as it applies to conventional document collections and illustrated the architecture of a simple IR system as well as introducing search engines. The first generation of full-text web search engines such as Lycos [LYCOS, 02] and WebCrawler [WEBCRAWLER, 02], have contributed to the huge popularity of the www. They were based on directly computing the similarity between a query and the text appearing in a web page and were, effectively, a direct application of the standard document retrieval techniques outlined in the previous chapter such as tf-idf or BM25. These web search engines process and index web documents that differ from other text in that they are encoded using HTML (HyperText Markup Language [W3C, 02], [Powell, 98]).

2.1.1 HYPERTEXT MARKUP LANGUAGE

Hypertext refers to an approach to information management that supports the utilisation of links within information to point to other pieces of information where this pointing or linking is based on content or meaning. These pieces of information are referred to as nodes and these nodes may be documents, groups of documents or sections within a document. Hypertext systems support non-sequential browsing of information as opposed to traditional text, such as a book, which supports linear reading. The browsing facility of a hypertext system is provided by hyperlinks or (simply) links which allow a user to jump from node to node, node to nodes or within nodes,

normally by means of a point and click interface. The history of hypertext stretches back to Vannevar Bush's seminal article [Bush, 45] in which he describes an extensible hypertext system called MEMEX, which, although never implemented is regarded as the origin of hypertext.

HTML, the standard for content encoding on the WWW is most people's only interaction with hypertext systems, although strictly speaking HTML can only be regarded as an implementation of a subset of hypertext principles. The majority of text-searchable WWW documents (web pages) are written in HTML, although many other formats of text-containing documents are becoming prevalent on today's web. Some of these include:

- XML documents
- PDF & PostScript documents
- Other document formats such as Word, PowerPoint & StarOffice documents.

HTML is nothing more than a text based markup language that consists of conventional content-bearing text augmented with tags and attributes that support the non-linear browsing of information. Tags are enclosed in angle brackets and will contain zero or more attributes. Many tags have an associated closing tag, prefixed with a forward slash (/), that results in all text between the opening and closing tags being affected. For example, this piece of HTML:

```
<FONT FACE="Verdana" color="Blue">Information Retrieval</FONT>
```

contains the tag "FONT" which indicates that the text 'Information Retrieval' should appear in the font specified by the attributes, i.e., in the font face 'Verdana' and in the colour 'blue'. HTML allows us to describe the layout of a document, incorporate non-textual elements such as images, audio and video and also provides the facility to link into other documents, or subsections of documents. However, HTML only attempts to describe the layout of the data on screen as opposed to describing the actual content of the documents themselves. The hypertext viewing application, which provides browsing facilities over HTML, is commonly referred to as a Web Browser, see Figure 2.1.

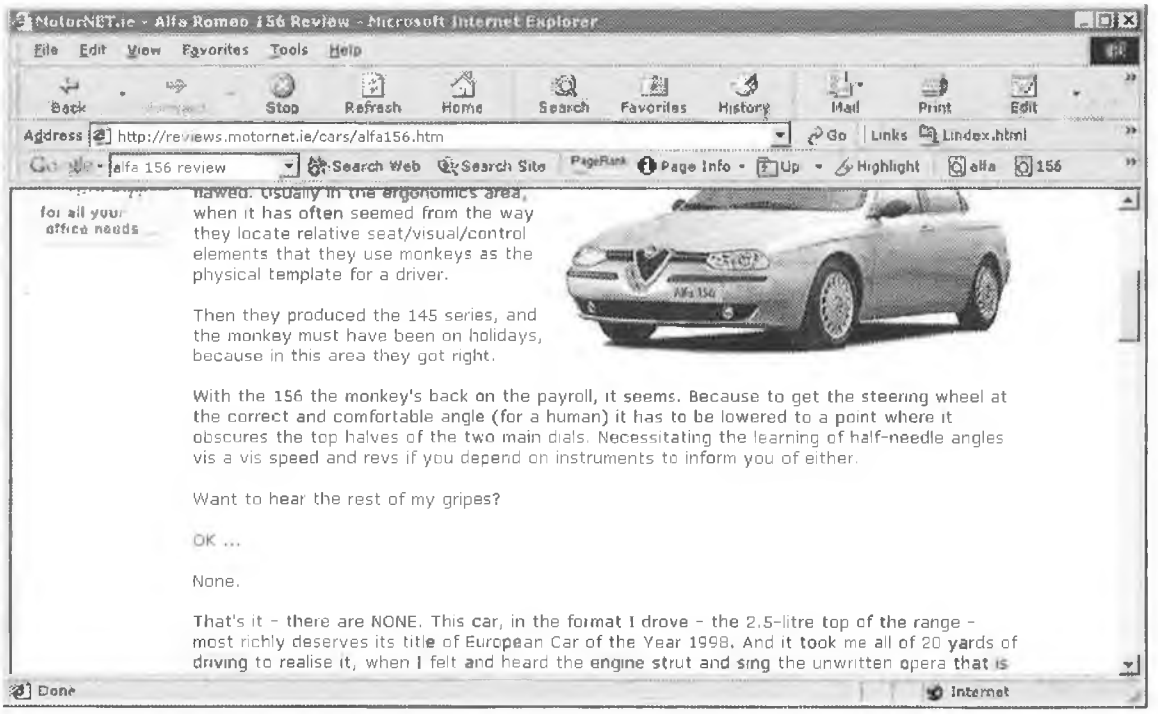


Figure 2.1 : Microsoft Internet Explorer as an example of a Web Browser

2.1.2 THE CHALLENGES OF WWW SEARCH

Automatic indexing and retrieval of a large number of HTML documents, as would be necessary for a search engine, poses a number of challenges (some of which are outlined in Modern Information Retrieval [Baeza-Yates & Ribeiro-Neto, 99]):

- **Unstructured Documents** – HTML is a very unstructured method of creating documents. It must be remembered that HTML has been developed as a technique for describing how a document should appear on the screen as opposed to describing the content of a document. In addition, any HTML document may be incorrect or invalid. This is due to the fact that HTML is an easy to understand mark-up language and many individuals choose to craft HTML documents using a text editor, which often results in the individual producing invalid HTML, instead of using one of the more accurate HTML editor tools such as Microsoft FrontPage [FRONTPAGE] or HTML HotDog from Sausage Software [HOTDOG] which will produce syntactically correct HTML. Web browsers themselves do not enforce the accurate encoding of HTML text and go to great lengths to parse and display erroneous HTML. Consequently, any system that deals with web information retrieval must be robust with well developed error handling abilities.

- **Heterogeneous Documents** – The WWW contains for the most part HTML documents, but these exist alongside:

    - Images: .gif, .jpg .bmp and .png are the most widely found images.
    - Audio Files: .mp3, .wmp, .acc, .wav, .midi and .rm/.rmj.
    - Video & Animation Files: .mpg, .avi, .qt and .gif.
    - Non-HTML Documents: text files, Word Docs, PowerPoint files, PDF and PostScript files.

  A standard search engine will only be able to index HTML files and text files. Some of the larger search engines offer search facilities over PDF documents as well as many standard office format documents. We will show later in this chapter, when discussing Hyperlink Vector Voting, how it is possible to apply standard text IR approaches to provide some limited search facility over the all types of documents mentioned above, without having to examine their contents, by using a form of linkage analysis.

- **Remote Documents** – Unlike pre-web document collections which may exist in one location or on one server, a web search engine must locate documents for its index. These documents are spread over almost 190 million remote servers [NetSizer, 02] and locating these documents requires the development of a crawler in addition to a retrieval engine. A web crawler, or robot, or spider, is a software application that must traverse the web by following the hyperlink structure of the web, gathering documents for inclusion in the retrieval engine's index as it travels. These remote servers may not be operational when the crawler visits, files may be missing or invalid, the HTML may be badly written making it difficult to parse links and in addition to all that, the robot's exclusion standard (see Chapter 5) should be adhered to. We will discuss these problems in detail in Chapter 5 when we discuss the architecture of a web crawler that we developed to support the research presented in this thesis.

- **Volume of Documents** – The WWW is growing on a daily basis and the latest estimates would put the web at well over 6 billion documents [SEARCHDAY291, 02]. In Figure 2.2 we can see the number of pages indexed (but not necessarily downloaded) by five of the major search engines, data coming from Search Engine Watch [Sullivan, 01], Google, AllTheWeb as well as Pandia Search World [PANDIA, 02]. It is worth noting that Google's

score is somewhat inaccurate because it is overestimated, which we will explain later in this chapter.

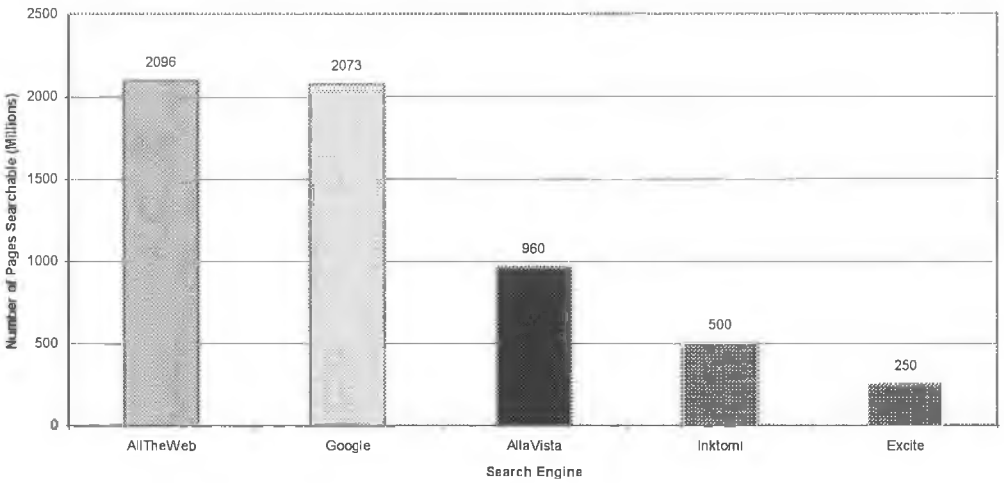


Figure 2.2 : Number of Searchable Pages for five large search engines

However big the web is and however we feel that this ever-increasing data size is impossible to deal with, it only amounts to about 25-30 terabytes [SEARCHDAY11, 01]. CERN's newest particle accelerator, the Large Hadron Collider, is expected to generate 100 petabytes (100,000 terabytes) of data in just a single experiment. This size of dataset (between 3-4,000 times the size of the web) is wholly beyond the capabilities of even the best-designed search engines in existence today. In order to solve this problem a team of researchers from Johns Hopkins University, Microsoft Corp., the California Institute of Technology, Fermilab and CERN are working together to build a massively distributed database that will be accessible via the web. The technological groundwork for future generations of large-scale web search engines is already being researched.

- **Invisible Documents** - Although hyperlinks abound on the web, many pages are not linked to each other via intermediary documents at all. A crawler may not know of the existence of many tens of millions of documents because it can only find pages from a given starting set by traversing the hyperlink structure of the WWW. This unreachable section is 8.24% of an Altavista crawl of over 200 million pages in size and contains 1.5 billion links [Broder et al., 00]. Scaling up to a (probably underestimated) 6 Billion document WWW this would suggest

over 450 million undiscovered pages. Unless web pages are submitted to the crawler from sites within these sections, they will remain inaccessible and invisible to the vast majority of web users. In addition, there are a huge number of pages that are dynamically generated from databases and thus not available to be indexed by most of the current generation of search engines.

- **Dynamic Documents** – Many web documents are not static and are constantly changing and this change must be updated in the search engine's inverted index, yet most pre-web IR systems were designed for static content. In order to keep its index up-to-date a search engine must constantly verify that all documents in its index have not changed and if documents have changed, they must be downloaded and the index updated. In addition new documents must be found and downloaded and documents that no longer exist must be removed from the index. However difficult this task is, the 'freshness' (degree to which the index reflects the web today) of a search engine's index is always open to scrutiny. Google's document caching facility (see Figure 2.3) helps to overcome this problem by maintaining a cached version of the document it downloaded, so if the document no longer exists, a snapshot exists of the document when it was downloaded which can be presented to the user.

Figure 2.3 : The Google Cache Facility (highlighted)

- **Useless content** – Many of the web documents in existence can be considered to contain useless information, while only a (useful) fraction is focused on useful and structured information. The important (including many company sites, literature, technical information) co-exist side by side with the ephemeral such as yesterday's lunch menu. In addition, many pages contain duplicate information, for example on a mirror web site. It is estimated that 30

% of the web is duplicated content [Huang, 00]. This content must either be removed at indexing-time, or parsed out of the results at query-time.

- **Languages** – We know that there are documents on the web written in over 100 languages as of January 2000 [Huang, 00]. Many minority languages such as Welsh or Breton have less than 10 million words of web text [Grefenstette & Nioche, 00]. The Irish language has less than 50 million words on the web. In addition, many of these languages are not based on Latin character sets and this may pose additional problems for a retrieval engine.

- **Users** – It is particularly ironic that the customer of each search engine should be one of the biggest problems facing web IR. The user poses the following main problems for a search engine:

    - Poorly articulated queries – Most users produce queries which are too short to adequately represent their information need [Silverstein et al., 98].

    - Unwillingness to browse - Only 15% of users look beyond the first screen (10 results) and 78% of users never modify their queries in light of results returned [Silverstein et al., 98].

    - Impatience for a prompt response – has lead certain search engines to only search subsets of their indexes, or use canned responses to popular queries. This is because people generally require very fast response times in order to maintain the illusion of the search process being an interactive one.

- **Search Engine Positioning (spamming)** – It is a fact of life for search engine developers that their search engines will be the subject of spamming attacks whereby individuals aim to improve the positioning of a (for the most part, commercial) website within search engine result pages. The basic idea is that since only 15% of users look beyond the first screen of results the website has to appear at or very close to the top of the ranked list in order to be visited by users. The optimization process often takes the form of careful keyword selection for web pages, but may also extend to artificially constructing the link structure around websites so as to aid the optimisation process. Many companies offer these facilities [WebPromotion, 02] [SearchEngineSerious, 02] with one company [HighSearchEngineRanking, 02] even stating that they "have seen some clients achieve a

650% increase in traffic" as a result of employing their services. An example, the following tips for circumventing the spam-preventing properties of linkage analysis algorithms (algorithms that operate based on the linkage structure of the WWW, which we shall discuss later in this chapter) are taken from Search Day [SEARCHDAY73, 01]:

1. "Signing guestbooks that are related to the theme of the website I am promoting. Then submitting the page".

2. "Also in Google in the search use + "keyword" + "add url" this will find all pages with your keyword plus the add url link so you can add your page to their links".

Notwithstanding of all these problems, a web IR system must accept many thousands of queries per second and return results to a user within a second or so. However the WWW does provide some additional sources of information besides the document text, which can be of benefit to the retrieval operation.

2.1.3 The Benefits of working with Web Data

WWW IR as opposed to conventional IR can draw on a number of additional sources of information to aid the retrieval process. Firstly, although the WWW is unstructured, chaotic even, a certain amount of information can be mined from the documents themselves. HTML tags impose a limited structure on web pages by distinguishing between different segments of a HTML document. It is possible to exploit this limited structure in the document retrieval process. By examining the HTML mark-up of a document it is easy to tell whether a term appears in the title, headings, is just bold for emphasis or is not marked-up for emphasis at all. Intuitively we believe that any text encoded with some of the 'important' tags is more valuable to the search process as these tags are more important than others. Consequently, it is often the case that text marked up as being one of the following types is integrated into the search process as text to be weighted more heavily than conventional text:

- Bold - … or …
- Italic - … or …
- Headline Text - <hx> … </hx>
- Title - <title> … </title>

- Meta Descriptions - <meta…>
- Text surrounding links <a href> … within a page.

It is even possible to evaluate text at the beginning of a document as being more important than text lower down a document's content. It is generally accepted that search results can be improved by considering this latent information. Certain document metrics may also provide a source of useful information. For example:

- A document with a shorter URL (a lower number of sub-directories) could be considered more useful if the user is specifically looking for a homepage or website. This theory was successfully evaluated as part of the latest TREC [Voorhees, 01] conference [TREC, 02].

- A user often considers a document with more images, or the inclusion of various media content, to be more useful than one with less media content, or containing text only. [Amento et al., 00].

- A popularity score for each web document can be generated based on the number of users who view documents returned in the results of search engines. This is known as behaviour based ranking. The more users view a certain document, the higher the score of that document's 'clickthrough-rate' (normalized by the position of the document in the ranked list).

Finally, the massive presence of hyperlinks on the web provides another source of useful information for information retrieval, which is the primary focus of this dissertation and which we shall discuss in far greater detail later.

2.1.4 Architecture of a Basic Web Search Engine

If we modify the IR system architecture outlined in Chapter 1 to reflect the additional information that we can mine from the structure of web documents we will get a more complex web retrieval system (shown in Figure 2.4), which must work with vastly larger datasets, into many hundreds of millions or even billions of documents. Note the inclusion of a web crawler *(1)* which gathers documents from the WWW and requires a URL queue *(2)* to store a list of documents to be fetched by the web crawler.

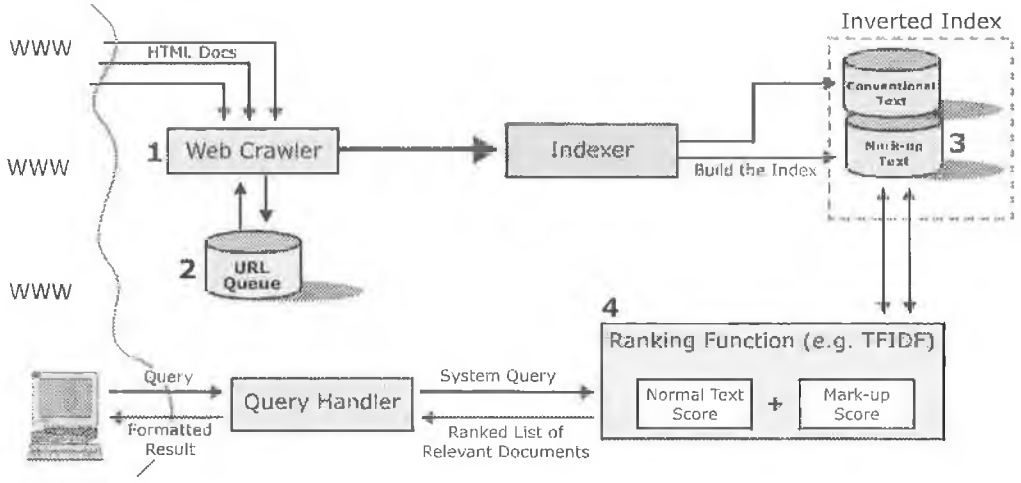


Figure 2.4 : Architecture of a basic web search engine

The approach to document representation that we have chosen to take in this example (and outline in Figure 2.5) is to have two logical documents for each single web document (3) (see Figure 2.5) with each logical document being ranked separately and the results combined (4) before final presentation to the user. In this way we combine evidence from two sources, although the number of sources may increase as required.

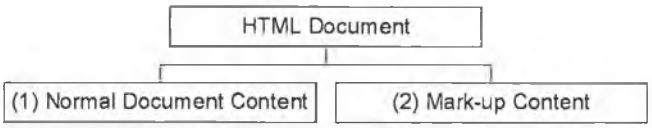


Figure 2.5 : Logical HTML documents

The two logical documents (1 & 2) from Figure 2.5 are based on the conventional text of the document *n* as well as the text within high scored tags within *n*. Letting *n_high* be the text associated with the high scored tags we have a search system which implements a simple ranking formula based on calculating the similarity *Sim* between a query *q* and a document *n* or *n_high*. Note that the normal text and mark-up text are weighted identically:

$$Sc'_n = Sim(q,n) + Sim(q,n\_high) \qquad (2.1)$$

Until recently, an approach such as this which took only the content of the document into account was the standard for most web search engines and the structure of the web had not played a part in the search engine process. Recently, however, that has all changed with the advent of a new set of techniques called Linkage Analysis.

2.2 Linkage Analysis as an augmentation to Web Search

A source of evidence that may aid the retrieval operation of WWW search engines is encoded in hyperlinks and this evidence can be extracted and used to aid retrieval performance. Hyperlinks have a massive presence on the WWW and our experiments show that the number of hyperlinks is up to 19 times the number of static documents (we will discuss these experiments in detail later). Exploitation of hyperlinks as an aid to web search is known as Linkage Analysis or Connectivity Analysis.

The goal of Linkage Analysis is to exploit latent human judgment on the web in the form of hyperlinks between documents in order to improve retrieval performance. In fact, as the WWW grows and it becomes increasingly difficult for standard IR approaches to operate effectively, the number of hypertext links, the building blocks of Linkage Analysis, are constantly increasing in number. This did not go un-noticed and even in the early days of web search hyperlinks were seen as a source of useful information. As early as 1993 a 'resource location tool[8]' called WWWW – the WWW Worm [McBryan, 94] was in operation as one of the first search engines on the Web. Interestingly the document text was not used in the retrieval process, rather the following was used to describe the content of a document:

- The title of the Document.
- Any reference hypertext (anchor text) from links within the document.
- The text within URL string of the document.

As of April 1994, the WWWW was receiving about 1,500 accesses per day and had an index of over 110,000 pages. The search engine was based on the UNIX egrep[9] program and generated an

[8] Resource Location Tool was an early name given to the systems that we now know as search engines.

[9] egrep is a UNIX program that supports searching source files for lines that match a regular expression

egrep search string based on the user's query. The WWWW is an example of an early search engine. These were soon replaced by larger, more powerful, search engines with architectures not unlike that in Figure 2.4. These search engines gained widespread popularity and use, yet they employed previously developed IR techniques and for the most part ignored the benefits that could be gained by examining the hyperlink structure of the web.

These first large-scale search engines that followed the WWWW did not innovate much in the approaches taken to searching. However, to be fair, they were not simply re-workings of previously developed systems. The standard IR techniques discussed in the precious chapter were developed for small document collections which were no larger than a few gigabytes. However the web was another matter entirely and consequently these first large-scale search engines have addressed the engineering problems of large-scale web spidering and efficient searching for large numbers of both users and documents. Now search engines such as Google [GOOGLE, 02], AltaVista [ALTAVISTA, 02] and Teoma [TEOMA, 02] have overcome these issues of scale and incorporate Linkage Analysis as an additional and integral part of their retrieval operation. Anecdotally this appears to have improved retrieval performance yet there has been little scientific evidence in support of the claims for better quality retrieval. As part of the three most recent TREC [TREC, 02] conferences, held in November 1999, 2000 and 2001 participants have been able to perform benchmarking of IR systems on web data and have had the option of using linkage information as part of their retrieval strategies [Hawking & Craswell, 01], all of which were found to be unsuccessful for regular search tasks[10]. We will discuss this in greater detail in later chapters.

Search engines, like any other IR system that incorporates term weighting, will score a web page in response to a query using some term weighting formula and will rank web pages according to these scores. Unlike conventional collections of independent documents, an additional source of latent information on the web is how documents are linked together and a search engine that exploits this linkage information combines information mined from the documents themselves, as well as information from the linkage structure of the web as a whole, into a final ranking formula. In most cases this linkage information is represented as a 'Connectivity' or a 'Linkage' score for each document which can be integrated into the weighting formula at query time.

[10] One task in TREC 2001, the homepage finding task did find that utilizing linkage information in locating a homepage response to a query improve retrieval performance over content-only experiments, however the top ranked experimental system from among all participants was based on simply utilising the length of the URL string as an indicator of whether a document was a homepage or not.

2.2.1 HyperLinks (Differentiating the WWW from a Conventional Document Collection)

A hyperlink (or link) on the WWW connects anchors on two (in most cases different) documents, the target (destination) document and the source document that contains the link, as in Figure 2.6. In HTML links are untyped and are one-way, i.e. one cannot follow a link backwards without using a 'go back' or 'history' facility in a WWW browser.

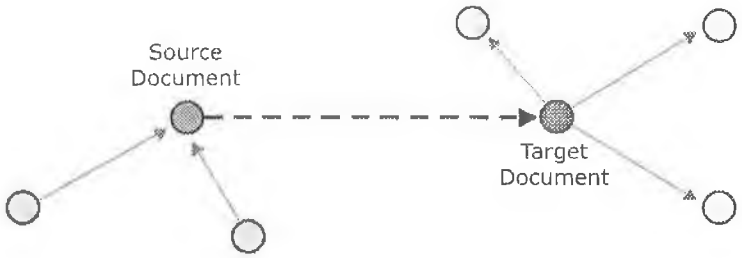


Figure 2.6 : A one-way hyperlink on the WWW connecting source to target

These links will convey some semantic meaning to the user regarding the content of the linked document by means of the human generated anchor text. This anchor text exists to provide a point from which the link has its origin and secondly to provide some indication of what the target document is about, as can be seen in any of the links (lighter coloured text) in Figure 2.7.

My research area is Web Search Engines, and (more precisely) how Linkage Analysis may be used to improve retrieval performance.

I have recently (4th February 2002) spoken at an [illegible] to the [illegible] in the [illegible]. The presentation was on [illegible] and the problems encountered by many in the research community when evaluating their algorithms and ideas.

Figure 2.7 : Links in HTML

Most hyperlinks in HTML are created using HTML similar to this:

```
<a href="URL">ANCHOR TEXT</a>
```

where URL is a Uniform Resource Locator or web address. However, in some cases links may be represented by images or imagemaps as opposed to anchor text descriptions.

2.2.1.1 URLs

URL is an abbreviation of Uniform Resource Locator. It is an Internet address which tells a user or a web browser where to locate specific data (file, application program, etc.). It can be thought of as a networked extension of the standard filename concept: for example, not only can it point to a file in a directory, but that file and that directory can exist on any machine on the web.

A URL is comprised of three distinct parts;

- A protocol - an identifier of the protocol (transport technique) used to fetch the file.
- A server - the name of the machine upon which the file exists.
- An identifier - a string that is converted by a web server into an identifier of the data being requested. Usually this takes the form of a directory path and a filename.

By combining these sections together, we get the unique path or address of each file on the WWW. For example, in the following URL:

```
http://www.ireland.com/dublin/entertainment/cinema/index.htm
```

The protocol is 'http', the server is 'www.ireland.com', the path is 'dublin/entertainment/cinema' and the file to be retrieved on this computer, in this directory, using this protocol is 'index.htm'. A URL may lack a file identifier, in which a default filename such as 'index.html' is assumed (at the root of the web server, or it may lack a directory path or filename element.

URLs are the method used to access documents in the web. The target of a link will be a URL so in the following example HTML:

```
<a href="www.ireland.com/index.html">Ireland Portal Page</a>
```

the target URL is 'www.ireland.com/index.html' and the anchor text description provided by the author of the page is "Ireland Portal Page". Therefore, a link now exists between the web page containing the hypertext link and the referenced web page 'www.ireland.com/index.html'.

2.2.2 Essential Terminology

Some terms which will be used throughout this thesis will now be defined. A hypertext-link associated with a document *d* will be referred to as an **in-link** of *d* if it starts in a different document

and points to document *d*. For an example see link 8 in Figure 2.8, which is an in-link of document F, originating from document G. If, however, the link starts in *d* and points to a different document then this link will be referred to as an **out-link** (see link 6, 4 or 10 as examples of out-links from document F). From graph theory, we know that the degree of a node is the number of directed edges incident to that node [COMAP, 96] so since each document may have multiple in-links and out-links we refer to the number of in-links into document *d* as the **in-degree** of *d* (the in-degree of F is 3). In a similar manner the number of out-links emanating from *d* is referred to as the **out-degree** of *d* (the out-degree of F is 3). Many links never cross document boundaries at all, these links originate in one section of a document and point to another section of the same document (link 2). These links are referred to as **self-links** [Agosti, 00] and carry no form of inter-document judgment and will only serve to facilitate a user browsing a long document.

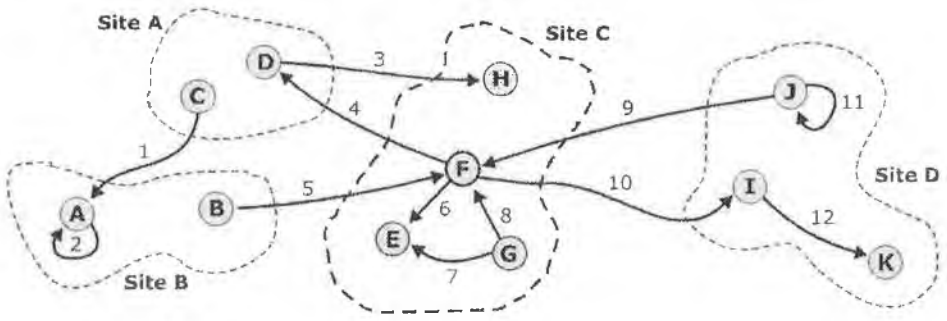


Figure 2.8 : Document Linkage

An author writing a WWW document will create semantically different types of hyperlinks between documents, even though HTML supports only one syntactic type of textual hyperlink (as described previously). Generally speaking, on the WWW we can separate links (both in-links and out-links) into either of two broad types based on their intended function when being created:

- **On-site** links connect documents within a particular domain. They exist to aid the user in navigating within a domain, or web site. On-site links can be further subdivided into upward, downward or crosswise links [Spertus, 97]. Upward links point to documents that contain a generalisation of the information in the source document. Downward links point at documents that are a specialisation of the topic explored in the source document, while crosswise links act as a link to different information on a different subtopic of the topic

explored in the parent document. In this way each web site can be constructed as a 'loose' hierarchy in a way not unlike the web directory mentioned in Chapter 1. As one descends the hierarchy the information contained within each page gets more specific. An example of an on-site link from Figure 2.8 is link 12, from document I to K, as the link never crosses a web-site boundary.

- **Off-site** (content, or outward) links on the other hand link documents in different domains (often across web site boundaries). They often link a source document to a target document that contains similar and, in the author's opinion, useful information. The reason why the author of a particular document will create the off-site link is because he/she is likely to have gained some usefulness or benefit from the content of the target document. After all, the creation of a content or off-site link implicitly imposes a cognitive load on the author, so the link will not be created without good reason. An example of an off-site link from Figure 2.8 is link 9 from J to F. Note that it originates in site D and points at a document in site C.

Finally, given that a document *d* may have many documents that link into it, we will often have to refer to this set of documents so we shall refer to this set as the **in-set** (the in-set of F contains documents B, G and J) and therefore the **out-set** (D, E and I) is referring to the set of documents that *d* points at. The size of the in-set is equal to the in-degree of that document, similarly for out-set and out-degree. To complete matters, we will need to be able to distinguish between on-site and off-site documents within the in-set or the out-set. Consequently we will refer to the **on-site in-set** of document F (in Figure 2.8 this contains G and not B or J) and the size of this off-site in-set is the **on-site indegree** of document F. In a similar manner we will refer to the **off-site in-set** of document F (this contains B and J and not G) and the size of this off-site in-set is the **off-site indegree** of document F. Similar logic is also applied to the **on-site out-sets** (for F this is a set containing E) and the **off-site out-sets** (D and I for document F).

2.2.3 Extracting Meaning from Links to aid Linkage Analysis

Recall from Chapter 1 that in the immediate aftermath of the Second World War, the US government began to pump huge amounts of money into research thus causing a huge increase the volume of scientific literature being produced. The need to provide quality IR facilities to this literature lead to the development of 'citation indexing' [Garfield, 01] which attempts to provide qualitative ranking of journals.

2.2.3.1 Impact Factors

Central to citation indexing is the 'impact factor' measurement. The impact factor of a journal [Garfield, 64], [Garfield 94] is based on two essential elements:

- the number of citations in the current year to any articles published in the journal over the previous two years.
- the number of sustentative articles published by the journal during these two years.

Letting j be a journal and IF_j be the Impact Factor of journal j, we have:

$$IF_j = \frac{\#\ Citations\ in\ last\ year\ (number\ of\ articles\ in\ last\ 2\ years)}{\#\ Published\ Articles(\ last\ 2\ years\ )} \qquad (2.2)$$

This "impact factor" was originally applied to medical journals as a simple method of comparing journals to each other regardless of their size. Garfield tells us that "the uncritical citation of disputed data by a writer, whether it be deliberate or not, is a serious matter" [Garfield, 64] and it is this that forms the cornerstone of citation indexing and approaches to linkage analysis.

Applying the concept of impact factors to the WWW gives us the 'web impact factor' which is based on the sum of the number of in-links or self-links into a page divided by the number of pages found on the referencing site or domain [Ingwersen, 98].

For the purpose of linkage analysis we are interested primarily in the number of (and later on, the quality of) citations that a web page receives. If a web page receives a lot of citations (in-links) then we can broadly conclude (ignoring document content) that this page may be a better page than one that receives significantly less citations. In addition to citation analysis, the Cluster Hypothesis [van Rijsbergen, 79] states that "closely associated documents tend to be relevant to the same requests". The process of an author constructing hyperlinks to other documents on the web indicated

a semantic relationship between the documents and therefore the utilisation of web links may provide valuable aids in the search process. Research carried out at IBM [Chakrabarti et al., 99] suggests that "Citations signify deliberate judgement by the page author. Although some fractions of citations are noisy, most citations are to semantically related material. Thus the relevance of a page is a reasonable indicator of the relevance of its neighbours, although the reliability of this rule falls off rapidly with increasing radius on average".

We are now in a position to state that the number of citations that that page has received from other web page authors (the page's in-degree) is an important measure and that data mined from these links could aid retrieval performance of search engines.

2.3.3.2 NOT ALL LINKS ARE CREATED EQUAL

When extracting information for linkage analysis from hyperlinks on the Web, two core properties of hyperlinks [Bharat & Henzinger, 98] can be assumed:

- **A link between two documents on the web carries the implication of related content**, otherwise the link would probably not have been created. There are scenarios in which this does not apply, examples being automatically generated links, or the links at the automatically included by free website hosting organizations such as Geocities [GEOCITIES, 02]. We will return to this problem later in this chapter when discussing the work of John Kleinberg.

- **If different people authored the documents[11], then the first author found the second document valuable.** We view all documents within a domain (connected via on-site links) as having being written/developed by the one author and thus represent the ideas of a single individual or organization, so any one author can't be allowed to influence the linkage score of documents within his/her domain. Of course, domains such as Geocities pose problems as Geocities contains many different authors' web sites within the one domain. In our experiments (outlined in the following chapters) we always treated "`www.dcu.ie`" as a separate domain to "`library.dcu.ie`", even though both were within the one organization.

[11] If the documents reside on different domains (are linked to via off-site links) then they are considered to be authored by different authors

Given that we emphasise off-site links, on-site links play a lesser role in linkage analysis and should not be viewed as a qualitative user judgment because it will not aid retrieval performance to allow individuals to directly influence the importance of their own web pages (vote for themselves). In this way one can help to prevent the problem of linkage-spamming, or search engine persuasion, whereby an author can artificially improve the ranking of a web page in search engine results by manipulating the document to rank highly in search engine result sets. This is, however, an ever-increasing problem for search engine designers.

2.3 Basic Connectivity Analysis Techniques

Linkage-based ranking schemes can be seen to belong to one of two distinct classes, from [Henzinger, 01]:

- Query-independent schemes, which assign a linkage score to a page independent of a given query. A score is assigned to a document once and used for all subsequent queries.

- Query-dependent schemes, which assign a linkage score to a page in the context of a given query. Additional hyperlink analysis is required for each query, but the benefit of this is that the hyperlink analysis can be tailored specifically to the query.

All linkage analysis techniques fall into one of these categories, with the former being the most common.

As previously mentioned we generally assume that the more popular a document is, the more in-links (higher citation count) of that document on the WWW. Therefore, given a set of query-relevant documents (perhaps returned from a Boolean IR system) one could rank them based on the number of off-site in-links into each document, which would be considered a query-independent scheme because the linkage score is not calculated at query-time nor is it dependent on the query. Let n be some web page and S_n be the set of off-site pages that link into document n. We can represent this as follows (assuming one link per page):

$$LSc_n = |S_n| \qquad (2.3)$$

In this case, the LSc_n score (linkage score) is based purely on the in-degree of document n. However this approach is very simplistic because the degree of overlap of the documents w.r.t. the query is disregarded, so the benefits if incorporating a (non-Boolean) term weighting scheme would be lost. For example if the relevant set of documents included documents ranging from the highly scored to documents of low scored then disregarding the relevance weightings could lead to lowly scored documents (if better linked) being ranked higher than the highly scored documents, or the highly scored documents never being seen by the user at all. Since this is obviously not the ideal situation, it would be far more useful to incorporate both scores into the ranking process thus gaining the benefits that both have to offer. Taking this approach, we utilise the document-query similarity score of each document and modify that to take into account a linkage score for that document. Once again, we let n be some web page, q be a query and S_n be the set of pages that link into document n and assuming some normalization of the two scores by α and δ:

$$Sc'_n = (\alpha \times Sim(q,n)) + (\delta \times |S_n|) \qquad (2.4)$$

This may seem like a very simple idea, but we know from experiments [Amento et al., 00] carried out at AT&T Research labs that implementing a linkage analysis component based on in-degree ranking has been found to be equally as good as other more advanced techniques (which shall be discussed in this and following chapters).

The main drawback of this approach (or similar) is that there is no attempt made to distinguish between the quality of a page pointed to (cited) by low quality pages and a page pointed to by the same number of high quality pages. So this idea needs to be expanded upon to incorporate a qualitative score for each link into a document and from a document. This qualitative score associated with a link would be based on the document from which the link originates. Consequently if a document has a link into it from Yahoo or the Open Directory web directory (as discussed in Chapter 1) then this document could be considered more useful that a similarly scored document (with an equal number of in-links) whose in-links come from web sites that could be considered to be less prestigious. It is this very fact that underlines most of the more advanced linkage analysis techniques that are found in this and later chapters.

Additionally, one can assume that a page with a large number of off-site out-links will act as an index page (as in the index of a book) and be a source of links to content that the author of the index (source) page found relevant and useful. Similarly, a page with a large number of cross-domain in-links could be seen as a popular page precisely because many people have found it useful, and consequently linked to it. One could take this a step further and say that a document on a topic such as Formula 1 Motor Racing which has in-links from a number of other documents relating to a similar topic, would be more useful to a user looking for information on Formula 1 than a document which may have a number of in-links from more diverse sources. We will now look at what techniques have been developed to incorporate linkage evidence (mined from the hyperlink structure of the WWW) to aid the retrieval process.

2.4 Building on the Basics of Linkage Analysis

Many more advanced techniques for exploiting the latent linkage information on the WWW exist, beyond that of simply counting the number of citations. We will now examine some of these.

2.4.1 Co-Citation Analysis

Authors, in general, cite works that fall within their subject area and since the creation of links requires a cognitive load, authors will not create links unnecessarily. This is often true on the WWW as well. A co-citation link exists between two articles if they are both cited by at least one other work. The strength of the co-citation link increases in relation to the number of articles that cite both [Garfield_2, 01].

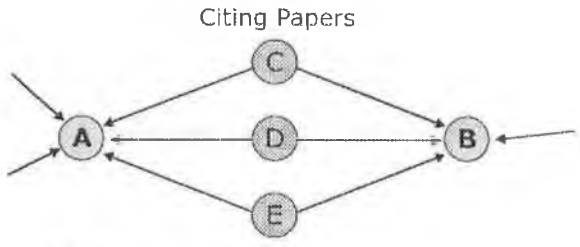


Figure 2.9 : Co-citation

In Figure 2.9 we can see that both documents *A* and *B* are cited by documents *C*, *D* and *E* so we can assume that documents *A* and *B* are related. If we can identify in a content-only search

session that document *A* is relevant, then we may also assume that document *B* is also relevant or at least increase the ranking of document *B* by some fraction of *A*'s weight.

2.4.2 Bibliographic Coupling

Closely related to the idea of co-citation analysis is bibliographic coupling where two or more articles are bibliographically coupled (related) if they contain citations to one (or more) other articles in common. Once again, the strength of the bibliographic coupling link increases as the number of common references increases [Kessler, 63].

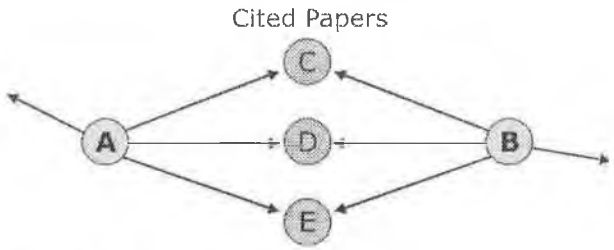


Figure 2.10 : Bibliographic Coupling

So in Figure 2.10 we can see that documents *A* and *B* can be considered bibliographically coupled because they both cite a number of the same documents *C*, *D* and *E*. Once again, if we can identify in a content-only search session that document *A* is relevant, then we may also assume that document *B* is relevant.

2.4.3 HyperLink Vector Voting

Hyperlink Vector Voting [Li, 98] is a query-independent and powerful approach to Linkage Analysis that has many uses extending beyond that of simply augmenting a ranking formula within a search engine. HVV allows for each link into a document to represent a vote for that document and to provide an associated description of that document's content. Thus a search service based on HVV uses the link anchor text from each citation as an indication of the semantic content of the target document and thus the number of citations (indirectly) affects the ranking of a document. This anchor text description can be the sole representation of a document's content, or be combined with the actual document content. The more links that a document has pointing at it, the broader is the resultant description of the content of that document and the more likely it is to be returned relevant in response to a query.

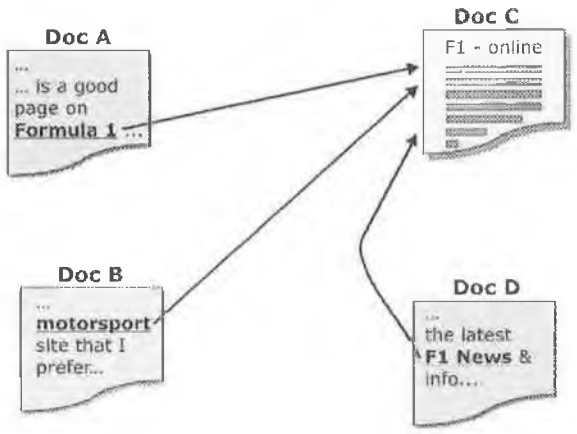


Figure 2.11 : Hyperlink Vector Voting illustrated

In effect, each in-link into a document can be thought of both as a vote for that document and as a source of information about the content of that document, with the documents content being described by others rather than by the author. In Figure 2.11, the content of document *C* would include, "`formula 1`", "`motorsport`" and "`F1 News`".

Letting *Sn* be the set of documents that cite document *n* (the in-set of *n*), we generate $Desc_n$ a textual description of the content of *n* based on the HVV method. Both $Desc_n$ and the user query *q* are represented as t-dimensional vectors[12], we could use this score for a simple query-document similarity calculation:

$$Sc'_n = Sim(q, Desc_n) \qquad (2.5)$$

2.4.3.1 Evaluating HVV

An experimental HVV-based Web search Engine called Rankdex [RANKDEX, 02] was developed to evaluate HVV. The total number of web pages indexed was 5.3 million documents. Since the document content is represented by the anchor text description of the in-links, the total inverted index size turns out to be 1/5 of the normally expected inverted index size for a similar sized dataset. The evaluation of Rankdex's performance [Li, 98] was based on sending 10 'popular' queries to what is referred to as an 'editors search engine'[13] and comparing how many of the top 10 results

[12] recall that *t* is the total number of index terms in the system.

[13] An 'Editor's Search Engine' is a manually constructed web directory providing search facilities over its high-quality index.

from Rankdex and a number of other search services were included in the high-quality manually generated results. The results are shown in Figure 2.12.

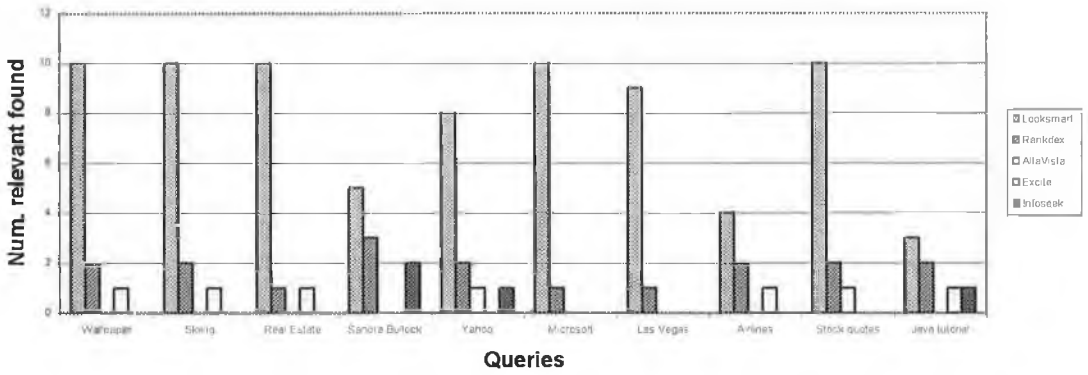


Figure 2.12 : HVV Evaluation

It can be seen that HVV most closely simulates a human editors efforts for all but one query, for which it is equal with Excite, and therefore was capable of meeting a user's information need better than the existing search engines could at the time. However subsequent experiments undertaken by researchers as part of the most recent TREC[14] conference (we will discuss TREC in detail in the next Chapter) have shown that HVV is of benefit in finding homepages of organizations and less so as an indexing technique. In addition, we can not know that the pages indexed by Rankdex and the 'editors search engine' are indexed at all by the other search engines that have been used in the experiment and the entire evaluation is done using a set of ten queries is so small as to make the results inconclusive.

2.4.3.2 Expanding on HVV

We can expand on HVV as it is described above by examining exactly what text is used for the anchor text descriptions. If one just takes an anchor text (text between the <a> and the HTML tags) then this may not provide adequate information as to the content of the destination document. It is the case, however, that in many cases no anchor text is associated with a link at all, or the anchor text is simply the word "here" or "link". Based on data from one of our crawls[15] of the

[14] TREC – Text REtrieval Conference, an annual IR conference which takes place each November in Gaithersburg, MD.

[15] Exploratory crawl of 126,997 web documents containing a total of 8,968,479 outLinks to 3,334,965 individual web documents which was carried out in January 2002.

WWW (specifically a subset of 2 million links from that crawl) we estimate that almost 13% of links contain no anchor text description and an additional 1 % are useless terms[16] for describing the content of a target document. Therefore 14% of links do not have anchor text descriptions that we consider to be beneficial to HVV, so we need to expand on HVV to gain any benefit from these 14% of links.

The approach taken is to extract text from the web page that surrounds the anchor text as a broader descriptor, both before and after. Care must be taken because, if too much text is extracted to form the anchor text description, we run the risk of including superfluous terms, which may not be appropriate and thus compromise the quality of retrieval. Research from IBM Research Labs [Chakrabarti et al, 98] suggests that a window of 50 bytes either side of the anchor text is optimal at describing the contents of the target document.

In an operational implementation of HVV we expect that the document content would be combined with the HVV document description, so if we let $Desc_n$ be the textual description of document n (generated using HVV) and q be the user query (both represented as t-dimensional vectors), with α and δ being constants used for tuning, we get the following formula to combine both sources of evidence at query time:

$$Sc'_n = \alpha * Sim(q,n) + \delta * Sim(q,Desc_n) \qquad (2.6)$$

2.4.3.3 Additional Benefits of HVV

Using a technique based on HVV (or a derivative such as that described above), it would be possible to provide search services over documents that a search engine has on its queue, but has not yet downloaded. Recall from Figure 2.2 that the Google search engine (as of July 2002) indexes 2,073 million documents, yet it has not downloaded and parsed all of these documents. It has only downloaded about 1,500 million documents and the other almost 600 million documents are represented by anchor text descriptions generated from documents that contain links into these documents. In Figure 2.13 we can see the true figures for Google. These non-downloaded documents can easily be identified by the lack of a 'cached' link after each document in Google's result pages.

[16] The 'useless terms' we speak about are : "more, links, link, view details, click here, here and about"

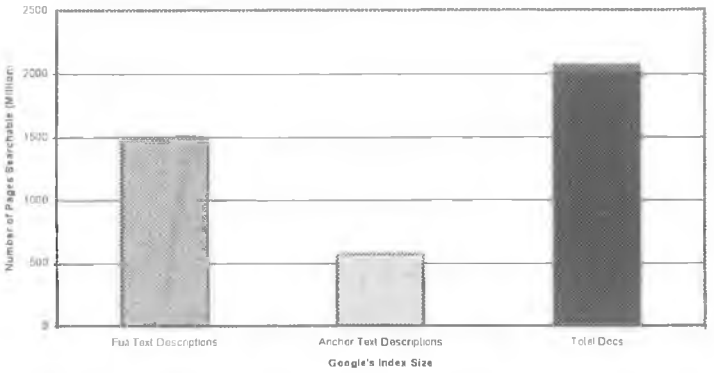


Figure 2.13 : Estimating the true size of the Google index

In addition, this process of associating the text of the source of a hyperlink does serve a purpose when one links to a non-HTML object such as image, video or audio data, all of which are not be searchable by conventional textual means. For example, if an image on the WWW has a number of in-links from various sources, the text associated with the source of the in-links can be used to generate content identifiers for the image [Harmandas et al., 97], and in so doing, providing a basis for the application of general IR principles on a textual description of the image. This textual description generated from the anchor text descriptions associated with links into an image, or even the text surrounding an image on a page is used as a basis for subsequent retrieval.

A less obvious benefit of using anchor text from in-links as a way to describe an object is derived from the fact that a third party describing a web page may use synonyms or different terms in describing the content of a document which could help to avoid the problem of 'aircraft' not being the same as 'airliner' and resulting in relevant documents being overlooked in the retrieval process. In addition there is the possibility of using the anchor text descriptions to identify synonyms, which can help in the building of thesauri or knowledge bases.

2.4.4 PAGERANK

The most visible linkage analysis technique in use on today's web is the PageRank [Page et al., 98] algorithm (as implemented in the Google search engine [Brin and Page, 98], [GOOGLE, 02]). This is another query-independent algorithm (post-indexing-time and pre-query-time) that generates a linkage score for each document in Google's index. Google themselves describe the algorithm as generating "an indicator of an individual page's value" and that, in a manner similar to Li's HVV, it "interprets a link from page A to page B as a vote, by page A, for page B" [Li, 98].

PageRank is an iterative algorithm, which generates a linkage score for each document in the inverted index. This score represents some weighting of that page, w.r.t. the pages that link into it. The algorithm can be described in terms of a random user's behavior while browsing the web. The user keeps clicking on successive links at random. However, problems can occur such as the user getting caught in page loops (where a page links to only one other page which itself contains just one link back to the referencing page). Rather than looping forever, the algorithm simulates a user getting "bored" and jumps to a new web page chosen at random using the E vector.

A simplified version of the PageRank algorithm, which ignores some of the issues that arise when viewing the WWW in graph theoretic terms, is discussed here. This algorithm iteratively calculates the PageRank of each document. Prior to calculation, each document is assigned an initial PageRank value as follows. Letting PR_n be the PageRank of document n, δ be a constant (usually 1) and N be the number of documents in the index:

$$for\ all\ n\ in N,\quad PR_n = \frac{\delta}{N} \qquad (2.7)$$

Once all documents have been assigned an initial PageRank, the iterative process begins and requires continual iterations[17] until an acceptable convergence level is reached. Given an in-set S for a particular document d, the PageRank of d is the sum of the PageRanks of every document in S divided by the out-degree of each document in S.

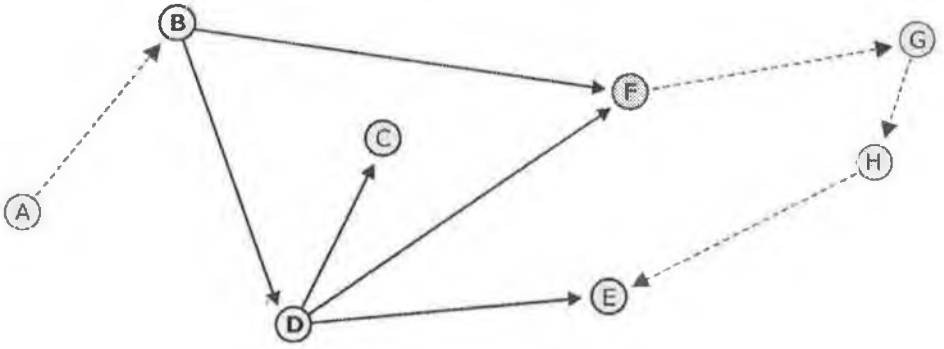


Figure 2.14 : A sample web graph to illustrate PageRank

[17] Based on experiments carried out by the author while developing a PageRank style algorithm (during an internship at AT&T Research Labs, NJ) the number of iterations is about 10 for 10 million documents. The PageRank authors put this figure at about 52 iterations for 322 million documents [Page et al., 98]. It can be seen that the number if iterations required depends in a large way on the size of the dataset among other factors.

In Figure 2.14 the PageRank PR_F of document F is equal to PR_B divided the out-degree of B summed with PR_D divided by the out-degree of D.

$$PR_F = \frac{PR_B}{2} + \frac{PR_D}{3} \qquad (2.8)$$

We can view a simple version of the PageRank algorithm, which generates a PageRank PR_n for a document from currently existing PR scores as follows. Let n be some web page and Sn be the in-set associated with n, let $outdegree_m$ be the size of the out-set of a document m and let c be a value that is used for normalisation during the iterative process:

$$for\ all\ n\ in\ N \qquad PR'_n = c \cdot \sum_{m \in S_n} \frac{PR_m}{outdegree_m} \qquad (2.9)$$

The algorithm to iteratively calculate PageRank scores for a set of documents can be written thus, letting N be the size of the document set, and PR' be a vector of temporary PageRank values:

$$PR_n \leftarrow \frac{1}{N}$$

loop :

for $n = 1,2,\ldots N$:

$$PR'_n = c \cdot \sum_{m \in S_n} \frac{PR_m}{outdegree_m}$$

end

$$PR_n \leftarrow PR'_n$$

while (not converged)

The algorithm will iterate until an acceptable level of convergence of PageRank values has occurred. Based on the linear algebra theory of eigenvectors we know that convergence will eventually occur as the number of iterations increased without bound.

One of the major benefits of a technique such as PageRank is the fact that all processing is implemented prior to query-time, generating a linkage score for each document which is included as part of the ranking formula at query-time. This means that no delay is required at query time which

make this type of approach much more applicable to use on the WWW. Additional details of PageRank can be found in Chapter 4, when we discuss the algorithm in more detail.

2.4.5 KLEINBERG'S ALGORITHM

From the previous section describing PageRank we know that a PageRank score is calculated for each document in the inverted index prior to the search engine ever processing a single a query and therefore it is a query-independent algorithm. An alternative approach would be to calculate a linkage score for each document after the query has been processed which would be a query dependent score. Such is Kleinberg's algorithm, which is actually quite similar to PageRank.

Kleinberg's algorithm [Kleinberg, 98] is also an iterative algorithm and in its original form is based purely on the linkage of the documents on the web. However, it does have some major differences:

- It is executed at query time (query dependent), and not at indexing time.
- It computes two scores per document (hub and authority) as opposed to a single score.
- It is processed on a small subset of highly scored (assumed relevant) documents generated by a content-only phase of the process, not on all documents as was the case with PageRank.

The fundamental idea behind the algorithm is that each web page is viewed as being of two types[18]:

HUB Page: a hub page is a page that contains a number of links to pages containing information about some topic, e.g. a resource page containing links to documents on a topic such as 'Formula 1 motor racing'. Each page has an associated hub score representing its quality as a source of links to content.

AUTHORITY Page: an authority page is one that contains quality information about some topic, an 'authoritive' page. Consequently, many pages will link to this page, thus giving us a means of

[18] Each page is both a hub and an authority, the strength of each type being based on that type's score.

identifying it. Each page also has an associated authority score representing its perceived quality by other people. Figure 2.15 shows an example of pages with high hub scores and high authority scores.

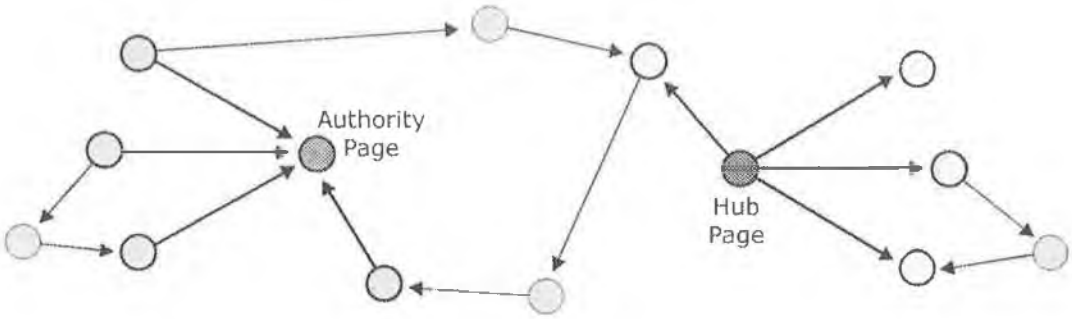


Figure 2.15 : Illustrating Hub and Authority pages

Documents with high authority scores are expected to contain relevant content, whereas documents with high hub scores are expected to contain links to relevant content. When examining the graph structure of the web, a recognised hub page links to many authority pages and a recognised authority page is linked to by many hub pages. Therefore, a document that links to many good authorities is a good hub and a document that is linked to by many good hubs is a good authority. A better hub is one that links to documents with higher authority and a better authority is a document that is linked to by many better hubs resulting in a mutually re-enforcing relationship. A document is not seen exclusively as a hub or an authority, rather each document will always have both scores and consequently be ranked in both lists. Although one would expect that a document with a high authority score would have a low hub score and vice-versa, this is not always the case.

The documents with the highest hub scores could be used to suggest documents to aid farther browsing while the documents with highest authority scores are documents that best satisfy the information need represented by the query.

2.4.5.1 The Search Process

The basic process of the algorithm to compute these scores is as follows; a user queries a search engine and which returns the top N (say 200) documents, referred to as the base-set. In Kleinberg's case the search engine was AltaVista. These documents represent a set of highly-scored documents, which are considered to be relevant to the query. The base-set is expanded along the off-

site in-links to and off-site out-links from these 200 documents (referred to as the neighbourhood) to produce an 'expanded-set' of documents (usually about 2,000). Each document in the expanded-set N begins with an identical hub and authority scores (usually 1.0) and the scores are updated according to the following formulae over a number of iterations. After each iteration the scores are normalised. The I operation updates the Hub Scores and the O operation updates the Authority scores. Assuming S_n is the in-set of n and that T_n is the out-set of n, the I operation is:

$$for\ all\ n\ in\ N,\quad Auth_n = \sum_{m \in S_n} Hub_m \qquad (2.10)$$

The O operation employing the same assumptions as the I operation is defined thus:

$$for\ all\ n\ in\ N,\quad Hub_n = \sum_{o \in T_n} Auth_o \qquad (2.11)$$

The authority and hub vectors will eventually converge, at which point the iterations can stop [Kleinberg, 98]. The convergence properties of the algorithm are based on standard results of linear algebra which states that the hub and authority weights will eventually converge as the number of iterations increases without bounds. Kleinberg found that that after about 20 iterations an acceptable convergence point is reached. The documents are then ranked into two groupings by hub (links to content) and authority (content) scores. Usually the top 10 documents from each vector are chosen to be presented to the user, the top 10 hubs as starting points for further browsing and the top 10 authorities as the best web sites to fulfill a users information need. The full algorithm is as follows:

$Hub_i \leftarrow 1$

$Auth_i \leftarrow 1$

loop :

for $n = 1,2,...k$:

$Auth'_n = \sum_{m \in S_n} Hub_m$ *for all n in N*

$Hub'_n = \sum_{n \in T_n} Auth_o$ *for all n in N*

Normalise $Auth'_n$, *obtaining* $Auth_n$

Normalise Hub'_n, *obtaining* Hub_n

end

while (not converged)

Example results for the query "search engine" are shown in Table 2.1 as the top documents returned by Kleinberg's algorithm and the results of the same query sent to the AltaVista search engine, in September 1999.

TOP 5 AUTHORITIES	TOP 5 ALTAVISTA RESULTS
Yahoo	Beaucoup Search Engine List
Excite	Register with S.E. Web Site Page
Magellan	Mamma Meta Search (about)
Lycos	Mamma Meta Search [mamma]
AltaVista	Search Engine Links

Table 2.1 : Comparing Kleinberg to AltaVista

Kleinberg's algorithm has been implemented as the HITS (Hyperlink Induced Topic Search) System at IBM [Chakrabarti et al., 98]. Intuitively we can see that the Kleinberg Algorithm produces better results in this case and intuitively we can see that the algorithm is very appealing and should yield high levels of retrieval effectiveness. However, proving this the case has been fruitless in experiments reported to date for all TREC participants in the Web Track[19] over the last three years. In the next chapter we will look into this problem in greater detail.

[19] TREC Web Track – Web IR evaluation experiments organised by NIST, further details of this are presented in Chapter 3

However intuitively this algorithm seems to work in the above example, it does not work well in all cases and a number of suggested improvements to this algorithm have been put forward by Bharat and Henzinger [Bharat & Henzinger, 98].

2.4.5.2 Improvements to Kleinberg's Algorithm

Three major weaknesses have been identified with Kleinberg's Algorithm [Bharat & Henzinger, 98], recall the two assumed properties of web hyperlinks from page 46. These problems can be described under the following three headings:

Mutually Re-inforcing Relationships Between Hosts

Sometimes a set of documents on one host point to a single document on a second host, serving to increase the authority score of the document on the second host and the hub scores of the documents on the first host. Since we assume that all documents on one host were authored by a single entity then this gives undue weight to the opinions of one person. Ideally, one would like all documents on a single host to have the same influence on the document they are connected to as a single document would. To achieve this, one gives fractional weights to links [Bharat & Henzinger, 98] in such cases (in so doing we view documents in terms of their hosts). If there are n links from documents on one site to a single document on a second site we give each edge an weight of $1/n$. Likewise for hub weight, a similar fractional technique is applied. This results in viewing links on the web-site level as opposed to the document level. In Chapter 4 we discuss how we applied this solution to PageRank instead of HITS while developing an alternative version of PageRank.

Automatically Generated Links

In the case of automatically generated links, the second assumption regarding the properties of web pages will not apply in many cases, as these documents were not created by an individual representing a value judgment for a document. This is solved by combining content and linkage analysis [Bharat & Henzinger, 98], so that we can determine the relevance of a document to the query topic and either eliminate irrelevant documents or regulate their influence. For example the hub score of a document is usually dependent on the sum of the authority scores of the documents it links to, but the transfer of these scores is now regulated by the similarity of each document to the original query, or an expanded version of same.

NON-RELEVANT NODES

This is perhaps the biggest flaw with many linkage analysis algorithms. In Kleinberg's approach it is often found that many of the documents comprising the neighbourhood graph would not be highly scored documents (with respect to the query topic). While the initial base-set would have continued many highly scored documents, the expanded set may contain a much lower density of highly scored documents. This can cause a problem known as 'topic drift' whereby the most highly ranked hubs and authorities tend not to be about the original query topic. Quite often 'topic drift' leads to the highly ranked documents being about a broader topic than the query. For example, a query on 'java socket exception' may produce the top ranked results all concerning java or computer programming in general. Once again this problem is solved by combining content and linkage analysis and regulating the influence of each document depending on its relevance to the query topic.

Initially query-document relevance was only considered in generating the base-set. However the document relevance (content) scores and linkage analysis are used when calculating the Hub and Authority scores and this changes the formulae that are used to calculate both scores by regulating the influence of each document based on its relevance to the query topic. Letting W_n = the query-document similarity score of document$_n$ the new formulae to calculate hubs and authorities are shown below.

The I operation:

$$for\ all\ n\ in\ N,\quad Auth_n = \sum_{m \in S_n} Hub_m \times W_m \qquad (2.12)$$

And the O operation:

$$for\ all\ n\ in\ N,\quad Hub_n = \sum_{o \in T_n} Auth_o \times W_o \qquad (2.13)$$

It should be noted that the query text is often not of sufficient quality to adequately represent the topic of the query. Consequently, the documents of the start set are used by Henzinger and Bharat to define a broader query to match documents against in a modified form of automatic query expansion, as discussed in chapter 1. More precisely the top 1,000 words of each document are used to generate a large expanded query, which is used to calculate the W_m and W_o values.

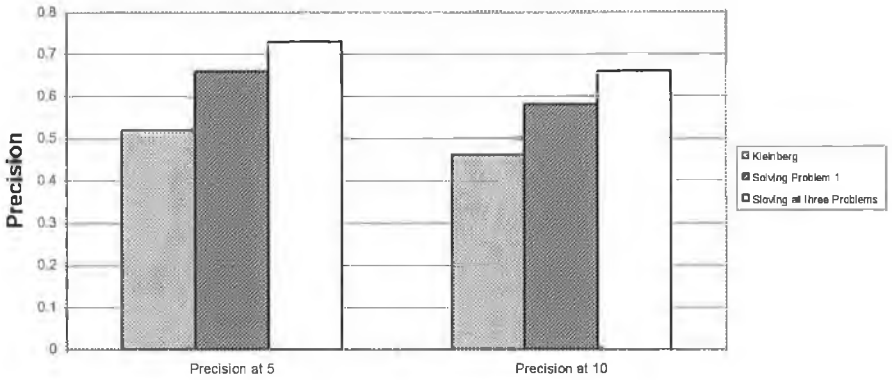


Figure 2.16 : Comparing basic Kleinberg with Bharat & Henzingers' improvements

Figure 2.16 shows the improvements gained by Bharat & Henzinger over the baseline result of Kleinberg's algorithm. Only the results for the top authority documents are shown here. Precision is plotted at both 5 and 10 document levels of recall for each of the three approaches. It can be seen that at both recall levels that even solving just the first problem mentioned above increases quality (precision[20]) noticeably (by 26% at 10 documents). Solving the other two problems leaves us with an additional increase of 12% at recall of 10. Therefore, we can see that incorporating content analysis into Kleinberg's algorithm produced beneficial results.

A final point about the similarities between PageRank and Kleinberg's algorithm is that Kleinberg requires similar memory requirements (during calculation) to those of PageRank, even though it produces two scores for each document. Recall that PageRank requires the storage of the old PageRanks from the previous iteration until the current iteration has completely concluded.

2.5 Architecture of a basic Linkage Analysis based WWW search system

Augmenting the architecture of the search engine presented in Figure 2.4 to incorporate a linkage analysis component adds greatly to the overall complexity of the system. A new component

[20] We will discuss qualities measures for evaluating IR techniques in the next chapter.

called a Connectivity Server is required to serve up timely linkage information, such as the off-site in-set, or on-site out-set of a document.

2.5.1 CONNECTIVITY SERVER

Any software that implements any form of linkage analysis over web pages must work in conjunction with a 'connectivity server'. A connectivity server [Bharat et al., 98] is a software application that provides fast access to the neighbourhood of any referenced URL, although a conventional DBMS may suffice if speed is not of vital importance. The DBMS approach is how we developed most connectivity servers that were required for our experiments.

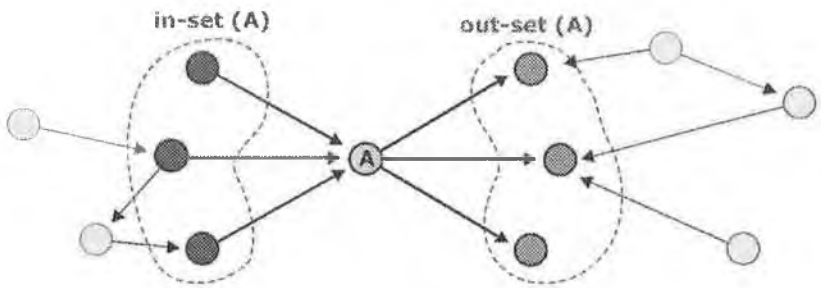


Figure 2.17 : Representing the Structure of the Web

Figure 2.17 shows the neighborhood graph of a base URL. It is quite possible that a document in the in-set may also exist in the out-set, as would be caused by a base URL page linking to another URL, which in turn links back to the base URL. Within documents on the same server, this is quite common. The primary requirements of a connectivity server can be defined as follows:

- To return the in-set or out-set of a document in response to a query.
- To return the in-degree or out-degree of a document in response to a query.
- To provide all responses in a timely manner.
- Possibly to return the in-link anchor text of a document.

In a manner similar to a conventional web search facility, the connectivity server must allow for updates to be made to its index, which must not affect the workings of the server. AltaVista used batch updates nightly to their prototype connectivity server. This was mainly due to the

composition of the underlying data storage system upon which the connectivity server was constructed. Recall that speed is a priority in a realistic WWW setting, with many thousands of queries being handled per second. In addition, a connectivity server may also be expected to respond to a query with a data structure containing documents and their associated anchor texts were this required.

A connectivity server views the WWW as being represented as a directed graph with a finite, non-empty set of nodes representing the documents and directed edges representing the links between any two documents in the graph. Before the connectivity server can represent a graph (G) of the WWW, we need a method of representing it. The adjacency matrix of the graph is a suitable method of doing this. In the adjacency matrix $A(G)$ for G, the entry in row i and column j is 1 if the nodes i and j are joined by an edge and 0 otherwise. See Figure 2.18 (left hand side) for an example of an adjacency matrix. Of course, an adjacency matrix for the entire WWW at a given point in time would be enormous, sparse and very difficult to model. Hence, some form of compression must be used, for example, CCS [Duff et al., 89] or CRS could be used in a manner similar to the compression of the term-document matrix discussed in Chapter 1. Assuming the adjacency matrix $A(G)$ can represent the out-links of the set of documents, the in-links into these documents can be represented by $A^T(G)$ which is the transpose of the matrix $A(G)$, so we do not have to keep separate matrices.

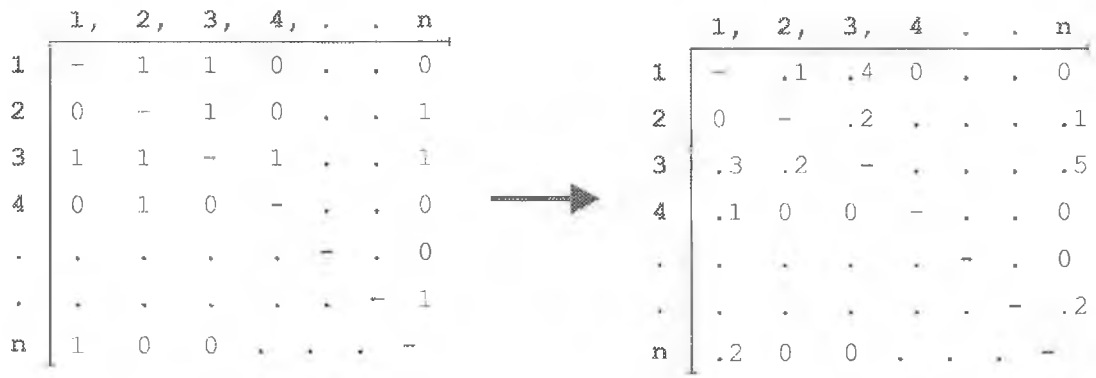


Figure 2.18 : Binary (left) and Weighted (right) Adjacency Matrices

By simply replacing the binary values in the graph with range values a weighted graph can be easily generated in which each edge has an associated weight depending on its relative importance, and/or the importance of the node from which it originates or points at. Figure 2.18 (right side) shows an example of a weighted adjacency matrix, which is independent of the binary matrix.

2.5.2 LINKAGE ANALYSIS SEARCH SYSTEM ARCHITECTURE

Based on what we have learned thus far regarding IR and linkage analysis we are now in a position to describe one possible architecture of a basic linkage-based search and retrieval system. Building on the architecture of the previous model) we can see that some additions have to be made to the system (shown in Figure 2.19) to reflect:

- The requirement for a connectivity server to serve up accurate and timely linkage data *(1)*.
- The additional source of content for indexing, which is the anchor text of the in-links and a window at either side around the in-links into the document *(2)*.
- A link analyser tool to generate a linkage score for each document such as PageRank, which is utilised by the system at query time *(3)*.

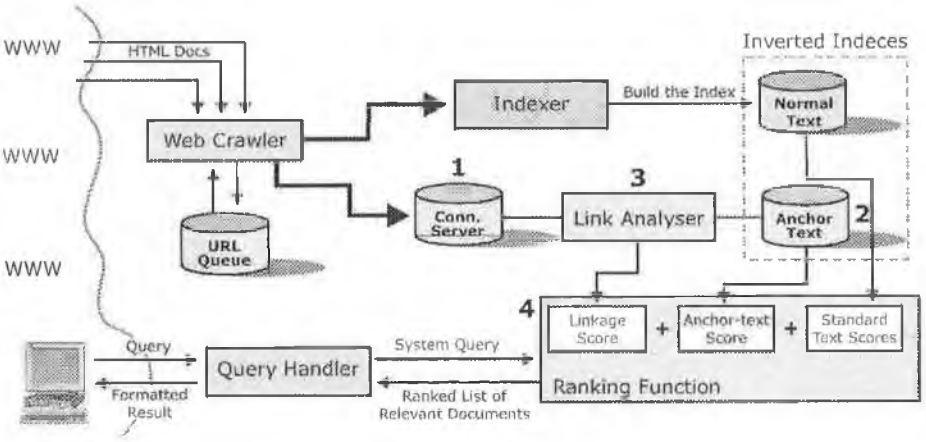


Figure 2.19 : Outlining the architecture of a search engine incorporating Linkage Analysis

The linkage score is integrated into the weighting formula to carry a certain influence on the overall score of the document. The influence of this score would be regulated (in most cases) by a constant, as would also be the standard text score[21] and the anchor-text score. These constants would be represented by best guess values as is the case with AT&T's TREC experiments [Singhal & Kaszkiel, 00] where the value chosen for the regulation constant associated with text score(s) was 1.0

[21] The mark-up text document description has been removed from the architecture to avoid unnecessary complexity.

and the anchor-text regulation constant was set at 0.25. No direct linkage score was used in the AT&T experiments.

Letting $Sim_{Desc(d),q}$ be the similarity of the in-link anchor texts to the query, $Sim_{d,q}$ be standard query document similarity for the document content and PR_d be some PageRank style linkage score for a document and α, δ, λ be constants to regulate influence of different components, we can generate a final weighting for a document d, from Figure 2.19 *(4)*, based on:

$$Wt_d = (\alpha \times Sim(d,q)) + (\delta \times Sim(Desc_d, q)) + (\lambda \times PR_d) \qquad (2.14)$$

In this way we can incorporate linkage analysis into a web search engine using linear combination in which the constants would require tuning, unless an alternative technique is chosen, such as the hub synthesis model [Achlioptas et al., 01].

2.6 Chapter Summary

In this chapter, we have identified the challenges of searching the WWW, but also identified the benefits that can be gained when working with web data before describing the architecture of a simple WWW Search Engine. One possible benefit that we can extract from working with web documents is the ability to incorporate linkage analysis into the retrieval process. Linkage analysis is the name given to a technique that mines latent human judgments from the very link structure of the WWW. We have outlined the essential terminology for the rest of the thesis and the assumptions necessary for incorporating linkage analysis into the retrieval process.

When discussing linkage analysis, we have identified the main techniques that can be implemented, from basic citation ranking or HVV to the more advanced algorithmic processes such as PageRank or Kleinberg's algorithm.

Citation ranking is based on the work of Eugene Garfield in producing the 'impact factor' measure for journals and is based on the idea that the more people link into a document, the more that document can be considered a useful document. HVV utilises anchor text descriptions of in-links

into a document to provide a textual description of the content of the document for subsequent conventional indexing and retrieval.

PageRank is a query-independent linkage analysis technique that assigns a single numerical score to each document, which is combined at query time with a content-only score to produce a final document ranking. Kleinberg's algorithm, on the other hand is a query dependent algorithm that operates on a small set of highly scored documents and produces two scores for each document. These two scores are called hub and authority scores with hub scores reflecting a document's usefulness as a source of links to possible relevant content and authority scores representing the usefulness of tee document itself with regard to a particular topic. Two final document rankings may be produced, by ordering documents in decreasing order of hub and authority scores.

Finally, we have described the architecture of a WWW search engine that incorporates a linkage analysis component.

Chapter 3

EXPERIMENTS IN LINKAGE BASED INFORMATION RETRIEVAL

We open this chapter with a discussion of the concept of relevance before we describe the common approaches to measuring retrieval performance of a conventional information retrieval system. The principles of performance which provided guidelines for our linkage-based experiments are outlined before we discuss the TREC series of conferences and our experiments for the web track of both the TREC-8 and TREC-9 conferences. Our linkage-based experiments and the findings from these experiments are discussed. The findings of these experiments were not positive as we failed to improve retrieval performance by incorporating linkage analysis into the retrieval process, as indeed was the case with all other TREC participants.

3.1 EVALUATING IR SYSTEMS

In order to determine the quality of an Information Retrieval system an evaluation of the system is usually carried out. Naive measurements of a system's performance can be done in terms of time and space. The shorter the response time taken from when a user submits a query to when the same user receives ranked output, the better the system is considered to be. In addition, lower memory requirements are seen as beneficial because larger numbers of documents can be processed without increasing memory requirements. Additional measurements [Cleverdon at al., 66] include the following:

- Coverage: the extent to which a document collection contains relevant material.
- Presentation : the form of presentation of the output of the search process.
- Effort: the load on the user in obtaining answers to an information requirement.

However, these measures do not actually provide an indication of the effectiveness of the retrieval system in satisfying a users information requirement. It is assumed that the more effective a system, the more it will satisfy a user. To gain an indication of a retrieval system's effectiveness we employ the Precision and Recall metrics and introduce the concept of relevance.

3.1.1 RELEVANCE

A document that satisfies a user's information requirement is said to be relevant to that information requirement. The notion of relevance should not be confused with highly scored documents (those ranked highly by an automatic retrieval system), rather relevance is based on human judgements of the utility of a document in satisfying an information requirement and a document can not be assumed relevant simply because automatic retrieval system has allocated it a high score.

Keith Van Rijsbergen [van Rijsbergen, 79] states that "relevance is a subjective notion" and that "different users may differ about the relevance or non-relevance of particular documents to given questions". Yet in the field of information retrieval, relevance judgements made by human subject experts in a particular area are acceptable as a basis for retrieval performance evaluation and do not invalidate experiments based on document collections (incorporating a set of queries and corresponding relevance judgements for these queries). Thus human judgements form the basis of qualitative ranking of Information Retrieval systems.

It is also a general assumption [van Rijsbergen, 79] in the field of information retrieval that should a retrieval strategy or algorithm fare well (rank the most relevant documents highly) under a large number of experimental trials then it is also likely to perform well in an operational situation, a situation in which relevance is not known in advance, such as searching the web. The assumption being made is that these experimental trials should be based on executing representative queries on a document collection, which is also representative of the nature of documents in an operational scenario. Consequently, should a retrieval strategy or algorithm produce unfavorable results under experimental trails on representative documents then the validity of the retrieval strategy or algorithm can be questioned. However, if the queries or the document collection are not representative of real world queries to documents, then the validity of the experiment results could be called into question.

The TREC series of conferences, which we shall discuss at length in this chapter, supports the evaluation of retrieval strategies by providing participants with data (upon which experiments into retrieval strategy performance are executed), queries and assistance in evaluating the retrieval effectiveness of participants experimental retrieval strategies.

3.1.2 MEASURING RETRIEVAL PERFORMANCE

Consider a user's information request I executed on a test collection of documents to be referred to as T. In order to accurately evaluate the performance of a retrieval system we need to know what documents are relevant (we will return to this problem later) and we refer to this set as the set R of relevant documents. Let $|A|$ be the number of documents that a particular retrieval strategy has ranked in response to the information request I and $|R|$ be the number of documents actually relevant to the information request I. In addition let $|RR|$ (Relevant Retrieved) indicate the number of documents in the intersection of sets R and A. This signifies the number of relevant documents retrieved, see Figure 3.1.

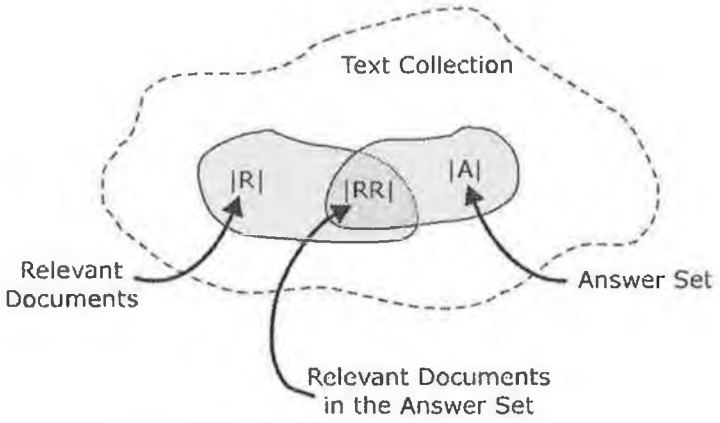


Figure 3.1 : Illustrating Precision and Recall

The measurements of precision and recall are based on these sets of documents.

Precision : is the fraction of retrieved documents that are relevant, i.e. the proportion of the set A that is relevant for a particular query and precision is represented by the following formula:

$$Precision = \frac{|RR|}{|A|} \qquad (3.1)$$

Recall : is the fraction of relevant documents that have been retrieved, i.e. the proportion of the set R that has been retrieved for a particular query and recall is represented by the following formula:

$$Recall = \frac{|RR|}{|R|} \qquad (3.2)$$

One assumption underlying Precision and Recall is that all documents in the answer set A have been judged (by a human judge, as opposed to being simply highly scored by an automatic retrieval system) to be either relevant or not-relevant prior to the calculation of the Precision and Recall values. For a large test collection, this is clearly not practical and we will see later how this is handled by TREC.

Due to the fact that a user usually doesn't see all scored documents resulting from a given query (a ranked list of the top N documents is normally presented instead) proper evaluation of retrieval performance includes measuring precision at fixed levels of recall and (often) plotting this on a precision versus recall curve. This precision versus recall curve (as shown in Figure 3.2) is usually based on 11 standard recall levels, which are 0% through to 100% at 10% intervals. The precision figure for each of the 11 points of recall is based on an interpolation procedure whereby the interpolated precision at the *n-th* standard level of recall is the maximum known precision at any recall level between the *n-th* and the *(n+1)th* level (iteratively). So the precision value at 10% recall is based on the maximum precision value between the 10% recall level and the 20% level and so on. An example precision versus recall curve for four of our experimental retrieval strategies (TREC-9 experiments) is shown in Figure 3.2.

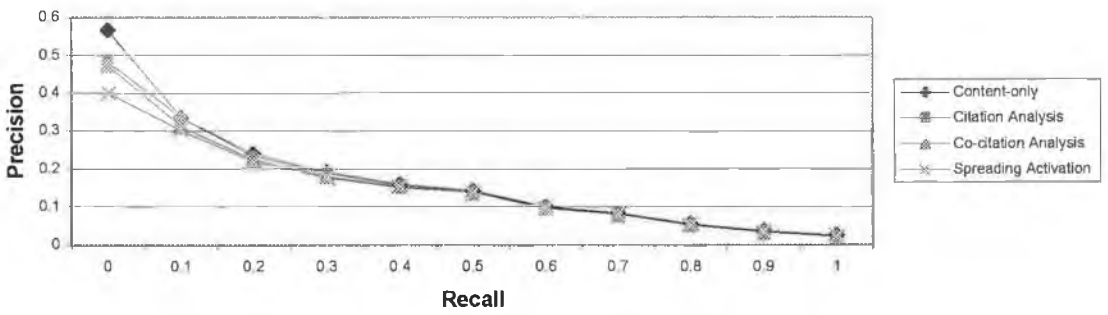


Figure 3.2 : Precision versus recall graph for our TREC-9 experiments

Given the fact that a user often only views the top 10 ranked documents resulting from a search then averaging the precision figure at 10 documents would be a useful measure of retrieval performance, i.e. how many documents ranked in the top 10 results are actually relevant. This is often

carried out at certain cut-off values such as 5, 10, 15, 20, 30, 50, 100, etc. and is plotted on a graph referred to as a precision ranking graph. A precision ranking graph will display the precision values for an experiment at fixed points (as above) in the answer set. For example, at point 10 on a precision ranking graph one can see the (non-interpolated) average precision of the top 10 ranked documents. A precision value of 1.0 would indicate that all top 10 ranked documents are considered relevant to the query topic. It is this measure that we will employ most often in evaluating the results of our experiments.

Usually retrieval algorithms are evaluated over a fixed number of queries and the average precision figures at each recall level is used. Therefore, if fifty queries are used as part of an evaluation process, the average precision at each recall level is based simply on summing the precision values at each level and dividing by the number of queries executed (in this case fifty).

One final measure that we will need to understand is referred to as generality (vanRijsbergen, 79). The generality of a document collection is a measure of the density of relevant documents within the collection and is represented by the following formula, where N is the number of documents in the collection.

$$Generality = \frac{|R|}{N} \qquad (3.3)$$

We will use the generality measure in this thesis to compare different document collections to one another to examine their support for various ranking algorithms and strategies.

3.1.3 Test Collections

The field of text information retrieval has a long tradition of using test collections as part of the evaluation process [Harman, 92]. Test collections, in general, consist of three components:

- **a set of documents (a dataset),** which should be representative of the documents which would be encountered in an operational situation. Yet, in certain circumstances this is not even sufficient, a case in point being experiments in linkage analysis techniques, which will require the linkage structure between documents with the test collection also to be representative of the link structure between documents on the WWW.

- **a set of queries,** which should be representative of the types of queries which would be encountered in an operational situation. In a WWW search evaluation scenario, one obvious source of queries would be from search engine query logs, which are logs of user queries that search engines produce and periodically make publicly available.

- **a set of relevance judgements** for use in evaluating the performance of an information retrieval system. Relevance judgements consist of a listing of all relevant documents (as exhaustive as possible) identified from the test collection set of documents, for a particular query. There will be a set of relevance judgements for each query. These, relevance judgements will have been generated by subject matter experts.

One of the major benefits of having relevance judgements available is that it is very easy at any time to run additional experiments and evaluate retrieval performance. Relevance judgements in early test collections were complete, that is, a relevance judgement was made for every document in the collection for every topic. Once the test collections began to grow in size one immediate problem encountered was how to obtain a complete and accurate set of relevance judgements for the queries. From [Grefenstette, 97] we can infer that it will take over 13,000 hours to judge the 1.69 million document dataset used recently in the TREC (see below) conference for just one query. This is clearly unrealistic as it would require 74 man-years to generate compete relevance judgements for all fifty queries for one years TREC experiments. One popular solution is the pooling technique where, for each query, a number of alternative ranking algorithms each submit a set of ranked documents. These documents are added to a pool of 'candidate' documents for future judging by a human assessor (subject matter expert and in some cases, the topic author) thus dramatically reducing the number of documents that require judgements.

Of course, it is possible that a number of relevant documents from the collection will not be contained in the pool and therefore these documents will not be judged, and by default will be considered to be irrelevant and we refer to these relevance judgements as non-complete relevance judgements. However the benefits of having relevance judgements at all far outweigh the problems with using non-complete relevance judgements on test collections.

One of the earliest test collections (from the 60s) was the Cranfield collection [Cleverdon at al., 66], which consisted of 1,400 documents (requiring 1.6MB of disk space) and included 255 queries and complete relevance judgements, which were not too difficult to produce given that the document collection only contained 1,400 documents. Following on from the Cranfield collection came other

notable collections such as the CACM collection [Fox, 1983] and the NPL collection [Sparck Jones & Webster, 79]. But these early datasets were small and as noted by Sparck Jones in 1981, there was "little or no consolidation between research groups in the field of IR" [Harman, 92]. There was a need for larger test collections to evaluate the then existing methodologies on realistic sized datasets. This challenge has been met by the TREC series of conferences, with the latest test collections consisting entirely of web data, and being used to test experiments into WWW information retrieval.

3.1.4 Evaluating Linkage Analysis

The usual output of a linkage-based retrieval algorithm is a ranked list of documents, which is similar to the output of a conventional IR system. The evaluation of such an algorithm normally follows the same procedure as conventional IR evaluation, in that we measure retrieval performance using the standard measures of precision and recall. However, since linkage analysis is based on exploiting the link structure of a document collection, simple precision and recall measures may not be adequate to evaluate linkage based retrieval strategies. The fact that a document may be scored lowly, but act as a source of links to highly scored documents, would mean that it would not score well using standard precision and recall measures. The effect of this would be that documents that would be highly scored as 'hub' documents (which would be useful as starting points to explore a topic) using Kleinberg's algorithm may (content depending) not be seen as relevant using current methodologies. It is our belief that the current measures do not adequately support retrieval performance evaluation of linkage based retrieval strategies and that new measures need to be developed which are more suited to evaluating the neighbourhood of a document as well as the document itself. However, this is beyond the scope of this thesis, where we have worked within conventional IR evaluation methodologies, but we will make allowances.

Should linkage-based retrieval (linkage analysis) be found to aid retrieval performance, then we believe that certain criteria must be met by the linkage analysis algorithm which we refer to as the 'principles of performance' of a linkage analysis algorithm.

3.1.4.1 Principles of Performance of a Linkage Analysis Algorithm

Our experiments in the field since 1998 have lead us to develop a number of requirements that any operational linkage analysis component, being developed for a Search Engine, should meet. We refer to these as the "Principles of Performance" for a linkage analysis component and any linkage analysis component:

1. *Must provide a useful & accurate connectivity score.* This score must accurately reflect a document's influence on the dataset being indexed and should aid retrieval performance.

2. *Must not adversely affect the query-time performance of the retrieval system.* That is, an approach such as Kleinberg's algorithm would have a query-time performance hit if implemented at query-time as a result post-processor. This was not an option in our experiments and unless Hubs and Authorities calculations could be integrated into a system withour causing any notable delay in query processing we would avoid such a technique.

3. *Must be scaleable to realistic sized datasets.* If one designs such a component for use with Web data then limiting the capabilities of the system should be avoided at all costs. This is influenced by a number of items; addressable & available memory, processor speeds and network throughput. Memory limitations of storing the arrays of PageRanks in RAM would have limited any single PC based algorithm to below 250 million documents assuming 2GB of available RAM. Early versions of PageRank [Brin & Page, 98] used to write the old PageRanks of each document to disk and only store the current PageRanks of documents in RAM, thus instantly doubling the capacity of a single computer, with no query-time performance hit.

4. *Must be robust in the face of WWW linkage irregularities and links of varying importance.* Link irregularities such as such as link circularity[22] must be handled competently. They must not result in undue scores being applied to non-deserving documents. All links are not created equal and this is something that should be addressed also. Bharat & Henzinger [Bharat & Henzinger, 98] regulate the influence of a link by the source node's similarity to an automatically expanded query, as we have seen in Chapter 2.

[22] Linkage circularity refers to particular structure of links between web pages that form loops and can cause problems for linkage-based retrieval algorithms.

3.2 TREC, THE TEXT RETRIEVAL CONFERENCE

TREC (Text REtrieval Conference) is an annual conference (since 1992), funded by DARPA[23] and organized by NIST[24], which draws participants from all over the world each year to take part in benchmarking exercises for information retrieval related tasks. Its aim is to provide a framework within which diverse research groups from around the world can run experiments on identical data using queries provided by the TREC organizers, and then come together to share their results and findings. Each participant writes a paper/report on their experiments and (usually) makes their algorithms publicly available so that the field as a whole can benefit.

As we have seen, the text retrieval community has a long tradition extending from the Cranfield collection, of using test collections for running retrieval experiments and TREC is certainly no exception. Recall that test collections consist of a set of documents, a set of queries and a set of relevance judgements for the queries based on documents within the test collection. TREC provides participants with:

- a set of documents which in recent years (to support experiments into ad-hoc and web retrieval) has been one of a 250,000 document collection, a 1.69 million document collection (both for the small web task) or an 18.5 million document collection (for the large web task, in which we did not participate). An example document from TREC-9 is shown below:

```
<DOC>
<DOCNO>WTX001-B02-100</DOCNO>
<DOCOLDNO>IA001-000000-B028-33</DOCOLDNO>
<DOCHDR>http://www.cdnemb-washdc.org:80/relat2-e.html
206.116.210.186 19970101014531 text/html 974
HTTP/1.0 200 OK
Date: Wed, 01 Jan 1997 01:40:31 GMT
Server: Apache/1.1.1
Content-type: text/html
Content-length: 804
Last-modified: Thu, 25 Jul 1996 02:12:19 GMT
</DOCHDR>

<html>
<head><title>Canada-U.S Trade Flows</title>
<!--This document was created on May 1, 1995 by Paul A.
Canniff, Canniff and Company on behalf of the Canadian
Embassy, Washington, DC-->
```

[23] DARPA - Defense Advanced Research Projects Agency.

[24] NIST – National Institute for Standards and Technology.

```
</head>
<body><h2>Canada is the United States' Best Export
Market.</h2>
<p>In 1993 Canada bought &#36;128.1 billion worth of
American merchandise and non-merchandise, 60 percent more
than Japan bought and twice as much as the United Kingdom
bought. In fact, the U.S. exported more to the province of
Ontario than it did to Japan. Canada accounted for 17
percent of U.S. exports to the entire world.
<p><center> <h3>U.S. Merchandise and Non-Merchandise
Exports<br> To Leading Trading Partners, 1993, in billions
of US$</h3> <p><img src="graph2.gif"> </center>
</body>
</html>
</DOC>
```

- a set of queries (called topics in TREC), each of which consists of an identifier (number), a title which for TREC-9 was a real-life WWW query (spelling irregularities included) which had been submitted to a search engine and extracted from the query log, a description of the topic requirements and a narrative section which helps to remove any ambiguities. An example of a TREC topic is shown below. A distinction is made in TREC between manual and automatic query generation methods, with manual representing any human involvement in the query generation process. Our runs were primarily based on manually generated queries.

```
<num> Number: 451
<title> What is a Bengals cat?

<desc> Description:
Provide information on the Bengal cat breed.

<narr> Narrative:
Item should include any information on the Bengal cat breed,
including description, origin, characteristics, breeding
program, names of breeders and catteries carrying bengals.
References which discuss bengal clubs only are not relevant.
Discussions of bengal tigers are not relevant.
```

- a set of relevance judgements which are made available after the participants have run experiments on the document collection. The TREC relevance judgements are usually binary

relevance judgements (a document is either relevant or it is not). Like many others, we feel that binary relevance judgements may not be the best approach for web documents. Apart from the fact that relevance is inherently subjective, web documents, although not themselves providing solutions to an information need, may link into highly relevant documents. The use of binary relevance judgements in this case cannot capture this subtlety though we accept that TREC does have budget and logistic limitations.

TREC organisers employ a pooling technique when generating their relevance judgements, where, for each topic, all participating groups submit their top 1,000 most highly scored documents (or less if 1,000 are not ranked) for each algorithm that they are evaluating. Not all submissions will be accepted for inclusion into the pool. Resources will only allow for a number of sets of ranked documents (called runs) to be pooled and these are known as official runs, with un-pooled runs being classified as unofficial. The top 100 documents from each official run are added to the pool of 'candidate' relevant documents. These documents are judged by an assessor (the topic author) using a binary relevance judgement scale (not relevant or relevant), or in the case of the Web Track of TREC-9, a three way relevance judgement scale (not relevant, relevant and highly relevant) and the relevant documents from these judgements are used as the list of relevant documents in evaluating the quality of the results obtained by different runs. For the official results of the Web track of TREC-9, both relevant and very-relevant documents were combined together to allow for binary relevance judgements. It is expected that the three way judgements will be used in the future to develop other evaluation schemes for web retrieval, i.e. not just evaluation measures based on precision and recall.. In addition for TREC-9 the single best document for each topic was chosen by the assessor from the pool for each query that should be ranked first in any runs. This would allow for evaluation of single document retrieval strategies, an example of which would be Google's "I'm Feeling Lucky" search feature [GOOGLE, 02], which takes the searcher directly to the first web page Google returned for your query.

3.2.1 The Goals of TREC

TREC exists to support and foster research into information retrieval related issues. The overriding spirit of TREC is sharing of knowledge, the knowledge gained from the experiments in the hope of further advancing the field of information retrieval. The declared goals [Voorhees & Harman, 01] of TREC are:

- To encourage information retrieval research based on large-scale collections.

- To increase communication between industry, academia and government by creating an open forum for the exchange of research ideas.

- To speed the transfer of knowledge from research labs into commercial products by demonstrating substantial improvements in retrieval methodologies on real world problems.

- To increase the availability of appropriate evaluation techniques for use by industry and academia, including the development of new evaluation techniques more appropriate to current systems.

- And from [Voorhees_2, 01], to create a series of test collections covering different aspects of information retrieval.

TREC supports experiments on a number of different information retrieval problems. These problems are represented in a number of separate 'tracks'. The tracks represented in TREC 2001 (the most recent) are:

- Filtering Track - for each document in a document stream, decide whether to retrieve it in response to a standing query.
- Cross-Language Track – ad-hoc search task for documents written in one language and queries in another.
- Interactive Track – task to accomplish search in an interactive environment using publicly accessible tools and the web.
- Question Answering Track - task to encourage research into systems that return answers, rather than ranked lists of documents, which is a strict sense is real information retrieval as opposed to document retrieval.
- Video Track – task to promote content-based retrieval from digital video data.
- Web Track – task to investigate information retrieval using web documents.

There have been a number of other tracks run during the lifetime of TREC, the most notable of which was the ad-hoc track, which in TREC-9 was replaced by the Web track. The ad-hoc track was

based on using conventional text documents, and in later years web documents, to evaluate non-WWW specific search tasks.

3.3 The TREC Web Track (1999)

The Web Track was first run in TREC-8 in 1999 [Voorhees & Harman, 99]. Within this track, there were two sub-tracks, the small web task and the large web task. We will only discuss our experiments with respect to the small web task. There were a number of reasons why the TREC organisers decided to run a Web Track (small in particular) in TREC 8. From [Hawking et al., 99] we know that the TREC organisers and participants were interested in evaluating whether:

- the best methods in the TREC Ad Hoc task also work best on the WT2g collection (see later) of web data.
- link information in web data can be used to obtain more effective search rankings than can be obtained using page content alone.

Consequently, the TREC Web track was introduced at the TREC-8 conference to foster research into retrieval of web data as web-based retrieval techniques had heretofore not been examined at TREC. It was feared that the techniques implemented by the Search Engines on the Web were far more advanced that what TREC had to offer. In total, seventeen participating groups submitted a total of 44 runs, 24 of which were content-only and 20 runs utilised linkage data.

Over the course of the Web track (until 2002) there have been two test collections employed by the TREC organisers (WT2g and a larger WT10g). Both test collections have been extracted from a 100GB VLC[25] collection which itself was extracted from a 300GB subset of an Internet Archive crawl made in 1997. The VLC test collection was used as the dataset for TREC-8's and TREC-9's large web task, which we did not participate in. There have been no studies of the degree of overlap between WT10g and WT2g so the overlap shown in Figure 3.3 is purely illustrative.

[25] VLC stands for Very Large Collection as the total disk space requirement for this dataset was 100GB.

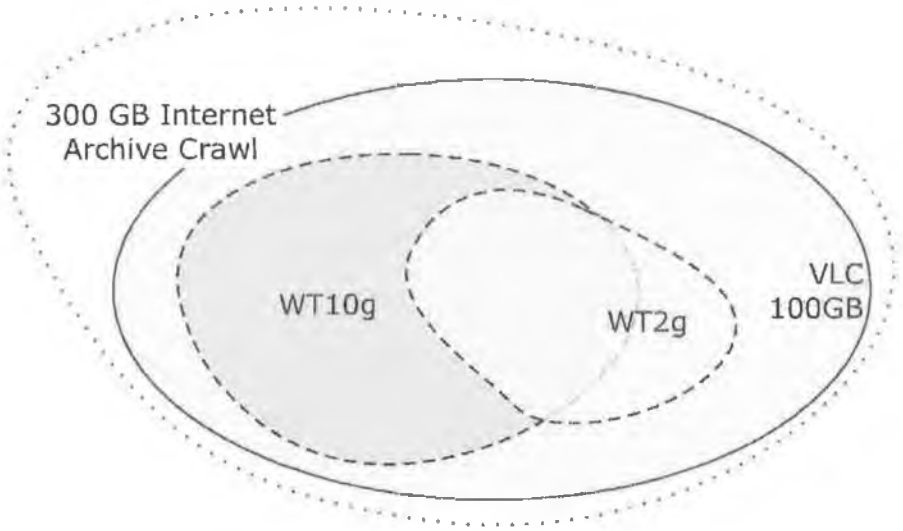


Figure 3.3 : TREC WT2g & WT10g Test Collections

3.3.1 THE WT2G TEST COLLECTION

The Small Task of TREC-8 employed a text collection that consisted of 247,491 documents, requiring 2GB of disk space, which was called the WT2g[26] collection. From [Hawking et al., 99] we know that the WT2g collection was generated with the following four goals in mind:

- To have a collection of comparable size to the TREC Ad-Hoc collection (the usual text IR collection) which had been used in previous years.
- To have a collection which is likely to contain a reasonable quantity of documents relevant to TREC-8 ad-hoc topics.
- To include naturally defined sub-collections.
- To have a collection containing an interesting quantity of closed hyperlinks (having both the source and target of the link, within the dataset.

Along with the WT2g collection a set of topics (numbered 401 – 450) were distributed and after the official runs of participants were submitted, a set of relevance judgements was generated (by

[26] WT2g refers to the fact that the collection was for the Web Track (WT) and was a size of 2GB.

NIST assessors) and made available. In addition to the provision of the documents, the TREC organisers (after an aborted attempt to provide online connectivity data because of access latency issues) distributed connectivity data for the collection. The success or otherwise of any experiments into linkage analysis is dependent on the density of the links within the dataset and the representativeness of these links. A brief examination of the construction methodology employed by the web track organisers when constructing WT2g illustrates some possible problems with the dataset. Recall that the dataset was extracted from the 100GB VLC collection, which was itself extracted from the 300GB Internet Archive[27] [INTERNET ARCHIVE, 02] crawl. This means that all the links within the WT2g dataset could only have originated from within the 100GB collection, which limits the number of links available. Add to that the fact that only links among the 247,491 documents themselves could be included limits the available links further. In Figure 3.4, the links would be any links between documents in sites 1,2 and 3 as well as any links (a, b and c) between documents on these websites to the exclusion of the links represented by the dashed lines.

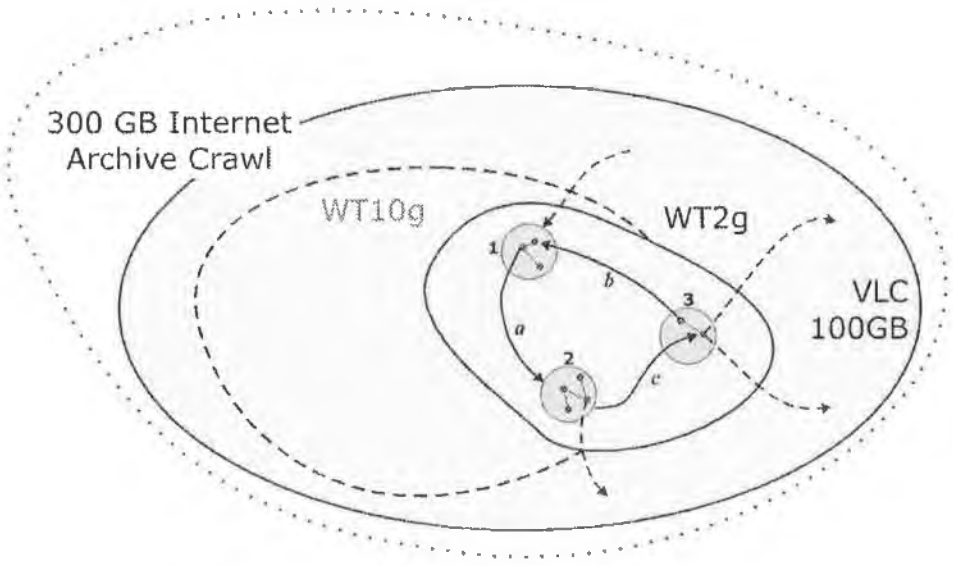


Figure 3.4 : The Construction of the WT2g text collection

Upon examination, it was found that the number of off-site links within WT2g is 2797 out of 1,166,702 or 0.24% [Hawking et al., 99], which we found to be insufficient to support linkage-

[27] The Internet Archive is a website dedicated to crawling and making publicly available snapshots of the web for historical purposes.

based web retrieval [Gurrin & Smeaton, 99]. Our findings concurred with the findings of all other participating groups.

3.3.2 Experiment Overview

TREC-8 gave us our first opportunity to experiment with applying linkage analysis algorithms to real-world web data [Gurrin & Smeaton, 99]. There were a number of well known algorithms such as PageRank and Kleinberg which we could have evaluated, but we felt that it would be more beneficial for us to engage in our own basic research into the area, and to evaluate some simple algorithms all based around citation ranking.

Our experiments were based on reranking a set of highly scored documents by applying one of a number of linkage analysis algorithms, so in effect a two-phase process was involved. The first phase (1) was to generate the set of highly scored documents and the second phase (2) was to rerank these documents based on their linkage structure. See Figure 3.5 for an illustration of the phases involved.

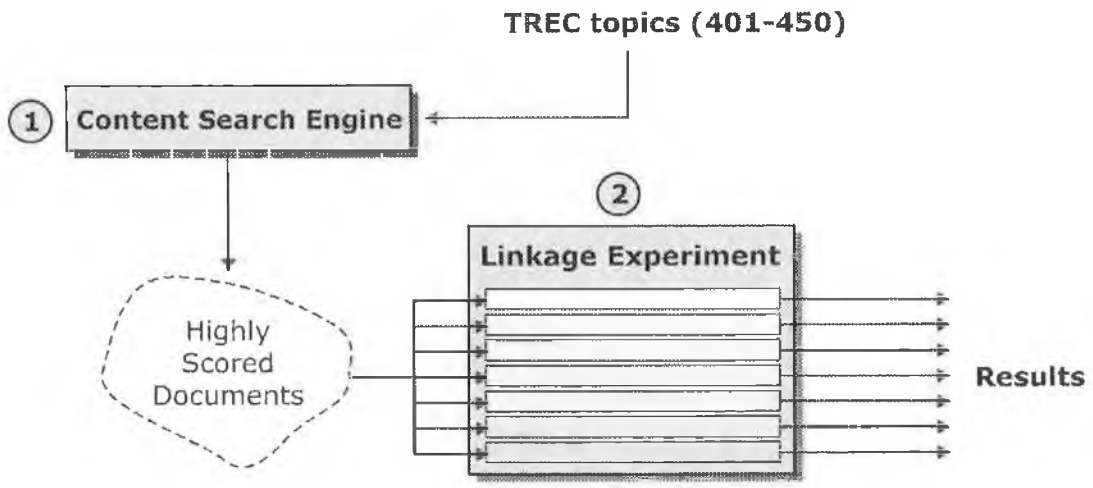


Figure 3.5 : TREC-8 Experiment Phases

3.3.2.1 System Architecture

The system architecture we employed for our experiments can be divided into three logical sections:

- A conventional search engine (the content search engine shown in Figure 3.5) to process queries and return ranked sets of highly scored documents. To generate the ranked sets of highly scored documents for this experiment, we used an off-the-shelf search engine as opposed to developing our own.

- A Connectivity Server to store and retrieve linkage information for each document in the collection.
- Software (the query processor in Figure 3.6) to process the ranked output of the conventional search engine by applying a number of linkage analysis algorithms with the aim of improving retrieval performance.

Figure 3.6 illustrates the architecture of our TREC-8 experimental software that we developed in order to execute the experiments that we now describe.

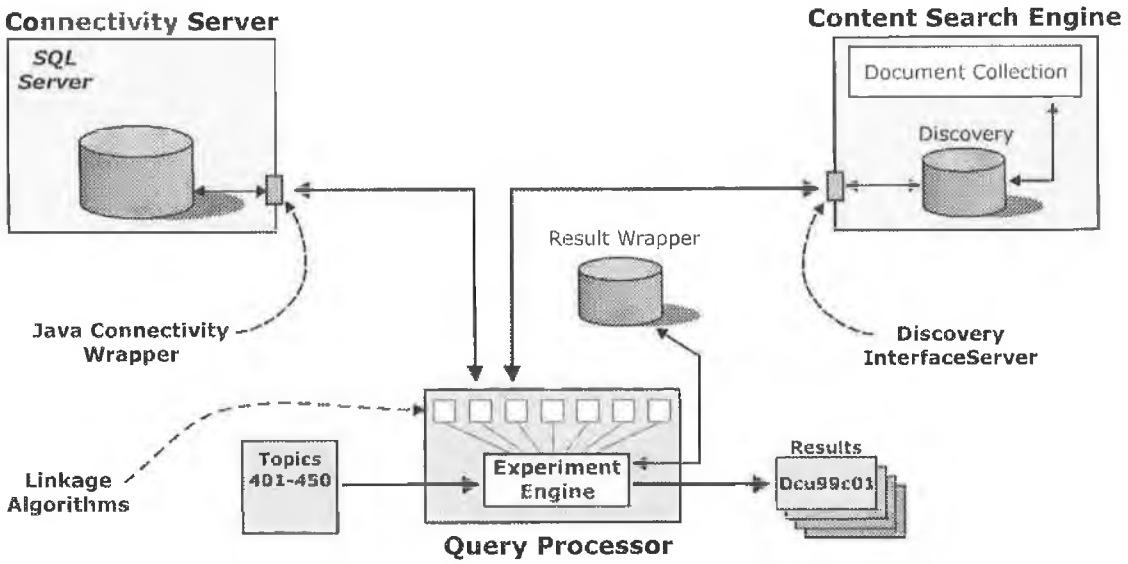


Figure 3.6 : Architecture of our TREC-8 Experimental System

In order to allow testing of the retrieval performance of our algorithms we extracted and stored a small amount of information from each document, which was used as part of an interface to our system, to provide a description of each document. The information we extracted consisted of:

- Document ID
- Document Title
- Document Text (256 bytes of text from the start of the document, to the nearest word)

In this way we could provide content-only results for each query, which were reranked in the second phase of the experiments using some linkage analysis algorithms.

Once we were able to get content-only results for a query, we then needed access to connectivity information for each document. We utilised Microsoft SQL server 6.5 [SQL SERVER] for this task. On average we found that the server could handle about 100 queries per second, which although slow, proved adequate. We found that increasing the specification of the hardware from 100MHz processors to 550MHz and the software to version 7 resulted in a tenfold improvement in the speed of processing queries to the Connectivity Server. These improvements were implemented for subsequent experiments after TREC-8. The structure of the connectivity servers that we used during our TREC experiments were based on our findings as we were developing the software and the connectivity server that we use is based on storing the following data about each link:

- Type (off-site or on-site)
- URLs (source and target)
- Web Site Identifiers (associated with source and target URLs)

To improve server performance we generated non-clustered indexes on both the source and target URL rows. All software was implemented in JAVA under Windows NT. For situations that require much interaction with a connectivity server (e.g. multi-iteration techniques such as Pagerank or SiteRank) we usually used a software connectivity server that we developed which stored connectivity information in RAM using JAVA structures like ArrayLists and Vectors to provide fast retrieval performance.

3.3.2.2. CONTENT EXPERIMENTS

As we have mentioned, each search firstly consisted of a content analysis stage performed on the test collection. This immediately posed problems in that we had not yet developed our own search engine and therefore we had to use off-the-shelf software and we choose AltaVista Discovery[28]. Discovery usually indexes Word, Excel, e-mail and other common file types, but also is capable of indexing HTML files stored locally on disk. Consequently it suited our needs for a basic search engine that was capable of accepting queries and returning sets of highly scored documents for additional processing.

[28] AltaVista Discovery was a desktop content-only search and retrieval application provided by AltaVista and was made freely downloadable from their web site. It was developed to provide search and retrieval facilities over email messages and PC application files on a desktop PC, laptop or shared file servers.

The queries (in a batch process) are passed to the Discovery Server (an application we developed to provide an interface between Discovery and our software, running on a computer that is running Discovery), which in turn, sends them in a HTTP request to Discovery, retrieves Discovery's result, strips any unnecessary HTML data and passes the content-only results back to our experimental software which is then utilised in the linkage experiments described below. However, using Discovery did pose a number of problems:

- Discovery would only list the top 200 documents in response to a query. We found no way around this limitation and consequently our results only ever contained a maximum of 200 documents, even when more were could have been relevant. Recall that TREC accepts the top 1000 documents in response to a query.
- Discovery did not provide scores for each ranked document, so the scores had to be simulated using the following formula. This score provided us with a content-only score for each document based on each document's rank within the 200-document result set. Assuming N is the total number of documents in the result-set and R_n is the ranked position of document$_n$ the formula to generate the score Sc_n for each document in the 'relevant-set' is as follows:

$$Sc_n = \sqrt{\frac{N - R_n}{N}} \qquad for(R_{1...N}) \qquad (3.4)$$

It was this that we sent to TREC as the results of our content only runs.

Our preliminary content experiments were based on using the title of the query alone, and upon submission of our results and the release by TREC of the relevance judgements we experimented with manually generated queries. We found that manually generated queries (generated by a human after examining the TREC topics) resulted in higher precision values so the following experiments are based on manual queries. Figure 3.7 shows the comparison between manual and automatic queries. It is clearly shown that manual queries produce better results than automatic queries where it matters, in the top ranked documents so for future TREC experiments (the following year) we used manually generated queries.

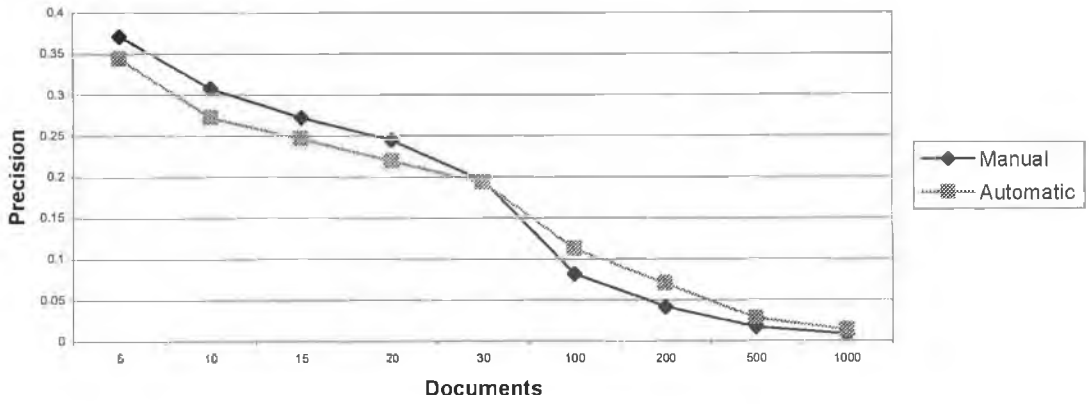


Figure 3.7 : Comparison between automatically and manually generated queries

3.3.2.3 LINKAGE EXPERIMENTS

From the content-only experiment described above, we are provided with a baseline set of documents, the 'Result-Set', which can be expected to consist of highly scored documents, many of which will be relevant to the query. We developed seven re-ranking schemes based on linkage metrics to re-rank this set of retrieved documents based on the indegree and outdegree of the pages. Each re-ranking scheme generates a final score for each document ($Sc^1_n \ldots Sc^7_n$). Consequently, were a number of (or all) ranked documents returned by Discovery to have identical linkage metrics then the ranked list of documents returned by Discovery would dictate final document ranking.

Recall that in linkage analysis we generally assume that the more popular a document is, the more in-links that document will have from the WWW. Let n be some web page and S_n be the set of pages that link into document n we can represent this as follows:

$$Sc^1_n = |S_n| \qquad (3.5)$$

In this case Sc^1_n is based purely on the indegree of document$_n$. We ran experiments using this simple formula for evaluation (called *link1*). We also evaluated (as Sc^2_n) the notion of limiting the indegree value to be influenced by a maximum of 50 in-links (called *link2*). This we felt would avoid any possible situation where documents with an excessively high indegree could hijack the ranking process. Note that we did not distinguish between link types in these two experiments and that the

content-score from phase 1 was only used to distinguish between documents with similar linkage scores.

Recall that we view links as being one of three types, self links, off-site links and on-site links, and that we normally ignore self-links and on-site links as they are not considered to be judgement bearing. As an experiment, we wanted to evaluate what effect on-site and off-site link types will have on the ranking so we developed another two metrics (called *link3* and *link4* respectively) based on ranking by just off-site links and just on-site links. Letting T_n be the off-site in-set of n and U_n be the on-site in-set of n, we have the following formulae which calculate two new scores (Sc^3_n and Sc^4_n) for each document:

$$Sc^3_n = |T_n| \qquad (3.6) \qquad\qquad Sc^4_n = |U_n| \qquad (3.7)$$

It is notable that a system that implements only one iteration of the basic Kleinberg's algorithm (authority list) is similar to a system that ranks pages based purely on off-site indegree.

Thus far the metrics were simple and were based on nothing more than basic citation counting, but we thought that examining the potential of a document to act as a source of links to further information (a *hub* document) may also aid the re-ranking process. Recall the first of two assumptions necessary for linkage analysis, that a link between two documents on the web carries the implication of related content, thus a document with a higher outdegree should contain links to more relevant content than a document with a smaller outdegree. Accordingly we developed another metric that utilised the outdegree (both off-site and on-site) of a document in addition to the indegree as part of the ranking process. Given that the outdegree of a document does not carry with it any indication of the authoritiveness of a document we decided to limit the outdegree to a maximum of 20. This was in order to avoid some large hub type documents swamping the results. In addition we felt that the indegree would be a far more useful measure so we weighted the indegree score to be four times[29] more influential than the outdegree score, giving the following formula where α was 1.0 and δ was set to 0.25 with Sin_n and $Sout_n$ being the set of in-links and out-links from document n respectively, which gave us our 5th score for each document (Sc^5_n):

$$Sc^5_n = (\alpha \times |Sin_n|) + (\delta \times |Sout_n|) \qquad (3.8)$$

[29] The figure was chosen as a best parameter value that was attained by examining the results of a number of sample queries prior to running the actual experiments. Most of our parameter values were generated in this manner.

Our parameter figures (for all TREC-8 and TREC-9 experiments) were selected by observing the results attained by a number of queries (not TREC supplied queries) and tuning the parameters accordingly. We will refer to this experiment as *link5*. Thus far, all our experiments had been based on simply reranking the set of highly scored documents returned from the content-only experiment and we had totally ignored the content scores which had been given to the documents save when two documents had the same linkage scores so we evaluated the usefulness of incorporating these scores into the linkage-based ranking algorithm in a more concrete manner. Building on the previous experiment, we expanded it to incorporate the content score generated from Discovery (Sc^6_n). Letting $DiscSc_n$ be the score applied to each document in the content experiment using the formula described earlier and α, δ and λ be constants for regulation of influence:

$$Sc^6_n = \left(\alpha \times |Sin_n|\right) + \left(\delta \times |Sout_n|\right) + \left(\lambda \times DiscSc_n\right) \qquad (3.9)$$

We varied the score of α from 2.0 and 4.0, but our test results illustrated that 4.0 was best, similarity δ was set to 2.0 and λ was set to 1.0. This experiment immediately flagged the problem of how precisely do we combine the linkage scores and the content score. In this experiment, we applied best guess parameters, which cannot be sufficient in all cases. An alternative approach was needed and one was proposed by AT&T's experiments in TREC for the subsequent year, which we shall examine, along with our own approach, later in this thesis.

However, simply using this formula did not complete our experiment. As we discussed earlier, it is intuitive that we view off-site links as being judgement bearing while on-site links are mainly used for internal website navigation purposes and do not carry any weight of human judgement. Desirable as the approach to only include off-site links may seem, we found that the lack of off-site links within the WT2g dataset prohibited us from fully implementing this approach. To overcome this problem we decided to allow on-site links to exert some influence on the final score of each document, our assumption being that a document with a higher on-site in-degree will be a more important page within a website. In reality, were we to take this approach with live web data, we would only serve to increase the ranking of documents from large, well inter-connected (within their domain), web sites. Consequently we weighted off-site in-links at four times (best guess) the weight of on-site in-links and once again limited the total link score for each document. This replaced our subgraph with a weighted subgraph with a select few edges (off-site) having a weight of four times that of the rest. The following formula was used to calculate the in-link score (Sin_n) of a document n,

replacing the simple $|Sin_n|$ scoring of the previous formula, letting T_n represent the off-site indegree of n and U_n represent the on-site indegree of n with α and δ being constants used for tuning:

$$Sin_n = (\alpha \times |T_n|) + (\delta \times |U_n|) \qquad (3.10)$$

The results of this algorithm were submitted as one of our official runs to TREC and will be referred to as *link6*.

The seventh and final algorithm (*link7*) we evaluated as part of our TREC-8 experiments was an attempt to overcome the lack of off-site links in the dataset. Our approach was to utilise the linkage information for each document from the WWW, as opposed to using the linkage provided with the dataset. Recall the method used to generate WT2g would restrict the number of off-site links that could be included in the dataset. The problems of using live WWW connectivity data on WT2g became apparent immediately:

- many of the documents were no longer in existence as the documents from which the dataset had been generated had been gathered by the Internet Archive in early 1997.
- many may not have been indexed by the linkage source we were querying.
- a number would have had their content modified since they had originally been gathered by a web crawler and thus the current WWW connectivity information would be considered invalid.

In order to get around this problem we ranked each document based on the root URL of the domain, as opposed to the actual URL of the document itself, our belief being that ranking documents on the basis of the quality of their websites may produce some interesting results. This domain root URL was then used to query the AltaVista search engine using the "link:URL" query that returns the number of and actual in-links into the document in question. We then ranked by the popularity of this page (usually index.html) within each domain as opposed to the actual document from WT2g. Letting S_m be the set of documents that link into the root of the domain m, and n be a member of the domain m:

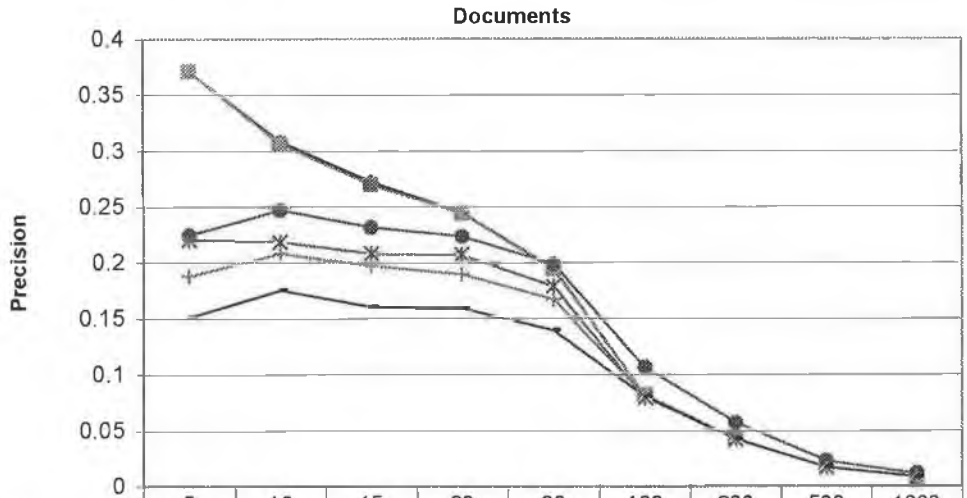
$$Sc^7{}_n = |S_m| \qquad (3.11)$$

This completed our experiments into linkage-based retrieval of WWW documents for TREC-8.

3.3.2.4 RESULTS

We entered three of our experimental approaches as official runs into TREC in August 1999, two of which were linkage-based (*link5* and *link6*) and the third our content-only run based on the results of AltaVista Discovery. Two of our runs were added to the pool for relevance judgement purposes (*content-only* & *link6*). As with other participants, none of our experiments found any improvement in precision when incorporating linkage-based retrieval algorithms in the ranking process in the overwhelming majority of cases, with any fractional improvements coming at lower levels of recall.

Figure 3.8 below shows the precision at a range of document cut-off values returned from running the manually generated queries on WT2g. The labels (link1…link7) for the experiments correspond directly to the seven algorithms just outlined and also correspond to the seven document scores (Sc^1_n… Sc^7_n).



	5	10	15	20	30	100	200	500	1000
Content-only	0.3714	0.3082	0.2721	0.2449	0.1946	0.0816	0.0422	0.0169	0.0084
Link3	0.3714	0.3061	0.2694	0.2439	0.1939	0.0816	0.0422	0.0169	0.0084
Link4	0.2204	0.2204	0.2082	0.2082	0.1789	0.0794	0.0422	0.0169	0.0084
Link2	0.2204	0.2184	0.2082	0.2071	0.1789	0.0794	0.0422	0.0169	0.0084
Link1	0.2204	0.2184	0.2082	0.2071	0.1789	0.0794	0.0422	0.0169	0.0084
Link7	0.2245	0.2469	0.2313	0.2235	0.1986	0.1067	0.0572	0.0229	0.0114
Link5	0.1878	0.2082	0.1973	0.1898	0.1673	0.0806	0.0422	0.0169	0.0084
Link6	0.151	0.1755	0.1605	0.1592	0.1395	0.0806	0.0422	0.0169	0.0084

Figure 3.8 : Results of our experimental runs on the WT2g collection

The lack of off-site links within WT2g is clearly illustrated by our results. In Figure 3.8 we can clearly see that *link3* (off-site indegree, equation 3.6) results are essentially equivalent to the content only results given that the sparsity of off-site links was such that the original ranking of Discovery was rarely changed. Given that in this experiment, the content-only result was only employed to decide on the final ranking for documents when indegree scores were equal, this shows that the off-site indegree ranking had no effect (good or bad) on retrieval performance. On examining the density of off-site links in WT10g, we found that only a tiny percentage of documents would possibly be affected by incorporating these into a ranking formula and as such, the result is not surprising. So although this experiment was the 'most successful' of all linkage experiments we are unable to make any conclusions as to the benefit or otherwise of this experiment.

As expected, we found that re-ranking by on-site indegree (*link4*, equation 3.7) is not effective as this would only serve to rank highest documents from within large, well inter-connected domains. Due to the lack of off-site links within WT2g, we found that the scores for *link1*, *link2* (equation 3.5) and *link4* are almost identical even though *link1* and *link2* do not distinguish between link types, unlike *link4*, which ranks by on-site links only. We did expect that ranking by on-site links would not serve to improve precision and thus these results came as no surprise. All decrease precision of the top 10 ranked documents (the normal search engine result set size) from .3082 to (at best) .2204.

The results from *link7* (equation 3.11) which used live WWW linkage data were noteworthy due to the fact that this was the only approach that increased precision over content-only results, but only from 30 to 1000 documents, which is not what is required for web search. Both *link5* and *link6* also produced disappointing results, even lower than taking by on-site indegree alone, so it seems likely that the integration of an out-degree score did not aid the retrieval process and surprisingly the inclusion of the content-only results into the equation for *link6* (equation 3.9) resulted in a decrease in overall performance over *link5* (equation 3.8). Although it must be noted that the out-degree of a page would have proportionally more inclusive over ranking in *link6* than in *link5*.

In all, our results were disappointing, in that we were unable to come to any concrete conclusions on the merits of incorporating any of our linkage algorithms into a web retrieval system. Initial observations could suggest that our algorithms failed to improve retrieval performance due to the nature of the algorithms, none of which implemented an iterative approach such as PageRank or

Kleinberg. However as discussed below, other participating groups ran alternative algorithms, including iterative algorithms such as PageRank or Kleinberg, also without improving effectiveness.

3.3.2.5 Discussion

In the TREC-8 web track, participating groups took part and those that utilised the link information implemented a variety of approaches including Kleinberg's and PageRank mentioned earlier. To the surprise of many participants, no participating group (save one with insignificant improvements) managed to improve precision by incorporating linkage information into the retrieval process over that obtained by their own conventional content-only searching. Hawking [Hawking, 01] states that "None of the participants in the TREC-8 Small Web Task, using a two gigabyte corpus (WT2g), managed to demonstrate any benefit whatever from using hyperlink methods in that particular retrieval task".

Aside from groups implementing Kleinberg and Pagerank, a number of groups evaluated alternative algorithms, which had either come from the field of citation indexing or were entirely new. Making use of sibling pages was the approach taken by RMIT/CSIRO [Fuller et al., 99]. This was based on the propagation of content-only scores from documents that link into relevant documents. Seoul National University [Shin et al., 99] implemented a similar technique called Score Propagation, neither with any positive effect on retrieval performance. The technique of spreading activation was evaluated by two groups, where the relevance of a document D to the query is computed in a preliminary step and these values are propagated to all linked documents from D through a certain number of cycles using a propagation (limiting) factor γ. The documents are then sorted according to their new score (called the activation). Both the Université de Neuchatêl [Savoy & Picard, 99] and IRIT/SIG [Boughanem et al., 99] implemented this technique. The former using only one cycle, and considering only the top 50 documents, found that all experiments resulted in a decrease in retrieval effectiveness. This finding was repeated by IRIT/SIG who recorded a decrease in average precision when using this technique, although they did only execute this process on the top 12 documents in one run, the second content-link run being executed on the top 40 documents. Both participating groups found no improvement in precision from implementing techniques based on spreading activation.

An examination of the results over all the participating groups shows that the differences between the content-only runs and the linkage-based runs are mostly very small and in the vast

majority of cases negative [Hawking et al., 99]. Any case where a large difference was found, they were all found to be negative.

The overriding belief among participants was that the WT2g collection did not support true investigation into the experiments that had been evaluated. The reasons put forward [Hawking et al., 99] as to why this could be the case were:

- The number of off-site links may have been too small, with only 0.24 % of closed (having both source and target within WT2g) off-site links being contained in the dataset. We believe this to be the most compelling reason for the failure of any linkage-based approach to produce any significant improvements over a standard content-only run in TREC-8. We ourselves have shown that ranking by on-site indegree does not have any beneficial effect and these experiments hold validity as each document (on average) contained 4.7 on-site in-links and thus the presence of links we felt was sufficient to provide an indication of the benefits or otherwise of ranking by on-site indegree.
- The queries used were not suitable to linkage analysis type experiments. This may have a part to play in the disappointing results and we speak about this at greater length when discussing the TREC-9 results.
- The relevance judgements used were not optimal for evaluating linkage analysis experiments. The use of binary relevance judgements on single web pages in TREC-8 prohibited the positive evaluation of web pages, which although not directly relevant themselves, may contain links to highly relevant pages. As we have mentioned earlier, we would recommend a more appropriate performance evaluation methodology than simple precision and recall based on binary relevance judgements as was the case with TREC-8 (and TREC-9). Perhaps a five-point scale for relevance judgments (1. not relevant, 2. links to relevant, 3. relevant, 4. relevant & links to relevant, 5. highly relevant) would be one (of many) option(s) that would have been more beneficial. An alternative option could allow us to rank a document on the basis of its support for continued browsing based on out-links.

The shortcomings of WT2g eventually led to the creation of a new collection, the WT10g collection which was used in TREC-9 and in the tenth TREC conference known as TREC-2001. The prevailing hope among participants was that the shortcomings of WT2g would be rectified in the WT10g dataset.

3.4 THE TREC 9 WEB TRACK (2000)

The ninth TREC conference was held in November, 2000. Seventy groups [Vorhees & Harman, 00] from 17 different countries participated. TREC 9 was the second year of running the web track, although this time, WT2g had been replaced by a 1,691,071-document (10 GB) collection called WT10g. The goals of TREC-9 Web track were twofold:

- To experiment with standard IR techniques to see how effective they are on real-world web data.
- To see if linkage analysis was a useful aid to web search on the new dataset.

3.4.1 THE WT10G DATASET

Like its predecessor, WT10g was a subset of the 100GB VLC2 dataset used in the large web task, which we recall was itself a subset of the 300 GB Internet Archive crawl. WT10g was extracted from the VLC2 in such a way that maximised the number of cross-site links that are contained in a subset of the 100GB VLC2 dataset. When preparing a suitable dataset for TREC-9 the reasons for the failure of the TREC-8 experiments on WT2g were considered. The organizers also wished to facilitate high levels of participation and in order to do so the dataset was tailored to suit requirements:

- A corpus size of 10GB was chosen which represented 1.69 million documents. At the time this could have fitted on a large inexpensive (< $250) IDE hard drive, thus keeping the cost of participation low for many of the groups. Figure 3.9 illustrates the comparative sized of the TREC collections used as part of the Web track.

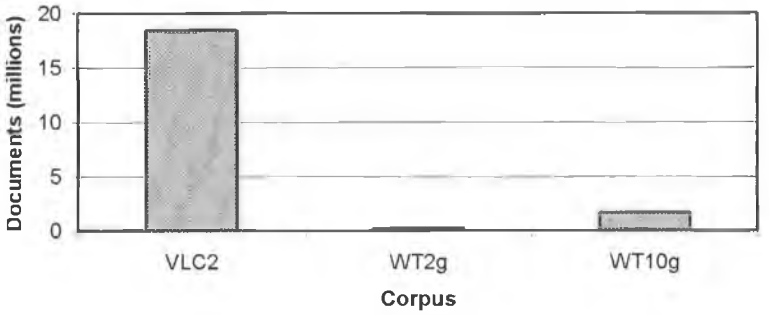


Figure 3.9 : Comparative sizes of the TREC collections

- Non-English data and binary data was not included in this corpus, although Web logs were and these comprised over 600 of the documents.

Since the WT10g collection (like WT2g) was extracted from the 100GB VLC2 collection (18.5 million web documents), this means that all the links within the WT10g collection could only have originated from within the 100GB collection (as the larger 300GB superset was not used in the construction of WT10g) which limits the number of links available. Add to that the fact that only links between the 1,692,096 documents could be included in the test collection, limits the available links even further, which means that very careful selection of the documents that comprise the dataset was needed to maximise the density of off-site links within the test collection. Figure 3.9 illustrates the nature of the links that could be extracted from VLC2.

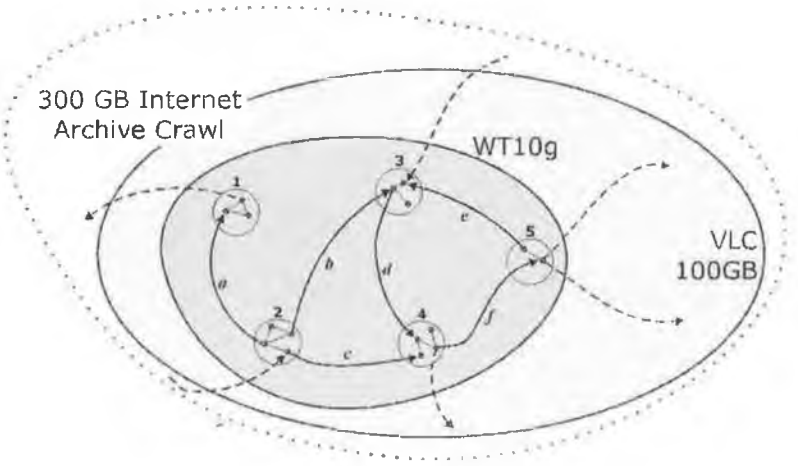


Figure 3.10 : Construction of the WT10g Collection

The links that comprise the WT10g collection would be all the links between documents within websites 1 to 5, including on-site links (not labelled) and off-site links (labelled a to f in Figure 3.10). For those taking part in TREC experiments, any other links were not available, unless a group had access to the VLC2 collection (the super set of documents) and this method of increasing the number of available links was not available to most of TREC's participating groups.

In addition, a number of other constraints [Bailey et al., 01] were placed on the selection procedure for documents, such as the fact that documents must be selected from servers to maintain an average of the number of documents chosen from each server and that servers were chosen that contained documents with homepages. Table 3.1 illustrates WT10g properties.

Quantity	**Value**
Documents	1,692,096
Servers	11,680
Average documents per server	144
Off-site links (within WT10g)	171,740
Servers with off-site in-links	9988
Servers with off-site out-links	8999
Documents with out-links	1,295,841
Documents with in-links	1,532,012
Servers without a homepage	0

Table 3.1: Properties of the WT10g test collection

3.4.2 TREC-9 Experiment Overview

Our experiments for TREC-9 were a continuation of our experiments for TREC-8, yet this time we had more experience of working with link data and were in a position to develop more advanced algorithms. Our experiments were, once again, based on our own algorithms [Gurrin & Smeaton, 00]. Each algorithm implemented a different aspect of linkage analysis (citation ranking, spreading activation and co-citation analysis) and we hoped that the new WT10g collection would provide the foundation for the successful evaluation of our algorithms. In addition, we also hoped that the results would illustrate to us which techniques would have offered the best prospects for improving retrieval performance so that we could have concentrated our research on those techniques.

As was the case with many of the participants, our experiments were based on reranking a set of highly scored documents by applying one of a number of linkage analysis algorithms, so once again, a two phase process was involved. The first phase was to generate the set of highly scored documents and the second phase was to rerank these documents based on their linkage structure.

3.4.2.1 System Architecture

In a fashion similar to our TREC-8 experiments, we used an 'off the shelf' search engine to generate content-only results for each query. Based on our experiences with AltaVista Discovery and specifically a problem we found with the limited number of high-scored documents retrieved and the less than impressive indexing speed, we felt it best to use a different application, so for TREC-9 we used Microsoft Index Server [INDEX SERVER] for this purpose. For subsequent experiments (outlined in Chapter 4) we developed our own search tools. This utilisation of Index Server did require the conversion of the WT10g collection into an artificial website, to allow Index Server's crawler to traverse all web pages from a given root page. During our experiments, one workstation was dedicated to Index Server and providing content results.

The linkage data was stored in a Microsoft SQL Server 7 [SQL SERVER] database (an upgrade from 6.5) running on another PIII workstation, which suited our requirements for a Connectivity Server. This allowed us to support over 1,000 queries per second. We used a third PIII workstation to process the queries, and calculate the linkage scores for each document and generate the results. All necessary code was written in JAVA (version 1.2) for Windows NT 4. We networked the computers together using a dedicated 100Mbit/s switch. An overview of the software set-up used is shown below:

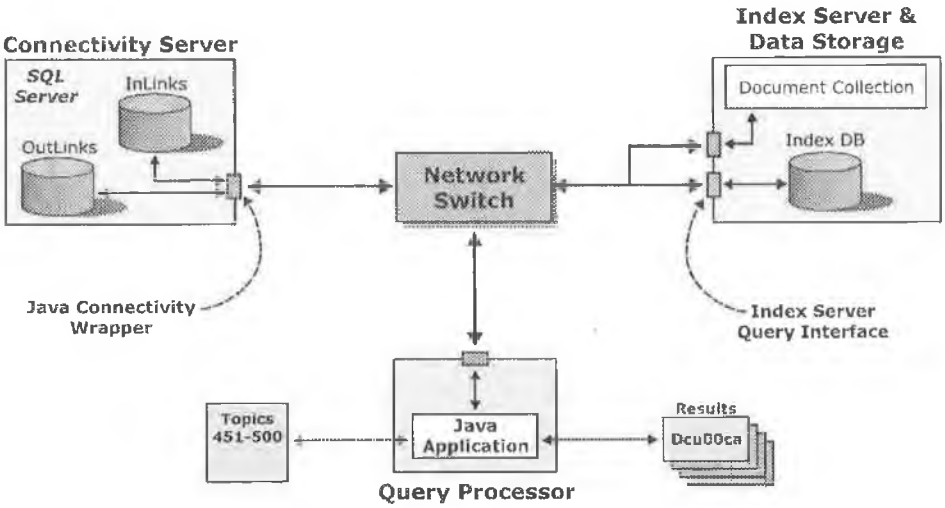


Figure 3.11 : Our TREC-9 System Architecture

Our TREC-9 experiments were devised so that we could evaluate specifically non-iterative approaches to linkage analysis. We submitted four runs for evaluation purposes, one content only run, as provided by Index Server and three linkage-based runs. This content run was executed before any of the linkage-based runs were executed as the basic output of Index Server was used as input into the three linkage runs.

3.4.2.2 CONTENT EXPERIMENT

Recall that our TREC-8 experiments required a two-phase process, and TREC-9 was no different to this with a first stage generating content-only results followed by a second reranking phase. The content-only stage involved sending the query to Index Server and extracting the results. The queries we sent were manually generated from the TREC topics. In order to do this we got a small number of postgraduate research students to generate the queries from the title, description and narrative of the topic.

Once the queries had been generated and sent to Index Server the top 2,000 result documents[30] were retrieved using the Vector Space query model (Index Server supported retrieval using a number of query models). These 2,000 documents were ranked by Index Server according to their degree of similarity to the query, but they were not scored, so, once again, we had to generate

[30] In a number of cases, less than 2,000 documents that number were scored by Index Server so a smaller result set was retrived.

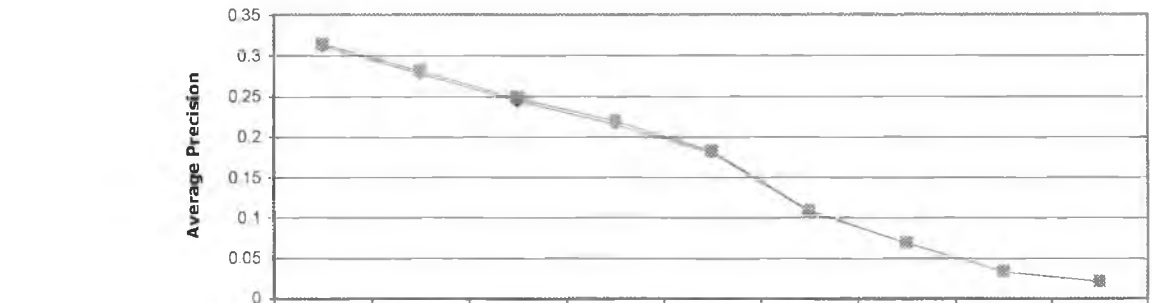
our own scores. Assuming N is the total number of documents in the result-set and R is the rank of that document in the result-set the formula to generate the score Sc_n for each document (same as TREC-8) is as follows:

$$Sc_n = \sqrt{\frac{N - R_n}{N}} \qquad for(R_{1...N}) \qquad (3.12)$$

We refer to this ranked set of documents as the 'relevant-set'. The top 1,000 results (where available) were extracted for each query and submitted as our single content-only run.

After the publication of the relevance judgements we ran two experiments; one to evaluate if using automatic (title only) queries would negatively affect retrieval performance and a second to evaluate our method of generating the result-set.

Our evaluation of our query generation method required executing a content-only run using automatic queries and comparing these results to the manual queries that formed the basis of our result set. To our surprise, we discovered that had we implemented an automatic run using just the titles only that we would actually have marginally improved precision as shown in Figure 3.12.



	5	10	15	20	30	100	200	500	1000
manual	0.312	0.278	0.244	0.216	0.18	0.1076	0.0686	0.033	0.0207
automatic (title only)	0.3143	0.2816	0.2476	0.2194	0.1823	0.1088	0.0687	0.0331	0.0209

Figure 3.12 : Comparing manual and automatic queries.

Obviously this did not affect the overall findings of the experiments which are discussed below. The second experiment that we ran after the results were published was to evaluate if (simply) choosing the top ranked 1,000 documents was more useful than the approach taken by Kleinberg to

build his base-set. If we look at Kleinberg's algorithm, a content-only set of documents of size 200 is generated and this is expanded to include near-neighbour documents, for the reason of increasing link density in the expanded set, albeit at the expense of 'topic drift' problems. However, this root-set expansion phase of Kleinberg's seems to lead to topic-drift problems [Bharat & Henzinger, 98], where the documents that are ranked highest are often generalisations of the topic represented by the query. Consequently, we ran an experiment to identify if simply selecting the top 1,000 documents (content-only result) resulted in more relevant documents being found than would implementing the root-set expansion phase of Kleinberg's technique. We found this to be the case. Kleinberg's approach resulted in an average loss of 5.68 relevant documents per query. See Figure 3.13 for the total recall figures over all queries based on the set generation process implemented, where 'base 200' is the top 200 documents from the content experiment, 'base 1000' is the top 1,000 documents and 'expanded-set' is based on Kleinberg's technique.

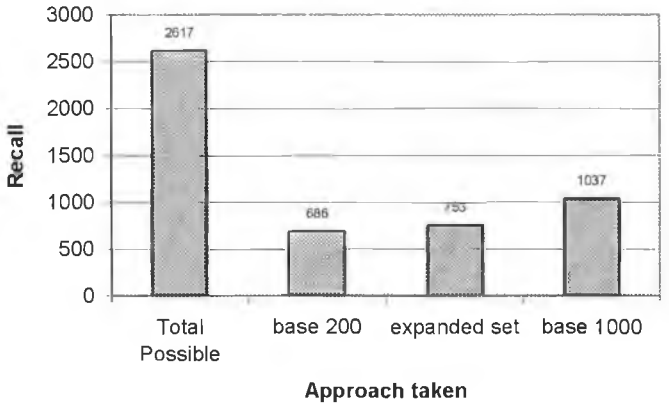


Figure 3.13 : Comparing Different Approaches to Relevant-Set Generation

However, this improvement in recall does have its drawbacks. Expanding the root-set along the links does produce an expanded-set that contains a high number of interconnected documents but selecting the top 1000 (or 2000) documents produces a set of documents having a much sparser set of interconnections. Experimental ranking techniques such as Spreading Activation are based on the notion that the result-set will firstly contain relevant documents, and secondly that these relevant documents will be linked together. It may be the case that using the expanded-set is better for linkage analysis because it will have a denser set of links between the documents. However, based on the results of other participants who implemented Kleinberg's algorithm this was not shown to be the

case and none of the participants who implemented Kleinberg's algorithm were able to produce any increase in retrieval performance.

3.4.2.3 LINKAGE EXPERIMENTS

Our three linkage experiments (generating three scores $Sc^1_n \ldots Sc^3_n$ for each document) were all executed at query time, unlike PageRank, but similar to Kleinberg, and were based on re-ranking the relevant-set of documents which were generated during the content stage outlined above, but only requiring one iteration. When we employ a query-time technique, such as Kleinberg's algorithm, we must be careful that the algorithm does not require so much processing as to make it prohibitively slow to use in the real-world (the second principle of performance). PageRank, on the other hand, requires a similar iterative process (on a vastly larger set of documents), but this is only done once per index update as opposed to once per query. The algorithms we developed for TREC-9, although processed at query-time, do not have an iterative process involved, which supports adherence to our principles of performance.

Citation Ranking

Our first linkage based experiment (*dcu00la*) was a modification to basic citation ranking. There are two basic methods of combining content analysis with linkage analysis [Bharat et al., 98], both assuming that we can determine the relevance of a document to the query topic. We can:

- Eliminate non-relevant documents from the linkage graph, or
- Regulate the influence of a document based on its relevance to the query topic.

In this experiment, we employed a form of citation ranking to rank documents (based on off-site indegree). However, following the first of the basic methods of combining content analysis with linkage analysis (eliminating non-relevant documents), the indegree of a document was now a 'qualified' indegree[31], in which we were only interested in the content-scored documents that linked into the document in question. If we implemented pure citation ranking we were allowing documents that are not scored to influence the ranking process, but in this way, only documents that are scored (in the content-only phase) can have influence over another document's ranking. In Figure 3.14 the

[31] Qualified indegree means that the indegree of a document was not the document's true indegree, rather an indegree based on the number of relevant documents that link into the document in question.

qualified indegree of document A is based on which documents were scored (have a score above 0.0 in Figure 3.14) which are documents C and D, thus giving a qualified indegree of 2 for document A.

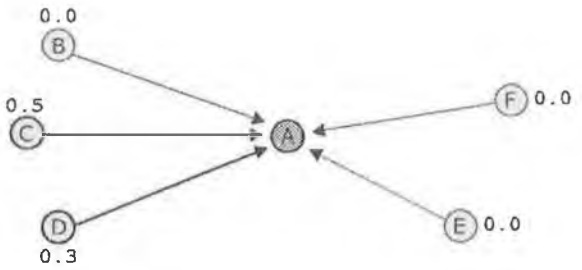


Figure 3.14 : Illustrating a qualified indegree for a document

In order to normalize for wide variations in document indegrees, we applied a logarithmic function to the qualified off-site indegree (+1) and this value was then multiplied by the original content score of the document, so that we did not loose the original content-only score of a document. This score is added to the original content-only score of the document to produce a new score for the document.

This use of a logarithmic function is similar to one of the techniques of normalizing *tf* scores for variations in document length that was outlined in Chapter 1. We generated Sc'_n as the new score for document n and ranked the documents by this score. Letting M be the set of scored documents from the off-site in-set of document n and let Sc_n be the content-only score for document n generated in the previous phase we have:

$$Sc'_n = Sc_n + \left(Sc_n \times Log\left(|M| + 1\right)\right) \qquad (3.13)$$

This was calculated for the top 30 documents. During development of the software for the official runs, we had experimented with a variety of cut-offs for the top-ranked documents and kept the value of 30 as we found that all values we tested seemed to affect ranking performance for the worse, but setting the cut-off to 30 helped limit the effect of the problem, although we did aim to keep this cut-off point as high as possible. This was submitted to TREC as an experimental run for evaluation.

Spreading Activation

Our second linkage-based algorithm (*dcu00lc*) employed the second of the basic approaches to combining content and linkage analysis (regulating a document's influence based on its content-only score) and is based on the concept of spreading activation. Spreading activation refers to a technique that propagates numerical values (or activation levels) among the connected nodes of a graph. In the context of our experiments it facilitates a document transferring its score across its out-links.

For each document in the result-set we identify what documents that comprise the off-site in-set for the document are actually part of the result-set. Each of the documents we have identified from the off-site in-set (of document A in Figure 3.15) should then have an associated content-only score (such as documents B and C). It is the content-only score that is divided equally among a document's out-links (similar to PageRank), thus applying a weighting to each link. In Figure 3.15 the content-only score of document B is spread evenly among its two out-links and a score of 0.1 is passed to document A (dashed line). In a similar manner, the content only score of document C is spread among four out-links and a score of 0.125 is passed onto Document C.

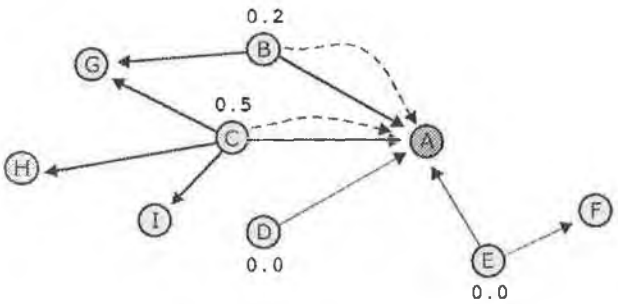


Figure 3.15 : Illustrating our Spreading Activation technique

This query-time process enables us to provide a final ranking for documents. So in effect, if a document has in-links from a number of relevant documents then its score is increased by an amount proportional to:

- its own relevance score
- the relevance score of the in-link document
- the number of out-links originating from the in-link document

Recall that all documents have received a score in the content only phase which is Sc_n. The formula for calculating each document score is shown below. Let Λ be the scored set of documents from phase 1 and S_n be the off-site in-set of document n therefore we calculate Sc^2_n:

$$Sc^2{}_n = Sc_n + \left(Sc_n \times \left(\sum_{(m \in S_n)} \frac{Sc_m}{outdegree_m + 1} \right) \right) \qquad for(m \in \Lambda) \qquad (3.14)$$

This was calculated for the top 250 documents in the result-set. Once again, this figure can be changed as seen fit, but 250 was our best parameter cut-off point as found when running sample queries prior to running the actual experiments. Documents were ranked in decreasing order of Sc^2_n.

Co-citation & Spreading Activation

Our final experimental technique in TREC-9 (*dcu00lb*) was based on co-citation analysis, but incorporated spreading activation and was once again calculated for a best parameter valued subset of the top documents in the result-set. This algorithm views in-link associated documents as hub type documents. Recall that Kleinberg [Kleinberg, 98] describes documents in terms of hub documents and authority documents with hub documents acting as a source of links into similar documents while authority documents are seen as sources of authority on a topic and are gathered together into cohesive communities by groups of hub documents. Our theory is based on the belief that a good document is pointed at by good hub documents. But what makes good hub documents? The documents that a hub document link into influence the quality of a hub document, therefore if a hub document links into many good documents, then this hub is better than one that links into a smaller number of good documents or any number of lower quality documents.

Our method of generating a hub score for a document is based on spreading activation. In Figure 3.16 we can see that the score of hub document A is influenced by documents B, C, D and E so our implementation of spreading activation will result in the scores (or part of the scores) being transferred from these documents (as the dashed lines show) but no score is transferred from D who's score is 0.0 (hence no dashed line).

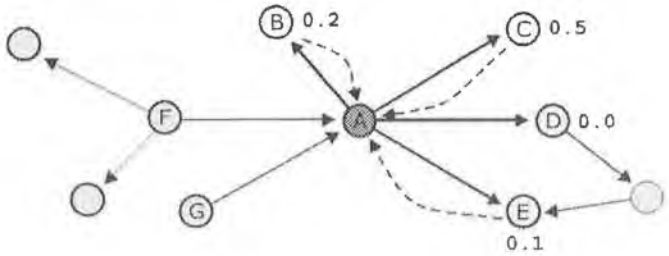


Figure 3.16 : How spreading activation influences a Hub document

Therefore, for any given document, we calculate hub scores (as in Figure 3.16) for each document in its in-set, regardless of whether the document is in the result set or not. Why this rule? We wanted to account for the fact that a hub document, though not ranked in the content-analysis phase may act as a source of links to scored documents.

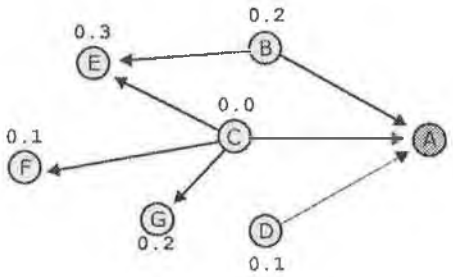


Figure 3.17 : Illustrating the possible influence of non-ranked hub documents

In Figure 3.17 document A has documents B, C and D in its in-set, but document C is not ranked. Examining the out-set of C, four ranked documents (including A) are present and employing co-citation means that the influence of these documents (E, F and G) will affect the score of A (and similar for E, F and G) so we do not want to ignore this information in our ranking formula. However, in a situation where the hub document is scored, then we wanted to include this fact in the ranking formula as well. Therefore, if a hub document is included in the relevant-set, its content-score (multiplied by a dampening factor α of value 0.45 which we selected as a best parameter value based on observing the results of sample queries prior to running the experiments), generates a score (along with the spreading activation score) for the hub document. Let A be the scored set of documents from phase 1, δ be a constant (also a best parameter value of 0.35) to limit the score being transferred from the target document of the link to the hub document, α be a constant to limit the score being transferred from the hub document to Sc^3_n during calculation of the hub document score (HSc_n), S_n

represent the in-set of document n and O_m represent the out-set of document m, giving the following formula:

$$HSc_m = (Sc_m \times \alpha) + \left(\sum_{p \in O_m} Sc_p \times \delta \right) \qquad for(p \in \Lambda) \qquad (3.15)$$

or, alternatively, if the hub document is not scored this formula is used:

$$HSc_m = \sum_{p \in O_m} Sc_p \times \delta \qquad for(p \in \Lambda) \qquad (3.16)$$

The consequence of this is that the hub document now has a score which reflects its own relevance as well as that of its out-link associated documents. Finally, this hub score is divided by the total number of out-links from it (+ 1), as was the case with the spreading activation experiment. This score is added to the Sc^3_n score of the document being re-ranked. The current values for α and δ were best-parameter values that we arrived while running the experiments. As mentioned previously, our best-parameter values were chosen by executing a number of (non-topic) queries on the dataset and examining the results.

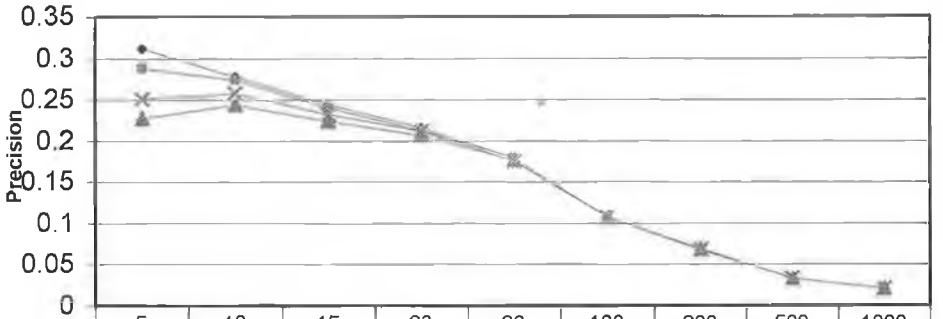
Finally, the Sc^3_n is generated from the original score Sc_n and all hub scores HSc_m from equation 3.15 or 3.16:

$$Sc^3_n = Sc_n + \sum_{m \in S_n} \frac{HSc_m}{outdegree_m + 1} \qquad (3.17)$$

The final Sc^3_n score is then used to rank the documents and the top 1,000 documents were submitted to TREC as one of our official runs.

3.4.2.4 RESULTS

Of the four approaches we submitted, the content-only run attained highest (or equal highest) precision across virtually all standard rank positions.

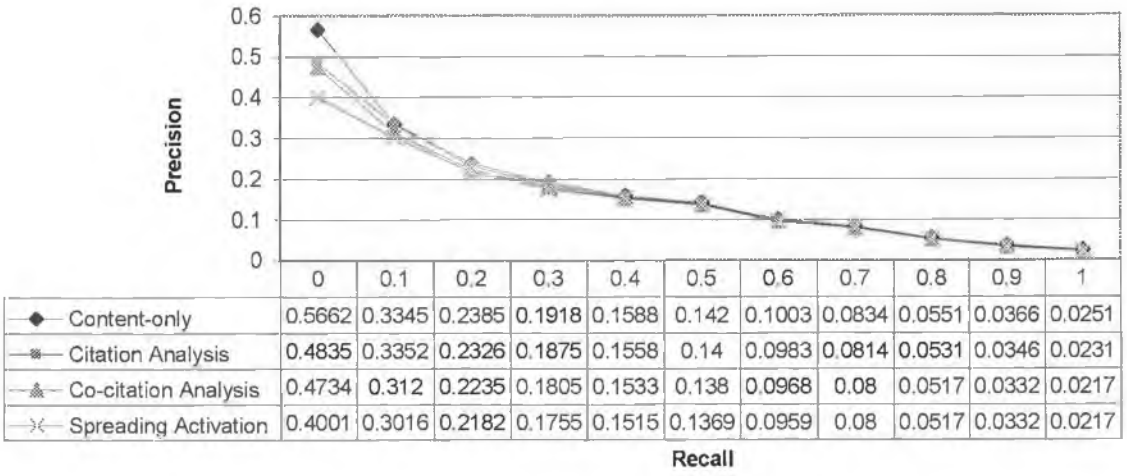


	5	10	15	20	30	100	200	500	1000
Content-only	0.312	0.278	0.244	0.216	0.18	0.1076	0.0686	0.033	0.0207
Citation Analysis	0.288	0.274	0.24	0.213	0.18	0.1076	0.0686	0.033	0.0207
Co-citation Analysis	0.228	0.244	0.224	0.207	0.1767	0.1076	0.0686	0.033	0.0207
Spreading Activation	0.252	0.258	0.232	0.212	0.176	0.1074	0.0685	0.033	0.0207

Documents

Figure 3.18 : Precision results of our TREC-9 Experiments

As can be seen from Figure 3.18, at all standard positions in a result set, the content-only experiment performed better or equal to than any of the linkage techniques outlined in the previous section. Given our use of best-parameter cut-off points in the experiments, equal values at points 500 and 1,000 are to be expected. Once again, we found these results to be disappointing. None of our algorithms seemed to be able to improve retrieval performance. The precision versus recall graph of all four experiments is shown in Figure 3.19.



	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Content-only	0.5662	0.3345	0.2385	0.1918	0.1588	0.142	0.1003	0.0834	0.0551	0.0366	0.0251
Citation Analysis	0.4835	0.3352	0.2326	0.1875	0.1558	0.14	0.0983	0.0814	0.0531	0.0346	0.0231
Co-citation Analysis	0.4734	0.312	0.2235	0.1805	0.1533	0.138	0.0968	0.08	0.0517	0.0332	0.0217
Spreading Activation	0.4001	0.3016	0.2182	0.1755	0.1515	0.1369	0.0959	0.08	0.0517	0.0332	0.0217

Figure 3.19 : Precision vs. recall graph for all four runs

Examining the results on a query-by-query basis, for all queries the content-experiment performed equally as well, or better than, all linkage based approaches. We believe that any situations in which the content-experiment was found to perform equally as well as the linkage approaches this was due to the sparseness of the linkage data. A lack of off-site links within a collection will leave minimal opportunity to rerank the documents.

For details of our average precision results for each of the four runs as well as the best, median and worst overall see Figure 3.20 below. As we can see, the four runs have produced very similar average precision figures across all the topics. This we feel is as a result of the sparsity of connectivity data available and our best-parameter constants that limited the number of documents reranked by the linkage algorithms. One solitary topic (topic 484) had its precision increased by applying the citation ranking algorithm and the spreading activation algorithm, but this is one single topic over the whole collection of topics and as such, we cannot draw any positive conclusions from this one result.

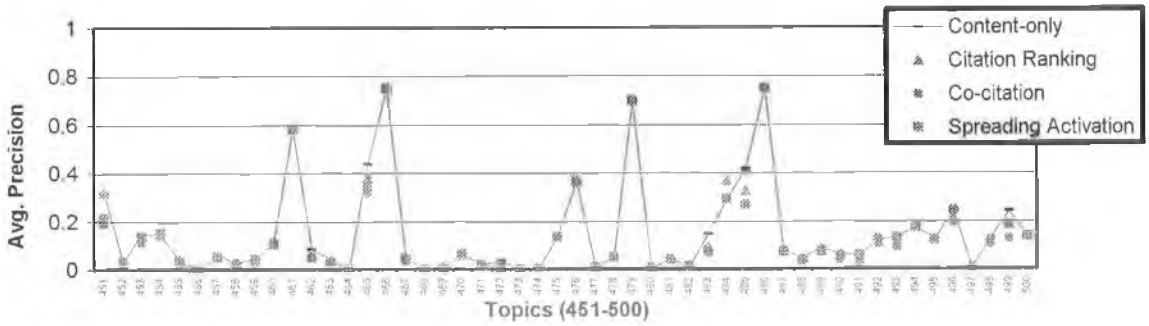


Figure 3.20 : Average precision per topic over all experiments

Our experiments have illustrated that applying our linkage analysis algorithms (all of which were non-iterative) failed to have any positive influence on retrieval performance. On the contrary, applying our algorithms will negatively affect retrieval performance. The cut-off parameters that we applied to our experiments serve to limit the (negative) impact of our algorithms. We were not alone in these findings. All other participants found that their experiments (both non-iterative and iterative) produced similar results. Consequently, it is clear that the nature of our experiments (non-iterative) was not the deciding factor in the success or otherwise of our experiments.

3.4.2.5. DISCUSSION

No group managed to increase precision on the TREC-9 Web track when using linkage information versus their own content-only runs and these results replicate the findings of the TREC-8 Web track. Once again, this came as a surprise to many of the participating groups as it was felt that the use of linkage analysis on the WWW does have tangible benefits and the WT10g test collection cleaned up some of the problems with the earlier WT2g.

A number of participating groups experimented with various linkage algorithms for the Web track of TREC-9. AT&T submitted runs [Singhal & Kaszkiel, 00] in both small and large web tasks using a new retrieval system called Tivra, which we will discuss in the next chapter. Overall, findings of AT&Ts experiments (including an experiment that uses anchor texts in one experiment, much like HVV) indicate that linkage analysis (as outlined above) does not help the retrieval effectiveness. However, it is stated that they would "not make this claim with certainty in the general Web environment, rather just with the Web track experiments on WT10g". Their best approach was an algorithm based on Rocchio query expansion [Singhal & Kaszkiel, 00], which this author programmed while researching the AT&T research facility at NJ. The other participating group that

utilised the anchor text associated with in-links was Justsystem Corporation. Their experiments centered on the theory of "aboutness" [Fujita, 00] as a representation of information objects (documents in this case). They also observe that no reliable improvement was observed when using anchor texts, but additional experiments were planned.

Two participating groups implemented experiments similar to our experiments in the previous year's TREC based on indegree & outdegree weighting. John Hopkins University [McNamee et al., 00] submitted two runs based on incorporating linkage data, which re-ranked a set of highly, scored documents based on a formula incorporating the indegree of the document. A similar technique came from Queens College [Kwok et al., 00] also with disappointing results.

As expected, a number of groups implemented Kleinberg's algorithm and PageRank but none with any success. Finally, using techniques similar to ours described above, TNO-TPD & University of Twente's [Kraaij & Westerveld, 00] implemented algorithms based on indegree, outdegree, co-citation and bibliographic coupling techniques, with obvious findings. Spreading Activation & Probabilistic Augmentation Systems from the Université de Neuchatêl [Savoy & Rasolofo, 00] were submitted again without success.

There were a number of reasons put forward both in the draft papers and at the TREC conference as to why the use of linkage analysis techniques decreased average precision, or in many cases, made no difference at all. In TREC-8 the linkage data did not contain the required density of off-site links to support linkage analysis experiments, so provision for this was made in the development of the WT10g dataset, which was generated to maximise the number of this link type. Therefore, why did no group manage to successfully incorporate linkage analysis techniques into their experiments? Are there not tried and tested techniques on the web as a whole which show that linkage analysis is of benefit to web search, or are we putting too much emphasis on linkage analysis as a panacea for many of the problems of searching the web? What is different between an implementation of Kleinberg's algorithm [Kleinberg, 98], or Henzinger & Bharat's 'improvements [Bharat & Henzinger, 98] to Kleinberg's algorithm or even Google's Pagerank algorithm [Page et al., 98] on the Web as opposed to on the WT10g dataset?

The obvious explanation lies in the dataset itself. Ultimately the WT10g dataset, like its predecessor, is a subset of a 100GB collection used for the Large Web Task, which (as mentioned) is itself a subset of a larger 300 GB subset of an Internet Archive crawl completed in early 1997. All

links in WT10g would come from the 100GB collection and any links to documents external to that 100GB collection would not have been included in the dataset. This limits us to a small subset of the indexable web, and we do not, therefore, have access to the links outside of this small subset of the Web. In addition, we do not have any information as to how the Web has changed since 1997. How sparse would a connectivity graph of the web be in 1997 when compared to the Web of today? Are more links being created on web pages today or less? Can we see a trend emerging as the web matures? All these questions remained unanswered after TREC-9.

Another explanation put forward for poor retrieval performance when linkage information was used was that TREC topics are not suitable for using linkage information due to the fact that they are too specific. A lot of this specificity is due to the synthetic nature of the topics. Although the TREC-9 Web track topics are extracted from query logs, the narrative associated with each topic often imposes a strict limitation on the scope of the query. At first glance however, this explanation may be seen as just an excuse for poor performance, due primarily, to the fact that the topic titles (upon which a lot of the queries were based) were chosen from an Excite query log. After all, these queries were actual Web queries and therefore were the Web track experiments to hold any authority in the real-world they must be able to work with actual web queries.

On the other hand, Kleinberg [Kleinberg, 98] does state that his algorithm works best on queries of a broad nature, examples being "Search Engines, Web Browsers" but that for narrow queries, such as "java malformedURLException reason" the algorithm does tend to generalise the topic and in this query the user may get back listings of java tutorials and resources. The specificity, which results from the presence of TREC topic narratives, will cause problems in such cases. Altavista researchers [Bharat & Henzinger, 98] have shown that incorporating content analysis into Kleinberg's algorithm does help to solve this problem of topic-drift. Not one participating group took this approach.

In looking at our own experiments and doing some failure analysis, we felt that the dataset must accept much of the responsibility. Our first experiments on the dataset had suggested that implementing techniques such as Kleinberg's algorithm or PageRank would not be successful, and other groups experiments have proven this point to be correct.

Another possible cause of our poor results would be the fact that TREC-9 topics contain on average only 47.4 relevant documents (as found using the pooling technique mentioned earlier) and to our experiments the number of these documents would have been quite important. In addition, with so few relevant documents in the dataset, the chance of many links existing between these documents is rather small. We did separate links into off-site and in-site links, which did aid us in only utilising links which may be of benefit, but there were too few of these.

In conclusion, we believed that efforts needed to be made to increase the density of off-site (in particular) links within a TREC dataset before any faithful experiments into linkage-based IR could be evaluated. There were three options open to us:

1. Wait until TREC realised the problem and using their (vastly superior when compared to our own) resources developed a new dataset that more accurately re-created the link structure of the WWW, and in so doing increased the density of off-site links.

2. Examine WT10g to see if it was possible to distill a more densely linked dataset out of the existing WT10g.

3. Generate our own test collection, which would reflect the linkage structure of the WWW, however resource issues made this the least likely option.

We chose to take the second option, which we will outline, in the following chapter.

3.5 Chapter Summary

In this chapter, we have discussed the concept of Relevance as it applies to information retrieval before describing how to measure the performance of an information retrieval system. The term 'relevant' is used to describe a document that satisfies a users information need. Relevance, however, is inherently humanly subjective and cannot merely be assigned by an automatic retrieval process.

In order to determine the quality of an information retrieval system, an evaluation is usually carried out. The primary measures of retrieval performance are Precision (fraction of retrieved documents that are relevant) and Recall (fraction of relevant documents that have been retrieved) which are often used in conjunction with test collections. Test collections comprise documents, queries and relevance judgements and are used in large-scale retrieval experiments. Generating a test collection is an extremely resource hungry activity and it requires organisations such as NIST, who have the resources, to co-ordinate large-scale retrieval experiments using test collections as is the case with the annual TREC series of conferences. The goal of the TREC series of conferences is to foster research into information retrieval and encourage participants to take part in retrieval benchmarking experiments in a spirit of openness and knowledge sharing.

In this chapter, we also discussed our TREC experiments for both the TREC-8 and TREC-9 web tracks. The aim of these tracks were to provide a framework within which participating research groups could come together and evaluate their retrieval techniques, including linkage-based techniques on a test collection of over 1.5 million web documents. Our TREC-8 experiments into linkage analysis (using a test collection provided by TREC called WT2g) were based around citation ranking and were unsuccessful at improving retrieval performance. This finding was mirrored by the other participants at TREC-8, which was a surprise to many as anecdotally it was felt that linkage analysis improved retrieval performance over conventional content-only retrieval. For the TREC-9 conference, we developed more advanced algorithms based on citation ranking, spreading activation and co-citation analysis incorporating spreading activation. The test collection provided by TREC was a much larger test collection called WT10g, which we have briefly examined. Our findings were that linkage-based retrieval was of no benefit to retrieval performance and these findings (once again) were shared by other participating groups. However, our belief was that the test collection was not capable of faithfully supporting linkage-based retrieval experiments. The next chapter shows the results of running our experiments on a revised version of the TREC-9 test collection.

Chapter 4

ADITIONAL EXPERIMENTS IN LINKAGE-BASED RETRIEVAL BEYOND TREC

In this chapter we discuss our experiments into Linkage based Retrieval beyond those carried out for the TREC Web Track of TREC-8 and TREC-9 as described in the previous chapter. Firstly, however, we will discuss SiteRank which is a version of PageRank that we developed for AT&T and is presented here as part of this research. Following on from SiteRank, we will identify some issues with the WT10g collection that cause problems for linkage-based retrieval experiments and proceed to extract from WT10g a densely linked subset in order to try to remedy these problems. Using this densely linked subset we run some experiments and show that moderate improvements in retrieval performance are possible using our densely linked subset and standard TREC evaluation procedures and measurements.

4.1 SITERANK – A NEW LINKAGE ANALYSIS ALGORITHM

During the course of this research, the opportunity arose to work on a project at AT&T Research Labs [AT&T, 02], NJ[32] to develop a linkage-based retrieval algorithm for a search engine called Tivra, which was under development at the time. Here we will discuss details of the algorithm that we developed.

4.1.1 TIVRA, THE AT&T SEARCH ENGINE

Tivra [Singhal & Kaszkiel, 00] was designed to be a large-scale web search engine, which was developed to run on conventional desktop workstations running Linux (similar to how Google operates), due to their being relatively inexpensive. Tivra was based on the SMART[33] retrieval system [Salton & McGill, 83] developed at Cornell in the 60s and used by many research groups ever since (in some variation) as a basis for retrieval experiments. Tivra had been benchmarked at indexing 15 gigabytes of web data per hour on a 700MHz Pentium PC with 1 gigabyte of RAM.

---

[32] AT&T Research Labs, Florham Park, New Jersey.

[33] Allegedly SMART is short for Salton's Magical Automatic Retriever of Text.

The Tivra system could compute several scores for each document. These scores could be based on:

- The document text (global)
- The document title
- Anchor text of off-site in-links
- Anchor text of on-site in-links
- Various proximity based scores (with extra credit applied for locality of query terms within the document body).

Tivra's ranking formula (excluding the linkage analysis component) views each document as a number of separate indexable components, each of which acts as a source of evidence, the influence of each source being regulated by best guess parameters.

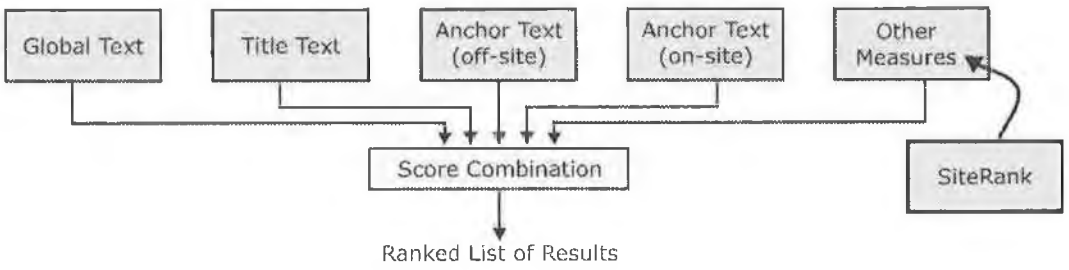


Figure 4.1 : Illustrating Tivra's use of numerous document representations

Figure 4.1 illustrates the different document representations used by Tivra when ranking documents. The linkage algorithm presented as part of this research was developed to a general AT&T specification for a query-independent website-centric algorithm, similar to PageRank, which would be included as one of the 'other measures' in Figure 4.1. Our original contribution, which is presented in this dissertation, was in taking this specification (for a PageRank style algorithm which operated on the average as opposed to the combined influence of documents from their respective websites) and developing the algorithm based on this specification as well as the associated software to operate efficiently over (at least) a 100 million-document collection and which could easily be integrated into Tivra's ranking algorithm. Letting n be a document, the formula that Tivra used to provide content-only results for the AT&T experiments for TREC 9 was similar to (the 'other measures' are not included) equation 4.1:

$$Sc_n = (1.0 \times GlobalWt_n) + (1.0 \times offSiteWt_n) + (0.25 \times onSiteWt_n) + (1.0 \times titleWt_n) \qquad (4.1)$$

However, the only linkage analysis component used for their TREC-9 experiments was based on HVV, in that anchor texts were used to indicate the content of a target document. The new component that was developed which shall be referred to as SiteRank, was not included in their TREC-9 experiments and unfortunately, we never integrated the algorithm into Tivra and any user experiment data we gathered was not processed prior to my departure. We will provide details of our own evaluation of a SiteRank algorithm later in this chapter.

4.1.2 The SiteRank Algorithm

As mentioned above, the SiteRank[34] algorithm was to be incorporated into a revised reranking formula for Tivra. Keeping in mind the 'Principles of Performance' that we discussed earlier the algorithm was a query independent algorithm which was based on PageRank, whereby a single score could be read from a vector of SiteRank values (one for each document) when the system was calculating final document ranking. We made a number of modifications to PageRank to produce SiteRank, which we will now describe in detail.

The SiteRank algorithm assigns a score (a rank) to a document based on the average rank of all documents grouped by websites that connect into it. Developing the algorithm to take a web site centric view of the WWW as opposed to an individual web document view, as is the case with PageRank, helps us to defeat the efforts of individuals who wish to artificially increase the rank of documents by exploiting certain loopholes in the PageRank algorithm.

4.1.2.1 An In-depth Examination of PageRank

Recall from Chapter 2 that the PageRank algorithm is a query-time, iterative algorithm that generates a single linkage score for each document, regardless of the content of the document. The simple version of PageRank (as described in Chapter 2) which generates a PageRank score Pr'_n for a document at each iteration is based on the following formula. Let n be some web page and S_n be the in-set associated with n, let out-degree$_m$ be the size of the out-set of a document m and let c be a value that is used for normalisation during the iterative process:

[34] SiteRank was so called because of the value it places on web sites in the rank calculation process

$$Pr'_n = c \cdot \sum_{m \in S_n} \frac{Pr_m}{outdegree_m} \qquad (4.2)$$

However implementing PageRank based on this formula does leave the algorithm open to two problems, namely dangling links and rank sinks. These problems have been identified by Brin and Page [Page et al., 98] and solutions have been incorporated into PageRank.

Dangling Links

Dangling links are links that point to a page which itself contains no out-links. In a real-world implementation of PageRank, there are a large number of such links, many of which are links that point at documents which the system knows about (and has anchor text descriptions for) but has not downloaded yet. Remember that a sizeable fraction of Google's declared index has not actually been downloaded by the algorithm, although assuming a weighted crawler implementing some form of link based crawling strategy [Cho et al., 98] which Larry Page[35] has researched while developing Google, the documents with highest linkage score will be downloaded first which would reduce the negative effect of dangling links. In Figure 4.2 an example of a dangling link would be the link from document A to document E because document E has no out-links (dashed lines are illustrative and are not actual links) so we do not know how or where the rank of E should be distributed.

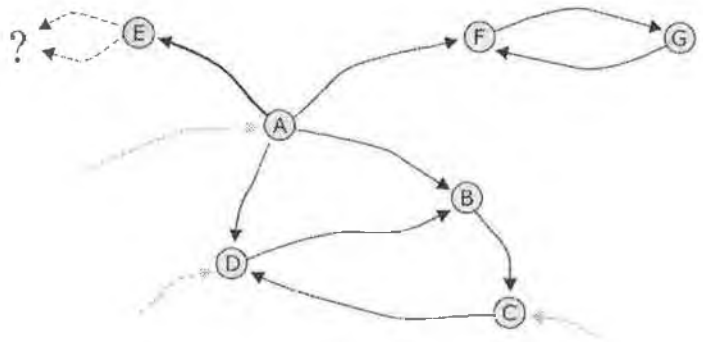


Figure 4.2 : illustrating the problem of Dangling Links

[35] Larry Page : the founding CEO of Google and now President of Products, formerly a Ph.D. candidate at Stanford with Sergey Brin who is now President of Technology at Google. Both Brin and Page developed Google while at Stanford.

If the PageRank from document E (Figure 4.2) is not redistributed at each iteration then it is lost from the process. However it is not clear how this PageRank should be distributed as there are no out-links to indicate target documents. Due to the fact that dangling links have no effect on other pages they are removed from the calculation and after the iterative process is completed they are added back in. The reason for their removal in an iterative process is that each iteration may produce more dangling links which themselves require removal. After the primary PageRank calculation process is complete the dangling links are added back in and iteration is rerun for as many times as it was required to remove the dangling links. Of course, the process of removing links from the connectivity graph has the effect of influencing the PageRanks calculated, but this effect is considered too small to cause concern [Page et al., 98].

Rank Sink

Rank Sinks occur because two or more pages that have out-links to each other, but to no other pages, gain an artificially high rank. Assuming we have at least one in-link into this set of pages from a page outside of this set, then at each iteration the rank enters these pages and never exits so these pages constantly accumulate rank at each iteration (never distributing it back into the main body of the linkage graph) and therefore artificially gain a higher rank than should be the case. This can be seen in the particular link structure of documents F and G in Figure 4.3 where a fraction of the rank from A is distributed to document F, which at the next iteration passes entirely to document G. One iteration later, document G passes the rank back to document F, which has been receiving additional rank from document A at each iteration. So we can see that rank which enters the rank sink never actually escapes and is constantly being augmented by rank from document A.

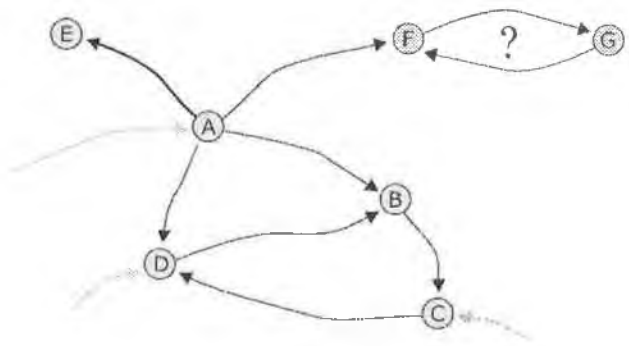


Figure 4.3 : Illustrating the problem of Rank Sinks

The issue of rank sinks may occur with any number of documents involved. Our example in Figure 4.3 only shows the problem for a small rank sink of two documents. Obviously, as the number of documents in the rank sink tends towards the required number of iterations then the effect of the problem becomes less noticeable because the rank will not be redistributed as often to document that comprise the rank sink.

The problem of rank sinks was overcome by the introduction of a virtual rank source, which gathers rank from all pages, including rank which would otherwise be lost in a rank sink, and using a vector E over all web pages in the index this gathered rank can be redistributed (in some way) back to all web pages in the collection. The intuition behind this is that a web surfer will traverse the web by following links, but at some point will become bored and jump to another (random page) which is modelled by the E vector which has an entry for each document and is used as an indicator of how to distribute any redundant rank back into the system. Each document's entry in the E vector represents the proportion of rank given to that document. In standard PageRank the E vector is uniform over all web pages with $||E|| = 0.15$.

The facility exists within the PageRank algorithm (as well as within SiteRank) to utilize the E vector to regulate the influence of particular documents. For example, if a user is interested in a particular topic such as 'motor sport' then the E vector could allow for results to be tailored towards that user's tastes applying positive values for pages that are highly scored for the 'motor racing' topic. This works by regulating what documents receive rank from the E vector at the end of each iteration. In this example, pages that are about 'motor sport' would receive most of the PageRank at the expense of all other pages. Consequently, these pages have a much higher influence on the final PageRanks of each document. In this way, PageRanks can be tailored to particular users or user groups. In the example put forward by Brin & Page the result of selecting just one page in the E vector, that of the computer scientist John McCarthy is that all the top results are related directly to John McCarthy. His homepage got the highest ranking, followed by documents linked to from his page.

We have seen that the E vector is a very powerful tool for focusing the user's search on different sections of the web. It is entirely feasible to identify topics on the web using a content-based clustering algorithm and maintain separate lists of PageRanks for each of identified topic. A user could identify topics which are of interest to him/her and have results of any search tailored to his/her interests. It is even possible (although wholly unreasonable) to generate separate PageRanks

for each individual user based on their 'favourites' or 'bookmarks' and tailor the results of a search accordingly. This could provide for personalised search engines, however unlikely this is to occur in reality due to the storage requirements of a vector of PageRanks for each user.

4.1.2.2 THE FULL ALGORITHM

By incorporating the idea of the E vector to solve the problem of rank sinks, we're now in a position to state the full PageRank formula. Letting E_n be some vector over the Web pages that corresponds to a source of rank, c is a constant which is maximised (normally 0.85) and $||Pr`|| = 1$ (sum of all PageRanks = 1), we have the following formula which is used to calculate the PageRank of a document at each iteration. Let n be some web page and S_n be the in-set associated with n, let out-degree$_m$ be the size of the out-set of a document m and let c be a constant that is used for normalisation during the iterative process with E be the E Vector we have:

$$Pr'_n = c \cdot \sum_{m \in S_n} \frac{Pr_m}{outdegree_m} + (1-c) \cdot E_n \qquad (4.3)$$

These PageRank values are calculated over a number of iterations until an acceptable level of convergence is reached. We are told that it takes up to 52 iterations until an acceptable convergence point is reached for a 24 million-document collection [Brin & Page, 98]. However the rate of convergence is influenced heavily by the size of the 'E' vector.

Unfortunately Brin and Page never published a full-scale evaluation of the benefit of PageRank to the indexing process so we are not in a position to provide definite answers as to its effectiveness using standard measures such as precision and recall.

4.1.2.3 REMAINING PROBLEMS WITH PAGERANK

The PageRank algorithm works in a similar manner to Kleinberg's algorithm, meaning that over a number of iterations, document ranks are distributed across links to target documents. Although Kleinberg calculated two scores for each document and operates at query-time on a considerably smaller collection of documents we can draw these parallels. Recall from the previous chapter the three problems that Henzinger & Bharat identified with Kleinberg's algorithm:

- Mutually re-enforcing relationships between hosts, which is solved by regulating the influence of documents from within one host.

- Automatically generated links, which is solved by content analysis.
- Non-relevant nodes, which is also solved by content analysis.

The latter two problems are solved by a search system which implements PageRank because the content score of a document is combined at query-time (using some unknown process) with the PageRank score and in this way integrates content-analysis into the ranking algorithm. However the first problem is not solved by PageRank. It is feasible for an individual who understands how the PageRank algorithm works to exploit this problem in order to influence the ranking process and the problem is compounded by the E vector. Recall that the E vector acts as a source of rank over all documents and that at any iteration a fraction (0.15) of the collection-wide PageRank score is passed back to all documents using the E vector. Therefore, any unscrupulous individual could create a large number of pages all of which link into one target page that the individual is trying to rank highly in a result-set. It does not matter how many web sites the individual used to carry out this operation, all that is important is that a large number of pages all target one page. In Figure 4.4 below, document A is the target of the spamming process and numerous documents from three websites all link to it.

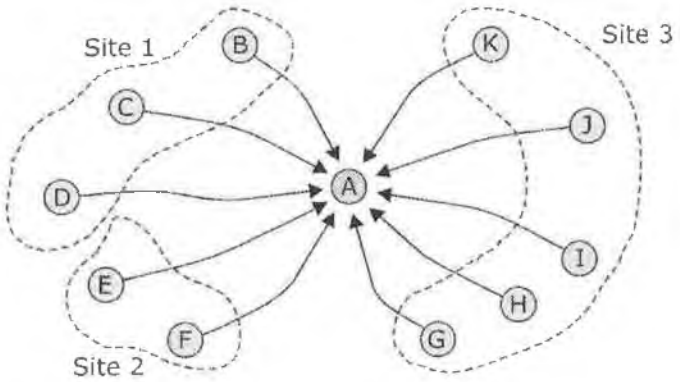


Figure 4.4 : Illustrating an unsuccessful PageRank spamming technique

However, this is a naive approach to spamming PageRank and will fail to have the desired effect. PageRank will have identified document A as a target of dangling links and will have removed it from the algorithmic process until after convergence has been reached. Simply linking from A to any other document on the web still improves matters for the spammer, but we are not allowing A to keep the rank it already has. Linking from A to documents B-K will not work either as this becomes a Rank-Sink, so a rather more clever approach is needed in order to spam PageRank, as shown in Figure 4.5.

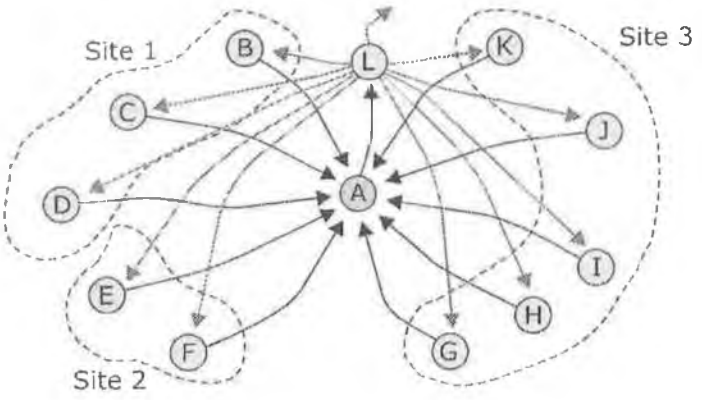


Figure 4.5 : Illustrating a successful Google spamming technique.

4.1.2.4 TO SUCCESSFULLY SPAM A PAGERANK ALGORITHM

Using Figure 4.5 as a worked example, we can see that the way to successfully spam a PageRank algorithm is to have document A containing a link back into the main body of the web so that it is not removed in a dangling link removal procedure. The way we have chosen to illustrate this is that a link exists from document A to another document L which will link back into each of documents B-K and although not essential (but this is a better idea because we assume that this loophole will have been noticed) a link back into the main body of the web, to any other page. In this way all of A's rank (except that that goes to the E vector) is transferred to L which transfers its rank divided evenly across all its link with the vast majority going back into documents B-K which then will give their higher rank back into A once again. Therefore, at each iteration B-K will constantly increase in rank and pass most of this accumulated rank onto A (which through L passes it back to B-K).

So where does the E vector fit into this? Well, the rank that is passed back to each document from the E vector, at each iteration, is based on the following formula, assuming uniform values for each document in the E vector. Letting N be the size of the collection, d be any document within that collection N and $||E||$ be the norm of the vector E (which is usually 0.15):

$$R_d = R_d + \frac{\|E\|}{N} \qquad (4.4)$$

So, regardless of the sparsity of links into documents B-K they will always receive rank from the E vector anyway and since the spamming process is very unlikely to be done manually it is a trivial task to develop software to automatically generate a large number of documents all of which link into a certain target document.

In 2000 a process that became known as the 'Liv Tyler nude' spamming of Google [Sullivan, 02] occurred, where a large number of pages[36] (concerning various actresses) were generated, all containing numerous links to the main page of the Nude Celebrity World News website. We assume that the reason was to boost the PageRank of the Nude Celebrity World News website. Google deny that the algorithm had much of an effect on its ranking and we are not in a position to validate this as the last public information on the PageRank algorithm was released in 1998 and the algorithm has likely undergone many modifications since then.

4.1.3 Our SiteRank Algorithm Description

The SiteRank algorithm was developed to be scalable to document collections up to (at least) 100 million documents, although we never applied any limits on document numbers in the design and programming phases of development. Recall the two problems that exist with the naive approach to PageRank outlined in the previous chapter. Dangling links point to documents that have no out-links therefore they would accumulate rank and never distribute any of it, thus acting as a rank leak from the whole graph.

We employed an alternative approach to the one taken in PageRank to solving these problems when developing SiteRank. Our technique does not involve the graph-pruning phase of PageRank, and also solves the second problem of Rank-Sinks, but only if implemented as part of the SiteRank algorithm and not just PageRank. PageRank views a static probability of a user becoming bored and jumping to a random page, whereas our approach assumes that a user is more likely to get bored on a page with a small number of links than a page with a large number of links. How we do this is as follows. Before any iterations were made, the algorithm added one new artificial node into the graph, which we referred to as the E document. This E document was connected to all other documents (nodes) via an in-link and an out-link and was given a pre-iteration value of 0.0.

[36] 8,004 pages in total, four for each of 2,001 celebrities, spread evenly over four web sites

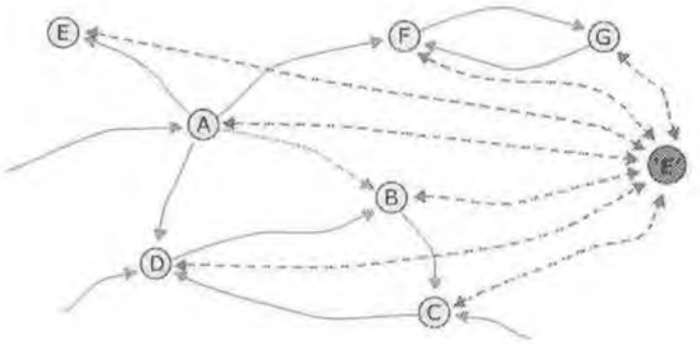


Figure 4.6 : Illustrating the E document

In this way a rank-sink will always have a 'leak' into the E document from all associated documents and a document at the end of a dangling link will always pass its full page-rank from the previous iteration directly into the E document. Associated with the E document is the E vector, a vector over all web pages that re-distributes the rank of the E document back to all web pages at the end of each iteration. Similar to the PageRank algorithm, it would be possible to provide for personalised SiteRanks by having non-uniform values in the E vector. Due to the fact that all documents have a link back to the E document, $||E||$ (the norm of vector E) will not be 0.15, but will have a value based on the following formula, assuming $||PR|| = 1.0$ and D is the set of documents in the system:

$$\|E\| = \frac{|D|}{\left(\sum_{n \in D} outdegree_n\right) + |D|} \qquad (4.5)$$

During each iteration, the rank that would otherwise be lost from dangling links and rank sinks is gathered by the E document is re-integrated into the system after each iteration, so that no rank is lost. In our work with the SiteRank algorithm, the E vector has values that are uniform over all web pages as shown in Figure 4.7.

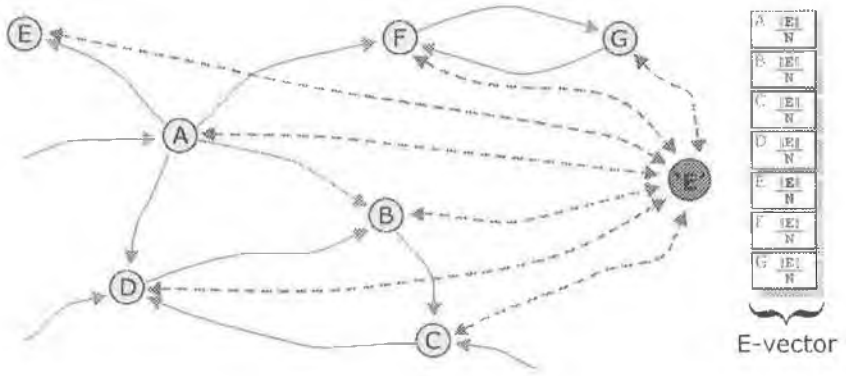


Figure 4.7 : Our use of the E-vector to distribute rank from the E document

4.1.3.1 INCORPORATING WEBSITES INTO THE PROCESS

We have previously (in Chapter 2) mentioned the problem of search engine positioning or optimisation whereby companies provide services which aim to improve the positioning of a (mostly commercial) website within search engine result pages. This process may take the form of artificially constructing a synthetic link structure around websites in order to influence the PageRank scores of documents form a particular websites, of which we have just shown an example. Our work at AT&T in developing the SiteRank algorithm would help to overcome this problem by taking a web site centric view of the PageRank calculation process.

Thus far, we have explained our process for generating a rank for each document which is based on the PageRank algorithm, but what really differentiates SiteRank from PageRank is this web site centric view which aids the avoidance of the problems of (commercial) manipulation of ranked results. Recall from [Bharat & Henzinger, 98] that we view all documents within a domain as having being written/developed by the one author and representing the ideas of a single individual or organisation. SiteRank limits the influence any one author may have on any document, regardless of the number of documents from any one website that link into that particular document. For example, if a document has three in-links, each of which are from one website, then the SiteRank of the target document is based on the average[37] SiteRank of these three documents that point into it.

[37] Average rank : we had considered using the max rank, but felt that the average rank would more accurately reflect the quality of the website that the documents originate from.

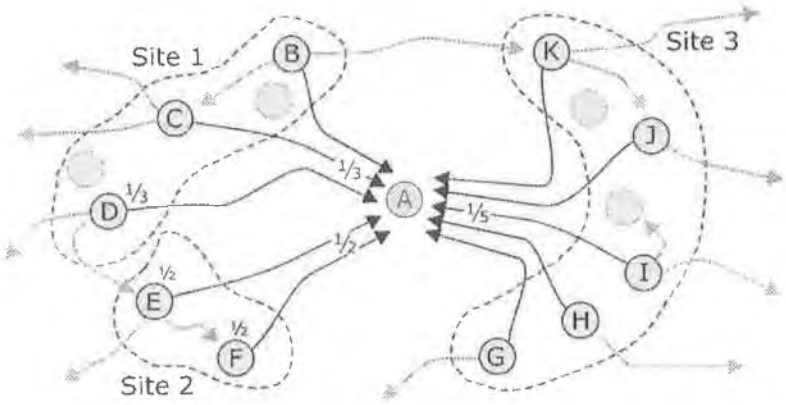


Figure 4.8 : Illustrating SiteRank for a single document

In Figure 4.8, the rank of document A would be based on the rank of all pages in the range of B-K. Recall from the basic PageRank algorithm that document B would transfer one third of its rank to Document A (because of the three links out of B), along with fractional rank from all other documents that link into A. However, we view all documents on one host having the one author which was identified in the previous chapter as the problem of 'Mutually Re- inforcing Relationships Between Hosts' which serves to increase the rank of the document on the second host and in so doing gives undue weight to the opinions of one person. Ideally, we would like all documents on a single host have the same influence as a single document would from another host. To achieve this we give fractional weights to edges in such cases, therefore the rank which is transferred from all the documents on site 1 (B, C and D) to document A is divided by the number of documents on site 1 that actually link into document A[38], giving the following formula to calculate the rank transferred to document A. Assuming S is the set of documents on a particular site that link into document A:

$$R_A = R_A + \sum_{n \in S} \frac{\frac{R_n}{outdegree_n}}{|S|} \qquad (4.6)$$

We have applied this technique to our SiteRank algorithm.

[38] Note we distinguish between the number of documents on site 1 and the number of documents that link into document A from site 1.

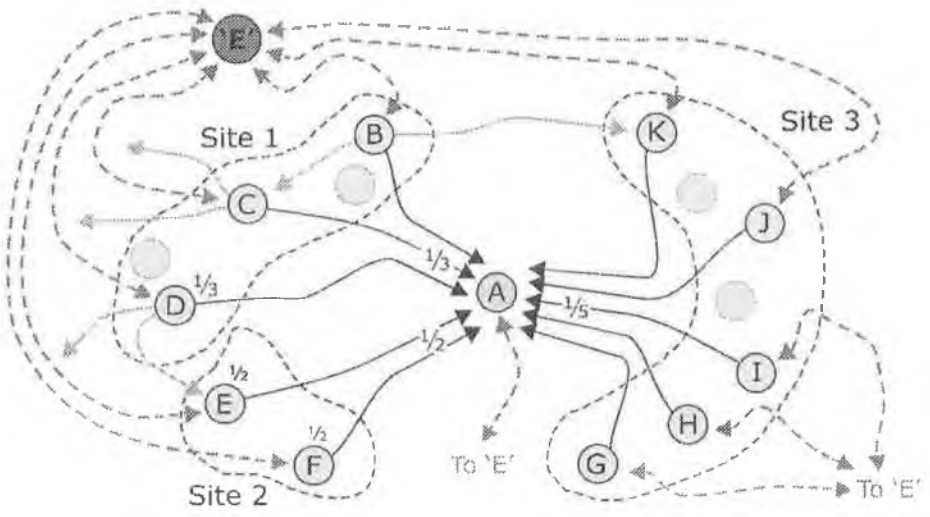


Figure 4.9 : Illustrating SiteRank for one document incorporating the E document

If we integrate the E document again into the process, all documents must pass a fraction of their rank to the E document at each iteration. Letting R_d be the current rank of document d and outdegree_d be the outdegree of d, not including the link to the E document, we have:

$$\|E\| = \|E\| + \frac{R_d}{outdegree_d + 1} \qquad (4.7)$$

In this way, the SiteRank of a document is based on the rank of the pages that link into it, yet we base our calculation on web sites as opposed to individual web pages. This has the intuitive result of ranking by the number and quality of websites that link into a document as opposed to simply the number of documents that are the source of links. Therefore it would be more expensive for a would be spammer to defeat this algorithm as the number and quality of websites play an important role in the SiteRank calculation process.

The algorithm to calculate SiteRank scores for each document n from a set of documents N is shown below. Note we are assuming that both lost SiteRank (rank lost from the system due to averaging the influence of documents from any one website) and E document SiteRank are gathered by one component and both redistributed using the E vector. Let N be the set of documents, SR_n be the current SiteRank for each document, SR'_n be the new SiteRank scores being calculated at this iteration, v, h and l be sets of webpages and $LostSr$ be the component that gathers all lost rank from the system:

$$SR_n \leftarrow \frac{1}{N} \qquad \text{for all } n \text{ in } N$$

loop :

for $n = 1,2,\ldots N$:

Let v *be the inset of* n

Let h *donate all sites within* v

for $j = 1,2,\ldots, |h|$

Let l *donate all pages within both* h *and* v

$$HR_j = \frac{\sum_{l \in v, l \in h} \frac{SR_l}{outdegree_l + 1}}{|l|}$$

$$LostSr \pm \left(\sum_{l \in v, l \in h} \frac{SR_l}{outdegree_l} \right) - HR_j$$

end

$$SR_j \leftarrow SR_j + \sum_{m \in h} HR_m \qquad \text{for all } m \text{ in } h$$

end

$$SR'_n \leftarrow SR'_n + (LostSr \times E) \qquad \text{for all } n \text{ in } N$$

$$SR_n \leftarrow SR'_n \qquad \text{for all } n \text{ in } N$$

while (not converged)

Had we gathered the E document rank and the rank lost from the system due to our solution of the 'Mutually Reinforcing Relationships between Hosts' problem separately it would not have affected the overall rank being allocated to each document. In our original AT&T implementation of the algorithm we did indeed gather both separately and passed back the rank separately.

4.1.4 Development Details

Language & Machinery

The computer we used to run our experimental SiteRank algorithm at AT&T was a UNIX based machine with 2GB RAM available to us for the calculations. We used single precision floating point numbers to store SiteRank values. Based on the 2GB upper memory limit, there was a limit of 256 million documents for which we can calculate a PageRank score (given that two PageRanks must be stored for each document during the iterative process). On a 10 million document collection, the total processing time to reach acceptable convergence (roughly 10 iterations) was about 8 minutes when we had all connectivity information available in RAM. In order to solve the memory problem of

keeping two PageRank scores for each document in RAM, Brin and Page use a two-pass process where one of the arrays of PageRank values are written to disk and not held in RAM.

Owing to the fact that the results of the experiments were never examined, we are not in a position to evaluate the retrieval performance of SiteRank, but we did have the option of evaluating the algorithm on the WT10g dataset. However, our previous experiences of using WT10g did not inspire confidence and we felt that evaluating a SiteRank algorithm on WT10g would have been a fruitless exercise. Rather, we turned our attention to examining WT10g in an effort to identify if any subset of its 1.69 million documents could be used to evaluate our algorithms.

The following table shows the top 15 documents from a SiteRank calculation process, ranked in decreasing order of SiteRank. These results appear to be intuitive in that these web pages could be considered to be the most popular on the WWW. A listing of the top 100 scored documents is included in appendix B.

RANK	SITERANK	URL
0	28.075829	http//www.netscape.com/
1	17.856512	http//www.microsoft.com/
2	16.047762	http//www.yahoo.com/
3	14.709599	http//home.netscape.com/
4	13.441017	http//www.microsoft.com/ie/
5	10.052464	http//www.adobe.com/
6	8.126648	http//home.netscape.com/comprod/mirror/index.html
7	7.921433	http//www.adobe.com/products/acrobat/readstep.html/
8	7.804879	http//www.excite.com/
9	6.879511	http//www.lycos.com/
10	6.708895	http//www.real.com/
11	6.688189	http//www.digits.com/
12	6.140473	http//worldwidemart.com/scripts/
13	5.627914	http//www.freeservers.com/
14	5.052332	http//www.stpt.com/

Table 4.1 : The SiteRank score of the top 15 documents.

4.2 EXAMINING WT10g AGAIN

Following from our experiences of working on the development of SiteRank using a 10 million-document dataset and our experiences with the TREC dataset when running our TREC experiments, we were in a position to identify two fundamental problems that seemed to be holding back successfully incorporating linkage-based retrieval techniques into web search experiments and the consequent evaluation using tried and trusted TREC evaluation procedures. These problems are:

- The lack of a suitable interlinked dataset to support the experiments. It seems to us (as well as other TREC participants) that WT10g was not capable of supporting this role.

- How to combine evidence from linkage and content sources to produce a final ranking for a document? The obvious approach is to regulate the influence of both sources by incorporating some best guess or tuned parameters, but we introduce an alternative technique based on a two-phase retrieval process, which we outline and evaluate later.

The rest of this chapter will be dedicated to describing our experiments using a subset of WT10g called WT_Connected, which increases the proportion of documents that have a non-zero off-site indegree.

4.2.1 WT_CONNECTED: A DENSELY LINKED SUBSET OF WT10g

Owing to the lack of off-site links within the WT10g dataset we felt it necessary to examine the dataset for clues as to how to proceed with our experiments. The WT10g corpus contained 171,740 off-site links, but if we examine the link structure of WT10g in detail, it becomes obvious that only 31,227 (or less than 2% of) documents actually are the target of at least one off-site in-link. Consequently, how could we reasonably expect any linkage analysis technique to improve retrieval performance given that such a small proportion of documents could have their rankings influenced by off-site in-links. As we have seen from chapter 3, none of the (experimental and anecdotally trusted) linkage analysis algorithms executed upon WT10g have resulted in any improvement in retrieval performance for any participating group.

Our experience and intuition suggested that it would be beneficial to isolate a densely linked subset from within WT10g and carry out experiments on this densely linked subset as previous

findings have illustrated clearly that carrying out experiments on the whole dataset was fruitless. Figure 4.10 shows the source of WT_Connected.

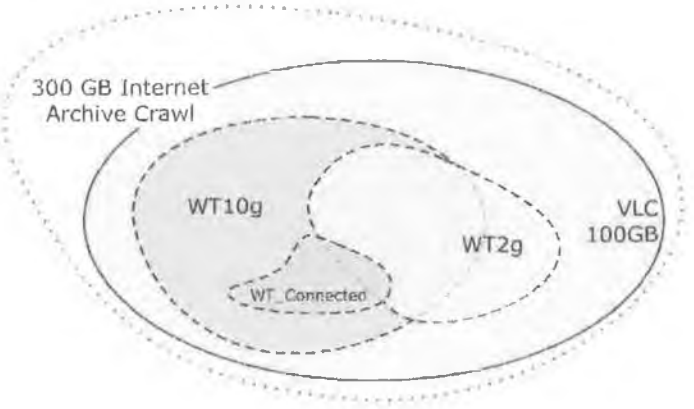


Figure 4.10 : Illustrating the source of WT_Connected

The primary advantage of using a subset of WT10g was that we still had the TREC relevance judgements available. This allowed us to evaluate our experiments using the prepared relevance judgements and by evaluating a content-only experiment we then could compare all of our linkage experiments against the benchmark content-only experiment. An alternative approach would have been to develop a web crawler (which in itself is not a trivial task) and gather a set of web documents (both time and resource consuming), but the generation of a set of topics and relevance judgements for these topics would have been beyond the resources at our disposal.

When generating this densely linked subset of WT10g we had two requirements for the new collection, these being:

1. To maximise the number of off-site links in the dataset.
2. To maximise the size of the new dataset itself.

Generating a dataset to satisfy these two rules was not difficult, we simply generated the dataset by following a four step procedure as described below.

1. We identified all the documents that have a non-zero off-site indegree. This produced a set of 31,227 documents and a total of 171,740 off-site links were examined in this step.
2. Based on these 31,227 documents we identified all documents from which these off-site links originated. There produced another 94,309 documents for the dataset which were necessary as without these documents the 171,740 off-site links would no longer exist within the dataset.

3. All documents identified in steps 1 and 2 were combined to produce a set of 120,494 unique documents.
4. Finally, all links between these documents were extracted from our Connectivity Server and used to support our experiments.

Figure 4.11 illustrates the links that identified the documents that comprise WT_Connected. They were the documents from WT10g that were either the source or the target of an off-site link (i.e. any document linked by links *a*...*f*).

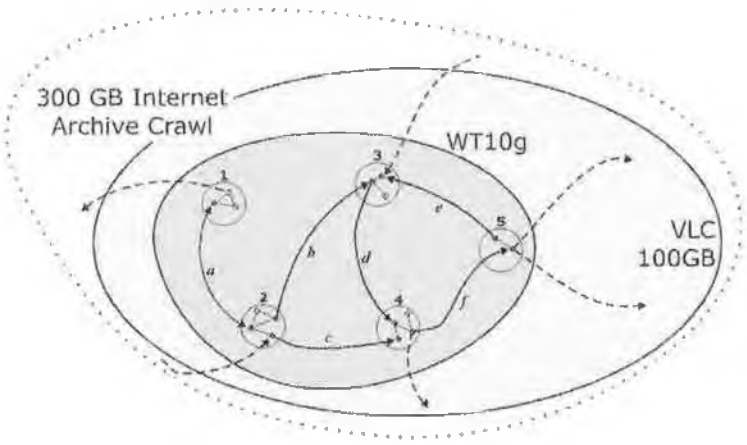


Figure 4.11 : Illustrating the documents that comprise WT_Connected

Doubtless there are more complex and sophisticated techniques that we could use based on, for example, graph traversal algorithms, that would provide us with a strongly connected component as opposed to a well linked subset, but we were not attempting to convert WT10g into the ideal dataset, rather we were interested in proof of concept that linkage improvements can be found using a dataset with a more dense off-site link structure, one that represents the WWW more accurately than WT10g.

After we generated the densely linked subset of WT10g, we processed the TREC relevance judgements, which provided relevance information over the whole WT10g collection and filtered out new relevance judgements based on the densely linked subset of WT10g only. This would allow us to

generate precision and recall values using TRECEVAL[39] that are more representative of true retrieval performance when evaluating experimental algorithms on the new dataset, which we will refer to as WT_Connected.

4.2.2 Comparing WT_Connected with WT10g

The strongly connected dataset is comprised of each document with a non-zero off-site indegree and the documents that are the source of the links into the first set of documents. The composition of WT_Connected (WT_Conn in this and all subsequent tables) compared with WT10g is summarised in the following table.

	WT10g	WT_ Conn
Number of Documents	1,692,096	120,494
Number of off-site links	171,740	171,740
Average off-site indegree	0.10	1.43
% of docs with non-zero off-site indegree	1.8%	25.9%
number of unique servers represented	11,680	11,611
Average number of docs per server	144	10
Generality	0.0015%	0.0021%
Number of Queries	50	36
Average number of relevant documents per query	52	7
Maximum number of relevant documents per query	519	26
Minimum number of relevant documents per query	1	1

Table 4.2 : Comparing WT10g and WT_Connected

As can be seen from Table 4.2, WT_Connected contains a far higher density of off-site links, an average of 1.43 per document while keeping the generality of the dataset for all queries very similar to WT10g. As expected the number of servers represented was almost identical in WT_Connected as it was in WT10g, this is because of the inclusion of all off-site links and both the source and target of each link. The one drawback of this is that the average number of documents on each server is only

[39] TRECEVAL is a program (provided by NIST) that evaluates TREC results using standard evaluation procedures such as Precision and Recall. TRECEVAL operates using relevance judgement listings provided by TREC, or in our case modified judgements filtered from the NIST provided relevance judgements over WT10g for our query set.

10 as opposed to 144 with WT10g, this is 6.9% of the number from WT10g. This is unavoidable as we only have 7.1% of the WT10g dataset represented in WT_Connected, but it means that we are taking the core pages, home pages and top pages from almost all of the 11,680 web servers in the WT10g collection. In addition, fourteen of the fifty TREC-9 queries had no relevant documents in WT_Connected and therefore our number of queries was reduced to 36.

Once we had generated this dataset we turned our attention to the problem of how best to combine evidence from both content and linkage sources when generating a final document ranking. This was the second fundamental issue that needed to be addressed before we ran our experiments.

4.3 Regulating the influence of Linkage Analysis

Regardless of the algorithm used to generate a linkage score for a document, the method of combining linkage scores with the content-only scores to produce the optimal ranking formula is a key problem that must be solved. This issue of how best to combine evidence from content sources and linkage sources has been overlooked by much of the research community who have been more focused on developing new and improved versions of algorithms such as Kleinberg's algorithm or PageRank.

Were a dataset such as WT_Connected to be able to faithfully support linkage experiments, problems combining content and linkage weights would lead to less than optimal retrieval performance or worse still could lead to negative evaluation of algorithms that otherwise could provide positive results. Recall that a number of different scores may need to be combined together to produce a final ranked output. For example a linkage analysis experiment may have to combine together scores from at least three sources:

- Content-only score
- Anchor text description score
- Numerical linkage score (e.g. PageRank).

A naïve approach (and common starting point) to solving this problem is to use some best guess parameter to generate a formula such as the following where $Sim(Desc_d,q)$ is the similarity score of the in-link anchor texts to the query, $Sim(d,q)$ is standard query document similarity score for the document content and PR_d is some PageRank style linkage score for a document and α, δ, λ are

constants used to regulate the influence of different components, we can generate a final weighting for a document *d* based on:

$$Wt_d = \alpha \times Sim_{d,q} + \delta \times Sim_{Desc_d,q} + \lambda \times PR_d \qquad (4.8)$$

The parameters (constants) α, δ, λ may be based on best guess figures or have been generated as a result of a parameter tuning process in which the optimal parameter values are found through a process of experimentation. We integrated an alternative technique to choosing and tuning parameters when deciding on values for the constants α, δ, λ into our experiments on WT_Connected and integrated a comparison between best-guess parameters and our alternative technique into our experiments.

4.3.1 The Query holds the Key to regulating the Influence of Linkage Analysis

Much work in the field of linkage analysis is evaluated using queries that could be considered broad in nature, which would suggest linkage analysis works best for broad queries. It is this idea that is exploited in our alternative technique to regulate the influence of linkage analysis. Kleinberg [Kleinberg, 98] states that there are two types of queries, broad and narrow. Narrow queries, he states, are representative of the *scarcity problem* in that there are very few pages that contain the required information, and it is difficult to determine the identity of these pages. Much of the classical work in information retrieval (such as the ranking techniques outlined in the first chapter) has focused on this type of problem.

The other types of queries are broad-topic queries, where many (tens or hundreds of) thousands of pages could be considered relevant. The scarcity problem is not an issue here, rather the *abundance problem*, where the number of pages that could reasonably be returned as relevant is far too large for a human to digest. Queries like this are best suited to the popularity or authoritative ranking of linkage analysis. If we examine the queries used in the evaluation of Kleinberg's algorithm by both Kleinberg [Kleinberg, 98] and Henzinger & Bharat [Bharat & Henzinger, 98] and even the experiments undertaken at AT&T into linkage based web search [Amento et al., 00] we notice that all of the queries could be considered to be broad in nature. We will return to the problem of how to automatically classify queries as being broad, narrow or neither later in this chapter.

4.3.1.1 The Existing Techniques which mine Information from Queries

In addition to using best-guess or experimentally tuned parameters, one recent attempt to combine linkage and content evidence has been to use the length (in words) of the query to indicate

the broadness of the topic represented by the query. The TREC-9 participation of AT&T [Singhal & Kaszkiel, 00] submitted one experimental run executed on WT10g where the contribution of the anchor text description of a document was reduced as a query string got longer, the idea being that short queries result in the abundance problem and consequently benefit from incorporating linkage analysis into the ranking formula, whereas long queries avoid abundance issues and thus gain reduced or even negative benefit from allowing linkage analysis to influence final document ranking. Unfortunately, as we have seen in the previous chapter, WT10g did not faithfully support the evaluation of the benefit (if any) to be gained by incorporating this technique.

At TREC-2001, a group from University of Twente & TNO-TPD [Westerveld et al., 01] implemented a similar approach to AT&T based on the length of the query. They normalised the document score by the query length (signified by α) with the resultant effect of the linkage score as in the formula:

$$Sc_d = (\alpha \cdot (Sim_{d,q})) + ((1-\alpha) \cdot (linkage\ score_d)) \qquad (4.9)$$

This technique seems intuitive and promising, but if we assume that broad queries are better suited to linkage based retrieval, then an alternative technique based on the number of documents considered relevant for a particular query could be beneficial. This we refer to as the scarcity-abundance technique for regulating linkage influence and we will now discuss it.

4.3.2 The Scarcity-Abundance Technique for Regulating Linkage Influence

The scarcity-abundance technique for regulating linkage influence exploits the size of the result-set for a query returned from a content-only experiment to identify how broad a query is and to regulate the influence of linkage analysis accordingly. It is intuitive that a query with broad focus will result in a proportionally larger result-set (causing the abundance problem) than a query with narrow focus. Therefore, if a query is identified as being broad in nature, then the influence of linkage analysis will be increased with the influence of content scores being reduced accordingly in an effort to present the user with the most qualitative pages from the relevant-set ranked highest. However, we must examine the concepts of Abundance and Scarcity as they apply to Information Retrieval systems before we describe the scarcity/abundance technique.

Abundance

Recall that the *abundance* problem refers to a situation in which the user of an IR system is presented with too many documents in response to an information need and hence the result set is

too large for a human to digest in a reasonable amount of time. If we assume that *perfect abundance* occurs only when all documents in the dataset are returned as being relevant to a query, then it follows that standard IR techniques will have difficulties in ranking the documents. A real-world example would be a situation where one must automatically choose the 'best' documents on the Web. One would instantly turn to some citation counting, PageRank, or Kleinberg type technique to identify the most qualitative pages. Consequently we would be very reliant on Linkage Analysis to choose the best documents to rank highly. To this end any query that generates perfect abundance should apply linkage analysis techniques as the ranking technique and the influence of content-based score should be reduced to zero. The influence of linkage scores should be reduced progressively as the result-set size decreases and the abundance problem becomes less of an issue for the user.

Scarcity

Recall that the scarcity problem is one in which very few pages contain the required information, and much work has been done in the field to determine the identity of these pages. The problem of *near-perfect scarcity* is one in which a single document (or a small, humanly manageable number of documents) can be highly weighted in response to a user's information need[40]. Consequently the use of linkage analysis techniques is of zero benefit to this result-set. So, at any point in between, perfect abundance and near-perfect scarcity both linkage and content scores will need to exert some influence to produce an optimal ranking formula.

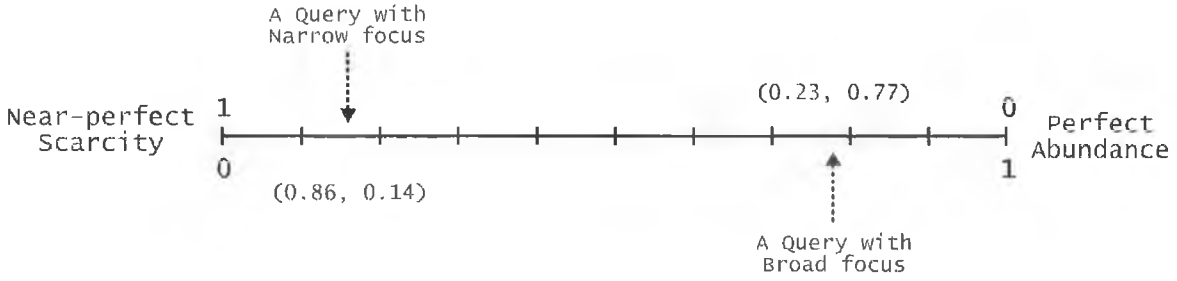


Figure 4.12 : Illustrating the sliding scale of linkage and content influence in the scarcity/abundance technique

Our experimental technique, which we call the *scarcity-abundance technique* incorporates a sliding scale upon which the optimal rate of influence of linkage analysis can be identified. If the scarcity value for a query is identified as being α then the abundance value for the same query must be $1-\alpha$. In Figure 4.12, we see two queries plotted; if we take the narrow focus query, which has a scarcity

[40] We avoid the issue of perfect-scarcity as were this to occur, zero relevant documents would be found and this being the case, no IR technique can operate.

score of 0.86, the associated abundance score (*1 - 0.86*) is *0.14*. We directly map from the scarcity value into a value for the parameter that influences the content-only score and a similar direct mapping exists from abundance to the linkage influencing parameter and we are only assuming two sources of information, content and linkage, though the technique may be expanded to incorporate other sources of evidence. This would require the mapping of each query onto a point in an n dimensional 'query space', where n is the number of sources of evidence available.

4.3.2.1 How can we identify a broad query?

In order to validate our belief that a broad topic query is one that results in a large set of relevant documents we carried out a web-based user study in which each user was asked to categorise 50 web-log queries into one of five categories (Very Broad; Broad; Standard; Narrow; Very Narrow). In total 25 users took part in the evaluations resulting in a total of 1,250 individual query evaluations. We randomly selected only 250 queries to be evaluated and since broadness/narrowness values for a query are inherently subjective we ensured that 5 different users would evaluate each query. To further combat bias in the experiments the queries were presented to the users in random order. Figure 4.13 is a screenshot of the judgement interface for one user.


If you are not familair with any meanings of the queries then please do go to www.dictionary.com or similar to disambiguate any term. It is very important that you are familiar with each query before you submit your judgements.

ID	QUERY						
1	books on tape of the month club	○	○	◉	○	○	○
2	where do i find a romantic christmas gift	○	◉	○	○	○	○
3	games	○	◉	○	○	○	○
4	What are some of the Schools Bill Gates attended	○	○	○	○	◉	○
5	www.santa.com	○	○	○	○	○	◉
6	nasa pictures of the Apollo space shuttle	○	○	○	◉	○	○
7	cats/kitten kittens tabby	○	○	○	○	◉	○
8	real time commodity future quotes	○	○	○	○	◉	○

Figure 4.13 : The Query Judgement Interface

Our users were all degree or post-graduate students from our computer science department who would have much experience in using search engines. The evaluation criteria we asked the users to evaluate the documents was into one of the following five categories:

- **Very Broad** – if the query is of a very broad nature then a very large number of relevant pages are likely to be found in response to a query and within this group of relevant pages there will be a number of possible sub-topics.

- **Broad** - If a query is of a broad nature then a lot of relevant pages are likely to be returned. These pages may span multiple sub-topics, but the query would be more focused than the Very Broad query.

- **Standard** – A standard query is neither broad nor narrow. It is a typical web query, which will return a many pages of highly scored documents from a search engine.

- **Narrow** - The query is more focused that any of the three above. A manageable set of documents is returned and these documents will only cover one main topic with possibly a very small number of sub-topics included.

- **Very Narrow** - This query type is extremely focused and unambiguous. The topic of the query is not in question and will not cause problems to a text retrieval system. You could imagine that a small number of relevant documents will be returned.

To ensure that the experiments were as accurate as possible, we did not choose the queries in a purely random manner from a web log, rather we identified the proportion of query terms that were 1, 2, 3 and more than 3 terms long from an AltaVista query log [Silverstein et al., 98], as in Figure 4.14 and randomly chose terms to match these length distributions. Users were not allowed to use a search engine when taking part in the evaluation, however on-line dictionaries could be used to help disambiguate terms with which they were not familiar.

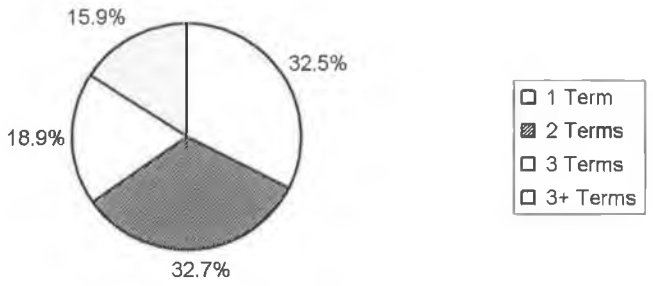


Figure 4.14 : Proportion of 1, 2, 3 and greater than 3 term queries from an AltaVista Query Log.

Once we had identified the percentage of each query length that was required, web-log queries were chosen at random from an Excite [EXCITE, 02] query log which was released in 1999 (although offensive queries were removed) to fit the distribution of terms illustrated in Figure 4.14.

Upon completion of the user evaluations, our five point scale was reduced to a three point scale, the very broad / broad queries, and the very narrow / narrow queries being combined into just broad and narrow leaving a (broad, standard and narrow) range of values. We found that users had problems in distinguishing between broad and very broad queries and similarly at the other end of the scale. Applying probability values to each of the 250 queries allowed us to identify queries that were more likely to be broad or narrow. For example, if all five users identified a particular query as being broad in focus then it has a 1.0 probability of being broad. However had one of the users evaluated the query as being narrow then the probability of the query being broad would be 0.8 and the probability of the query being narrow would be 0.2. We only examined queries that were 100% broad or 100% narrow. This resulted in us having a set of 40 broad queries and a set of 10 narrow queries available for use.

We randomly selected 10 of the broad queries and sent then to the Google search engine examining the number of scored documents returned for each query and we found that the average broad query (as identified by our users) produced a result-set of 11,633,750 documents or 0.0058% of then entire Google index. Applying the same procedure to the narrow queries we found that the average result-set was only 7,202 or 0.0000036%. This validated our belief that a narrow query is one that produces a small result set while a broad query is one that produces a larger result set.

4.3.2.2 UTILISING BROAD AND NARROW QUERIES

Based on this information we were in a position to apply a basic categorisation to each query based on the size of the result set generated[41] in a content-only phase. By comparing fraction of the total size of the dataset (n/N) where n is the size of the result set and N is the size of the entire dataset, to our findings from Google as to the proportion of documents returned in a broad query and a narrow query, we were in a position to regulate linkage weight on a sliding scale with broad queries getting maximum linkage influence (a parameter value of 50% for linkage and this resulted in 50% for content) and narrow queries getting minimum linkage influence (100% content). The formula we used to calculate the linkage influence (*linkInf*) is shown below. Letting S be the result set, N be the size of the document collection, *narrow* be the fraction of the Google index returned for an average narrow query, *broad* be the fraction of the Google index returned for a broad query and α be a constant used for normalisation of the differences in document frequency values between the Google index and our collection we have:

$$linkInf = \frac{\left(\frac{|S|}{N} \times \alpha\right) - narrow}{broad - narrow} \qquad (4.10)$$

At any point in between a query may be mapped onto the sliding scale based on the n/N figure and the corresponding weights applied to the content and linkage scores, as shown in Figure 4.15.

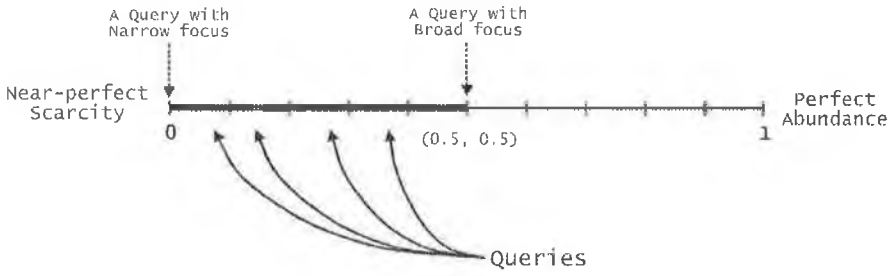


Figure 4.15 : Plotting a query onto the sliding scale

[41] Due to differences in the term distribution between Google's index and our WT10g sub collection we found that a normalization factor had to be included in the calculation procedure.

This technique is only our first attempt at dynamically regulating the influence of linkage analysis and more research would be needed to identify an optimal formula to calculate linkage and content influence. The parameters in the technique ('minimum linkage influence' and 'maximum linkage influence') should be tuned on each dataset used.

When incorporated into our linkage experiments, the retrieval process then consisted of two distinct phases:

1. A query was processed and a set of relevant documents was generated, which, in a classical retrieval system would be the ranked result returned to the user, however we used this result as the input into phase 2 of the process.
2. Based on the size of the ranked set of documents from the previous phase, the linkage weight and content weight were regulated using the scarcity-abundance technique described above and the documents ranked by using an optimal rate of linkage and content influence and returned to the user.

4.4 Our Experiments on WT_Connected

We ran many of the experiments that have been discussed in the previous chapters on the new dataset. The experiments can be divided into four distinct categories:

- Content Experiment
- Citation Ranking Experiments
- Spreading Activation Experiments
- SiteRank and PageRank Experiments.

Appendix C includes averaged results for all ten experiments that we carried out as well as detailed results from four of the experiments. All result output has been generated by TRECEVAL.

4.4.1 Content Experiment

For the content experiment we felt that continued use of Microsoft Index Server or AltaVista Discovery would hamper our attempts to evaluate our algorithms given that neither of the

two applications supported returning the relevance score of a document. Consequently we developed our own search engine for use in these experiments.

4.4.1.1 SEARCH ENGINE DESCRIPTION

The Search Engine was written in GNU C++, making use of the Standard Template Library (STL[42] [Josuttis, 99]) where feasible. The chosen platform for our Search Engine was LINUX (RedHat 7.1). The architecture of the search engine is outlined in Figure 4.16. We also refer the reader to Figure 1.8 for a more detailed overview of how the Indexer and 'Search Server' function.

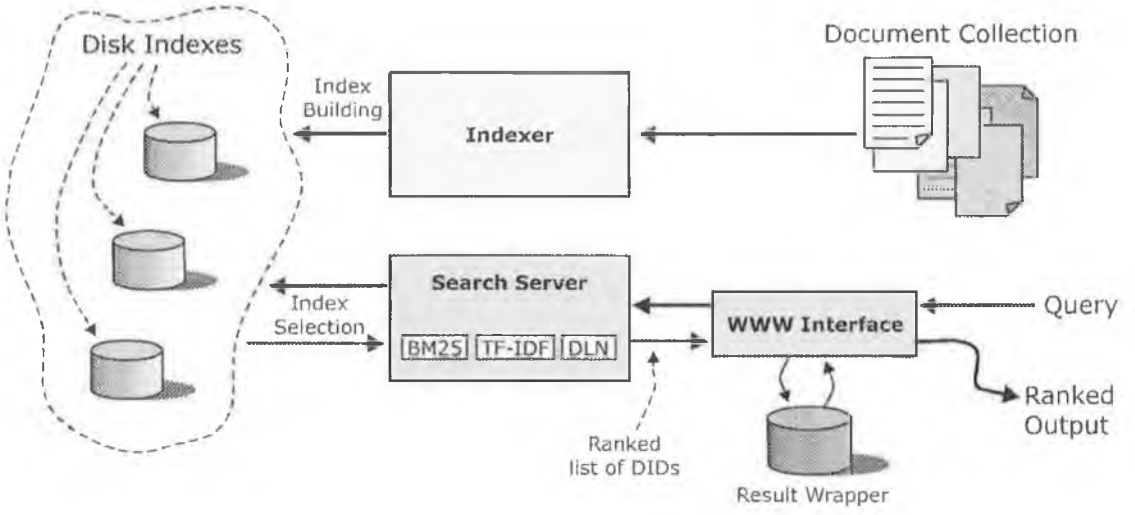


Figure 4.16 : The Outline Architecture of our Search Engine

The Search Engine can be divided into three inter-functioning components as described thus:

1. The Indexer, a software program that converts a collection of documents into an inverted index (as described in Chapter 1) and writes this inverted index out to disk for future use by the 'Search Server'. This inverted index is independent of any ranking algorithm (such as tf-idf or BM25) as only the *tf* values of any term are included in the inverted index. The reason for this is that we wished to be able to interchange ranking algorithms as required independently of the indexing process. Any number of inverted indexes may exist on disk at any one time.

[42] The STL is a library of advanced templates and functions whose purpose is to provide tried and tested implementations of common algorithms and data structures, for example, vectors, stacks, queues & maps.

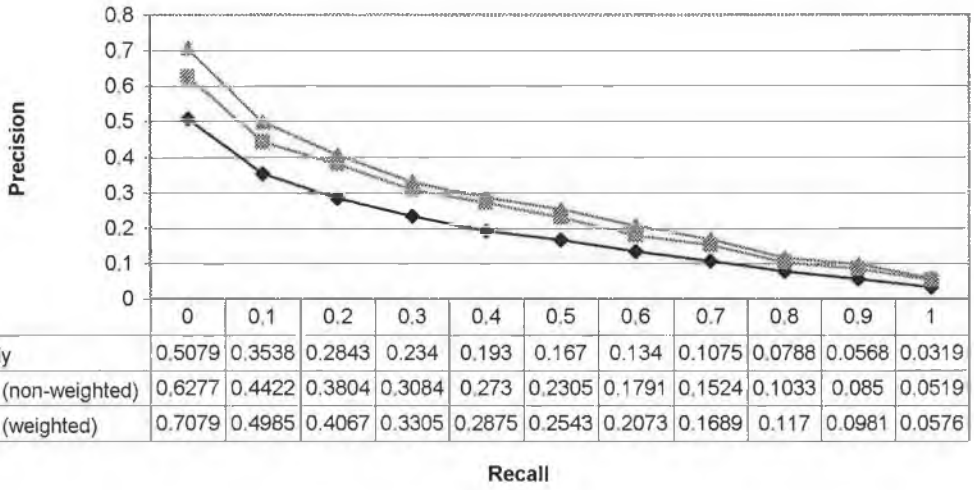
2. The 'Search Server', also a software program, provides content-based retrieval facilities using a disk based inverted index (one of possibly many) that has been previously constructed by the Indexer and these retrieval facilities are based on one of the following three algorithms:
   - tf-idf - basic tf-idf ranking using the basic tf-idf formula on page 13 of this thesis.
   - tf-idf with document length normalisation – incorporating document length normalisation based on taking the log of the *tf* value of a term within a document as in the length normalised formula (a) on page 13.
   - BM25 – this was BM25 ranking (as in the formula on page 14) based on the following parameter values which were set according to the best performance achieved on the WT2g collection from TREC-8 [Savoy et. al., 00] whereby advl = 900, b = 0.75, k_1 = 1.2 and k_3 = 1000.

   As previously mentioned the 'Search Server' operates using a disk based inverted index that is independent of any ranking algorithm. This allows us to choose both our disk index (processed dataset) and ranking algorithm when starting the server and these can be changed without having to rebuild the inverted index which allowed us to change ranking algorithms quickly and efficiently, allowing us to immediately see the effect on retrieval performance of altering the ranking algorithm. One limitation of the 'Search Server' was that it did not support Boolean queries (e.g. cat AND dog NOT horse), however we did add global Boolean support into the 'Search Server' whereby we could choose how to process queries when starting the server so that all query terms are ANDed or Ored, depending on our requirements. On start-up of the server, it reads (into RAM) the required sections of the inverted index from disk and (depending on the algorithm chosen) calculates term weights for each term in the (compressed using CRS) document-term matrix. This is done prior to accepting and processing any queries, which are processed in a sequential fashion, returning a sorted set of documents identifiers for each query.
3. The 'WWW Interface' was a developed in PHP and operated in conjunction with the Apache HTTP Server [APACHE]. This interface component accepts queries, interacts with the search server to generate a sorted result-set of scored document identifiers and then queries a second server (the 'Result Wrapper') to format the results in the manner expected from a WWW search engine. An example of the response from our search engine is shown in Figure 5.6.

The search engine just described produced the results of our content experiment as well as the scored set of documents used by our linkage algorithms for our linkage experiments. The top 1,000 ranked documents from the content experiment produced our content-only results.

4.4.1.2 DETAILS OF THE CONTENT-ONLY EXPERIMENT

For the purpose of our content-only runs we utilised the BM25 ranking algorithm with OR query handling in operation. In order to choose optimal queries for our experiments we indexed all 70,070 documents (59,720 unique documents) identified in the TREC relevance judgements for WT10g (TREC 9 queries) and evaluated three separate query generation techniques on this small set of documents, one automatic and two manual (weighted) query generation techniques and we produced the following results as shown in Figure 4.17. We wanted our queries to produce content-only runs with high retrieval performance as limiting the retrieval performance of the content-only run would give wider (and unfair) scope for the linkage algorithms to improve retrieval performance over the content-only results.



	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Title only	0.5079	0.3538	0.2843	0.234	0.193	0.167	0.134	0.1075	0.0788	0.0568	0.0319
Manual (non-weighted)	0.6277	0.4422	0.3804	0.3084	0.273	0.2305	0.1791	0.1524	0.1033	0.085	0.0519
Manual (weighted)	0.7079	0.4985	0.4067	0.3305	0.2875	0.2543	0.2073	0.1689	0.117	0.0981	0.0576

Figure 4.17 : Precision v.s. Recall curve for three alternative query generation methods.

The results of this short experiment illustrated that our manual (weighted[43]) queries outperformed both non-weighted manual queries and title only queries, therefore we ran all our subsequent content-experiments on WT_Connected using the very same manual (weighted) queries.

[43] Weighted queries refers to the fact that each query term can be weighted w.r.t. its importance to the query topic. This weighting was accomplished by increasing the term frequency of certain terms within the query.

The top 1,000 ranked documents for each query from our search engine produced the results of our content-only experiment for subsequent evaluation. The ranked set of documents from the content experiments along with their relevance weights gave us a result with which we can employ as a benchmark against which to compare the retrieval performance of our linkage-based algorithms, which we discuss in the following section. The top 2,000 ranked documents along with their relevance weights were saved for later processing by our linkage-based experiments.

4.4.2 LINKAGE EXPERIMENTS

We evaluated nine different algorithms for our linkage experiments. In a manner similar to the technique we employed for TREC-9, each experiment was based on reranking a set of 2,000 documents produced in the content-only phase utilising hyperlink information between documents in WT_Connected. Our experiments were based on citation ranking, spreading activation, PageRank and SiteRank and we also evaluated the benefit of integrating the scarcity/abundance technique for dynamically regulating the influence of linkage analysis by comparing the results to best parameter methods based on values used in the AT&T experiments for TREC-9

4.4.2.1 CITATION RANKING EXPERIMENTS

In all we evaluated four citation ranking techniques on the WT_connected dataset. The first was basic indegree ranking.

Indegree Ranking

This experiment (like all the following linkage-based experiments) was based on simply extracting the top 2,000 documents from the content-only experiments and reranking them based on the off-site indegree of each document. Let n be some web page and S_n be the set of off-site pages that link into document n we rerank by Sc'_n:

$$Sc'_n = |S_n| \qquad (4.11)$$

The results of this experiment will be referred to as '*indegree 1*'. Recall that for all the linkage experiments, although the document ranking was based on a score generated by combining both content and linkage evidence, the final score allocated to a document in the ranked output was based

on the linkage evidence and content evidence was only used to rank documents with equivalent linkage scores.

Indegree Log Weighted Ranking

In this experiment, the log of the off-site indegree of a document was used to increase the content only score by a value proportional to the content only score. Let Sc_n be the content-only score of a document and $indegree_n$ be the off-site indegree of n we rerank by Sc'_n:

$$Sc'_n = Sc_n + Sc_n \times Log(indegree_n + 1) \qquad (4.12)$$

In the results, this experiment will be referred to as '*indegree 2*'.

Best Guess Parameter Ranking using Normalised InDegree Scores

Before combining content and linkage scores for this experiment and all subsequent ones we normalised the linkage scores so that they would be in an equivalent range to the content-only scores. Once this was done we allocated scores based on the normalised indegree of the document added to the original content-only score of the document producing a new score Sc'_n for the document. Let *norm* refer to a normalised score, $indegree_n$ be the off-site indegree of n and α be a constant (value of 0.25) used to regulate the influence of linkage evidence (based on values used by AT&T in TREC-9 [Singhal & Kaszkiel, 00]), we have the following formula.

$$Sc'_n = Sc_n + (norm(indegree_n) \times \alpha) \qquad (4.13)$$

In the results, this experiment will be referred to as *indegree_3*.

Indegree Ranking using Normalised InDegree Scores incorporating the Scarcity Abundance technique

Once again we normalised the linkage scores so that they would be equivalent to the content-only scores. This experiment incorporated the scarcity-abundance technique for regulating linkage influence described above to dynamically regulate the influence of both the normalised indegree score as well as the content score. Letting *linkInf* be the dynamically generated regulator of the linkage influence generated using the Scarcity-Abundance technique discussed earlier and $indegree_n$ be the off-site indegree of n, we have the following formula.

$$Sc'_n = (Sc_n \times (1 - linkInf)) + (norm(indegree_n) \times linkInf) \qquad (4.14)$$

In the results, this experiment will be referred to as *indegree_4* and this allowed us to directly compare the benefit of the scarcity – abundance technique to best guess parameters.

4.4.2.2 SPREADING ACTIVATION EXPERIMENT

Recall from the previous chapter that spreading activation refers to a technique that propagates numerical values (or activation levels) among the connected nodes of a graph. In the context of this experiment it facilitates a document transferring its content-only score across its out-links. The process we have implemented for these experiments is thus; for each document in the result-set we identify what documents comprise the in-set for the document and propagate their scores along all out-links equally, with a fraction of the score being propagated to the current document. If one of the inset documents is not part of the result-set then its score is 0.0 and will not have any positive effect on the overall document ranking. The formula for calculating each document score is shown below. Let S be the relevant set of documents and S_n be the in-set of n, therefore:

$$Sc'_n = Sc_n + \sum_{m \in S_n} \frac{Sc_m}{outdegree_m} \qquad for(m \in S) \qquad (4.15)$$

This experiment we shall refer to in our evaluation as *SpreadAct*. Finally, we carried out our own evaluation of a SiteRank algorithm along with an implementation of PageRank.

4.4.2.3 SITERANK AND PAGERANK EXPERIMENTS

In this set of experiments we evaluated both SiteRank and PageRank twice, once with best guess parameter values (once again based on values used by AT&T in TREC-9) to regulate the influence of the SiteRank and PageRank scores and subsequently using the scarcity/abundance technique. The SiteRank algorithm was as described earlier in this chapter and the Pagerank algorithm was described both in this chapter and Chapter 2. We ran the iterative process to calculate both PageRank scores and SiteRank Scores over twenty iterations (enough to ensure convergence) to produce the SiteRank and PageRank scores for all 120,494 documents

Both SiteRank and PageRank scores were calculated in the one processing run and the source code was developed in JAVA and executed on a Pentium III Zeon processor running Windows NT4 with 512MB of installed RAM. Connectivity data was served by a Microsoft SQL

Server 7 database, which was installed on the same computer that ran the calculation process. The total process required 55MB of RAM and the twenty iterations required 65 minutes of CPU time.

In our best-guess parameter experiments, the formula used to calculate the final score for a document is based on the following formula for SiteRank and a similar formula for PageRank.

$$Sc'_n = Sc_n + (SR_n \times 0.25) \qquad (4.16)$$

The results of the best-guess parameter experiments for both SiteRank and PageRank will be referred to as *SiteRank_param* and *PageRank_param*. When incorporating the scarcity/abundance technique the formulae used to calculate the final score for a document is as follows and once again a similar formula for PageRank.

$$Sc'_n = (Sc_n \times (1 - linkInf)) + (SR_n \times linkInf) \qquad (4.17)$$

The results of the experiments for both SiteRank and PageRank (incorporating the scarcity/abundance technique) and these will be referred to as *SiteRank_query* and *PageRank_query* respectively.

4.4.3 Results of the Experiments

As mentioned previously our linkage experiments were based on reranking content-only results and this gave us the ability to compare the linkage results with the content-only results. As can be seen from Figure 4.18 and Table 4.3 seven of our experiments produced very similar results. On closer inspection we see that some of the linkage experiments actually achieved small improvements in precision (shown as bold in Table 4.3) over the content-only runs when examined at 5 and in some cases 10 and larger document retrieval points. This is encouraging because up until we ran these experiments, TREC participants were unable to illustrate any improvement in retrieval performance when using WT10g data, and although WT_Connected is not the same dataset as WT10g, we were using a subset of both the dataset and the relevance judgements so we were following the TREC procedure for evaluation of IR systems, which now can be shown to produce results which at least are not all negative.

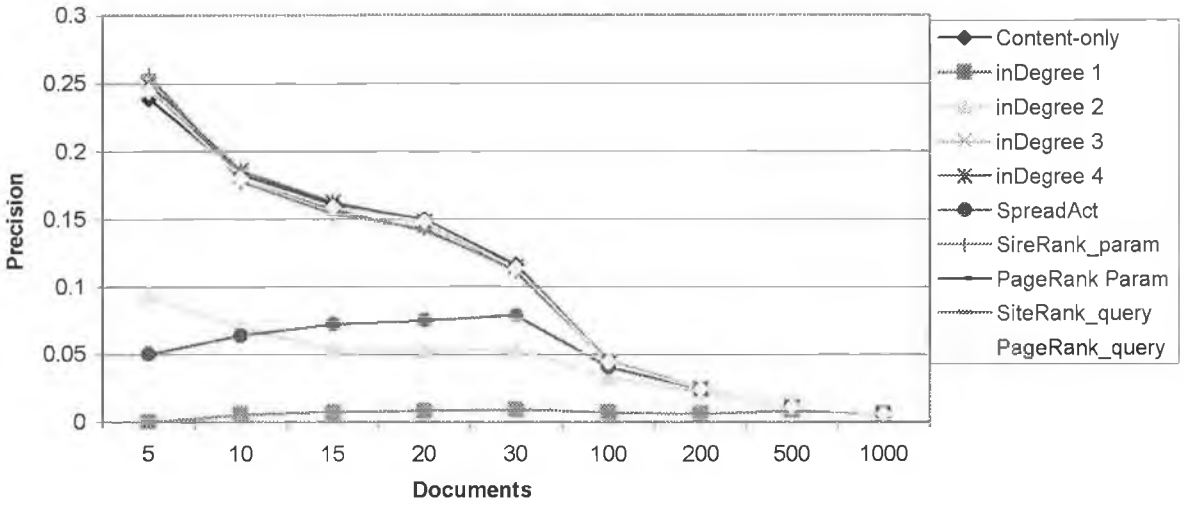


Figure 4.18 : Precision at standard levels of documents

Below we illustrate the data from Figure 4.18 in tabular form with improvements in precision over the content-only experiment highlighted using bold text.

	CONTENT-ONLY	INDEGREE 1	INDEGREE 2	INDEGREE 3	INDEGREE 4	SPREADACT	SITERANK_PARAM	PAGERANK PARAM	SITERANK_QUERY	PAGERANK_QUERY
5	0.2389	0.0	0.0944	**0.2444**	**0.25**	0.05	**0.2556**	**0.2444**	**0.2556**	**0.2444**
10	0.1833	0.0056	0.0694	0.1833	**0.1861**	0.0639	0.1778	0.1833	0.1806	0.1806
15	0.1611	0.0074	0.0537	**0.163**	**0.163**	0.0722	0.1537	**0.163**	0.1574	0.1593
20	0.15	0.0083	0.0528	0.1472	0.1486	0.075	0.1431	0.1486	0.1417	0.1486
30	0.1167	0.0093	0.0537	0.1148	0.1148	0.0787	0.112	0.113	0.1139	0.1139
100	0.0444	0.0072	0.0331	0.0444	0.0442	0.0406	**0.0447**	0.0444	**0.045**	0.0444
200	0.0246	0.0061	0.0222	**0.0247**	**0.0249**	0.0244	**0.025**	0.0246	**0.0249**	0.0246
500	0.0114	0.0085	0.0109	0.0114	0.0114	0.0114	0.0113	0.0114	0.0113	0.0114
1000	0.0061	0.0061	0.0061	0.0061	0.0061	0.0061	0.0061	0.0061	0.0061	0.0061

Table 4.3 : Precision values for the Experiments on WT_Connected

Both methods of combining linkage information with the content information for both SiteRank and PageRank algorithms illustrate improvements in precision at 5 documents of almost 7%, but an overall decline in precision is shown at 10 documents in all but the PageRank experiment, which has precision values equal to the content-only experiment. The best experiment overall would be the *SiteRank_query* experiment which attains joint-highest precision at 5 documents and second highest at 10 documents. The best indegree based experiment is the *'indegree 4'* experiment (equation 4.14) which ranks based on the normalised off-site indegree of a document combined with the content-only score using the scarcity-abundance technique. This experiment outperforms the *content-only* experiment at all levels up to and including 15 documents.

A number of experiments produced quite disappointing results, but this is not entirely unexpected. The '*indegree 1*' experiment (equation 4.11) allows all 2,000 documents to be ranked based on no evidence other than off-site indegree alone. In our previous TREC-9 experiments we would have limited the number of documents that were reranked by such a technique to a small number of top ranked documents and in this way we helped keep the precision values of the experiment higher, but in these experiments no such limitation was incorporated. The same caveat applies to the other low scoring experiments '*inDegree 2*' (equation 4.12) and *spreadAct* (equation 4.15).

If we examine the comparison between using best guess parameters ('*inDegree 3*', equation 4.13) and the scarcity-abundance technique ('*inDegree 4*', equation 4.14) when ranking based on normalised off-site indegree scores we see that the scarcity-abundance technique outperforms the best-guess parameters (shown in bold in Table 4.4) at 5 and 10 documents, and is at least equal to the best-guess parameter values at all levels until 30 documents. However, the performance improvement is only just over 2%, which is not large enough to prove the benefit of the scarcity-abundance technique.

	CONTENT-ONLY	INDEGREE 3	INDEGREE 4
5	0.2389	0.2444	**0.25**
10	0.1833	0.1833	**0.1861**
15	0.1611	0.163	**0.163**
20	**0.15**	0.1472	0.1486
30	0.1167	0.1148	**0.1148**
100	0.0444	0.0444	**0.0442**
200	0.0246	0.0247	**0.0249**
500	0.0114	0.0114	0.0114
1000	0.0061	0.0061	0.0061

Table 4.4 : Comparing Precision values for the best guess and scarcity-abundance technique

The Precision results from Table 4.4 are plotted in Figure 4.19.

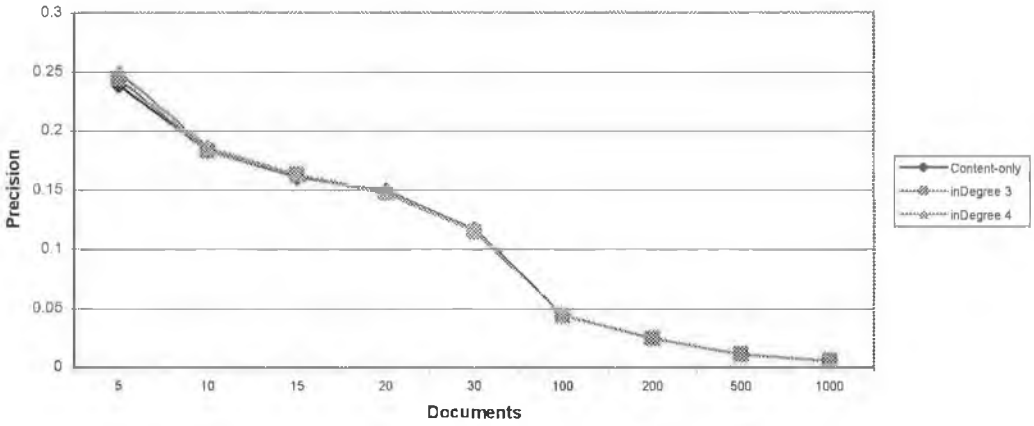


Figure 4.19 : Comparing best-guess parameter and scarcity/abundance technique of combining scores

So how did SiteRank compare to PageRank? At document retrieval point 5, SiteRank outperformed PageRank but the results were mixed at 10 documents and beyond as can be seen from Figure 4.20. It is encouraging to see that both SiteRank and PageRank marginally outperformed the content-only experiment at 5 documents. Had the dataset contained a larger density of relevant documents then perhaps our results could have been more positive.

	CONTENT-ONLY	SITERANK_PARAM	PAGERANK_PARAM	SITERANK_QUERY	PAGERANK_QUERY
5	0.2389	**0.2556**	0.2444	**0.2556**	0.2444
10	0.1833	0.1778	0.1833	0.1806	0.1806
15	0.1611	0.1537	0.163	0.1574	0.1593
20	0.15	0.1431	0.1486	0.1417	0.1486
30	0.1167	0.112	0.113	0.1139	0.1139
100	0.0444	**0.0447**	0.0444	**0.045**	0.0444
200	0.0246	**0.025**	0.0246	**0.0249**	0.0246
500	0.0114	0.0113	0.0114	0.0113	0.0114
1000	0.0061	0.0061	0.0061	0.0061	0.0061

Table 4.5 : Comparing Precision values for the PageRank and SiteRank experiments

The Precision results from Table 4.5 are plotted in Figure 4.20.

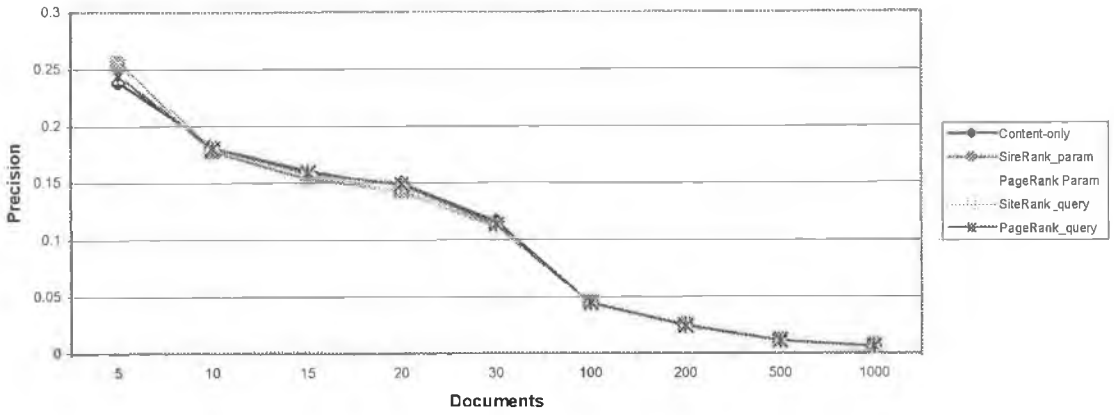


Figure 4.20 : Examining the results attained by SiteRank and PageRank

If we compare the average precision figures for all these experimental runs as shown in Figure 4.17, SiteRank outperforms PageRank for both best guess experiments and the scarcity-abundance technique. However, the average precision of the *content-only* experiment is marginally higher than the SiteRank and PageRank experiments. The only other experiments with notable average precision values are '*indegree 3*' (equation 4.13) and '*indegree 4*'(equation 4.14) which are similar, although a little lower than the SiteRank scores.

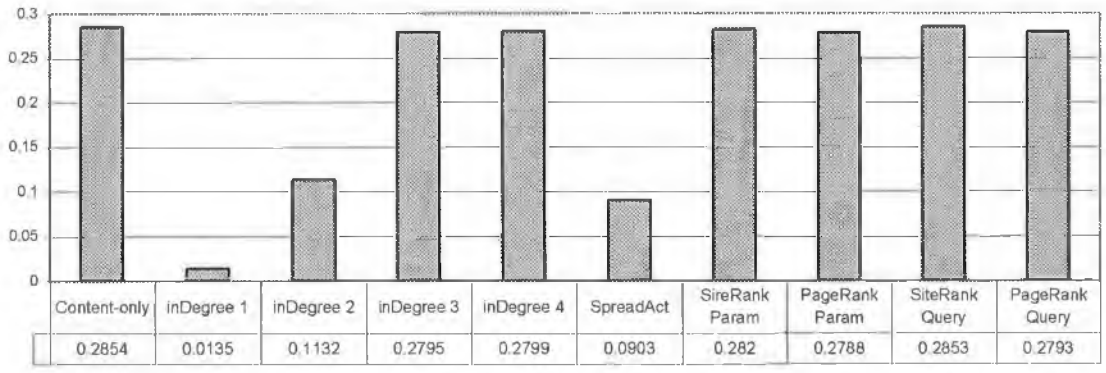
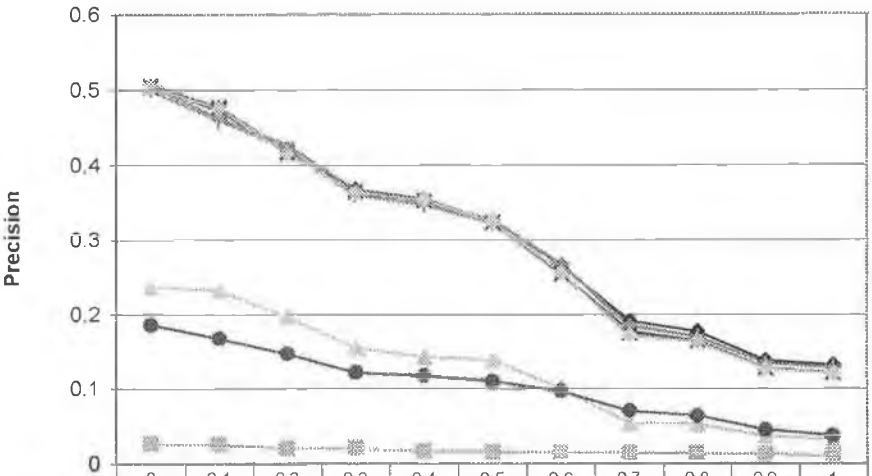


Figure 4.21 : Average Precision of all experiments on WT_Connected

Finally, the precision v.s. recall curve for the experiments is shown in Figure 4.22. Based on the results illustrated in the previous pages there are no surprising findings.



	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Content-only	0.505	0.477	0.4171	0.3677	0.3548	0.3244	0.2654	0.1908	0.1767	0.1384	0.1316
inDegree 1	0.027	0.0258	0.0209	0.0208	0.0166	0.0163	0.0152	0.0144	0.0141	0.0129	0.0107
inDegree 2	0.2362	0.2325	0.1966	0.155	0.143	0.1388	0.1005	0.0541	0.052	0.037	0.0327
inDegree 3	0.5042	0.4737	0.4164	0.3655	0.3528	0.3236	0.2542	0.1747	0.1645	0.1277	0.121
inDegree 4	0.5057	0.4776	0.4178	0.3648	0.352	0.3228	0.2553	0.1746	0.1647	0.128	0.1214
SpreadAct	0.1857	0.1672	0.147	0.1219	0.1178	0.1106	0.0973	0.0707	0.0644	0.0452	0.0377
SireRank_param	0.4994	0.4597	0.4218	0.3599	0.3467	0.3224	0.2665	0.1838	0.1699	0.1347	0.129
PageRank Param	0.5037	0.4755	0.4161	0.362	0.3526	0.3226	0.2546	0.1773	0.1636	0.1272	0.1205
SiteRank_query	0.505	0.4635	0.4274	0.3676	0.3501	0.3269	0.2669	0.186	0.1699	0.1342	0.1285
PageRank_query	0.5036	0.4759	0.4161	0.3634	0.3538	0.3238	0.2558	0.1716	0.1629	0.1272	0.1206

Recall

Figure 4.22 : Precision – Recall curve of all experiments on WT_Connected.

4.5 Conclusions to be drawn from these Experiments

In this chapter we have shown that marginal improvements in retrieval performance can be obtained when the dataset used has a higher link density and therefore is better capable of supporting experiments into linkage analysis and when linkage information is used in a version of PageRank. Recall from the previous chapter that our findings for TREC-9 illustrate that similar experiments on WT10g yielded no improvement in retrieval performance, rather a dis-improvement, and this finding was shared by all other participating groups. While the results are far from ideal, it does seem as if increasing the density of off-site links within a dataset beyond that contained within WT10g will better support experiments into linkage based IR. Recall that 25.9% of documents in WT_Connected contain a non-zero off-site indegree and compare this to only 1.8% of documents in the original WT10g dataset.

Therefore, we conclude that the lack of a representative dataset is the major factor holding back successful evaluation of linkage analysis algorithms. However, we do not know for definite how representative WT_Connected (or WT10g for that matter) is of the WWW as a whole. The obvious way to compare WT_Connected to the WWW was to develop a web crawler and do some explorative crawls of the WWW. If WT_Connected is found to be underestimating the link density of the WWW then we could hope for more notable improvements in retrieval performance using a new dataset of real-world WWW data. The next chapter will describe some of the crawls of web data that we made to examine the structure of WWW documents.

4.6 SUMMARY

In this chapter, we described the SiteRank linkage-based retrieval algorithm which was based on PageRank, but propagated rank between web pages by taking a website-centric view of the process. Given that SiteRank is based on PageRank, we provided an in-depth examination of PageRank algorithm and identified two notable problems with PageRank as it was described in Chapter 2. These problems are:

- Dangling Links, which are links that point to a page that contain no out-links, hence it is not known where the rank of the page with no links should be distributed.

- Rank Sinks, which are loops in the linkage structure of web pages, creating what can be described as a 'black hole' into which rank can enter, but never exit.

The conventional PageRank algorithm solves these two issues, but it is still possible to artificially increase the rank of a web page by creating a clever synthetic linkage structure surrounding the page. The SiteRank algorithm that we describe in detail avoids this problem by limiting the influence of web pages from any one site, thus making it more difficult and expensive to artificially increase a web page's rank in this manner.

However, the problem of how to evaluate linkage-based algorithms remains. We have seen in Chapter 3 that the TREC web track has not yet demonstrated any improvement in retrieval performance using standard evaluation techniques of linkage-based retrieval, so we extracted a densely linked subset from WT10g called WT_Connected. WT_Connected maximised the density of off-site

links, as much as we could given that WT10g was our source. A number of experiments based on our previous TREC experiments (for both TREC-8 and TREC 9) and SiteRank and PageRank were executed on this new dataset. In addition, we evaluated a new method of combining linkage and content evidence together to produce a final ranking. Prior to this the most widely used method was to incorporate best guess parameters into the process, or some other technique based on the number of terms in the query. Our technique was based on the size of the result-set of highly scored documents.

Our findings show that it is possible to gain moderate improvements in retrieval performance when running experiments using standard TREC evaluation procedures and measurements on WT_Connected as opposed to WT10g. The question that remains is how representative was WT_Connected of the true linkage structure of live WWW data, this was the issue we tackled in the following chapter.

Chapter 5

UTILISING WEB CRAWLERS TO EXPLORE THE LINKAGE STRUCTURE OF LIVE WEB DATA

We begin this chapter with a discussion of web crawlers and some of the issues involved in managing a web crawler that gathers live WWW data. We then discuss three separate crawls of WWW data that we made. Two of the crawls were conventional crawls, each of which implemented a different queuing algorithm and the third crawl was an Irish language specific crawl. We were hoping to come to some firm conclusions as to the mature of the linkage structure of the WWW and if it is possible to crawl a dataset which would support faithful experiments into linkage-based retrieval of web documents.

5.1 An Introduction to Web Crawlers

In the previous chapters, we highlighted the limited linkage structure of the WT10g experimental test collection and the consequent failure of any participating group in any of the conventional TREC experiments to enhance retrieval performance by incorporating linkage analysis. Surprised by this we extracted a densely linked subset called WT_Connected. Our results from WT_Connected, presented in Chapter 4, seemed promising but we are not sure if WT_Connected accurately reflects the linkage structure of the actual WWW, and whether a dataset similar in structure could be generated by sending out a web crawler to gather documents. To examine if this was the case we built a web crawler[44], generated three datasets of WWW documents, each using a different crawling strategy, and compared these to WT_Connected and to WT10g.

We have briefly introduced web crawlers in Chapter 2 and we know that a web crawler is a software tool that gathers web pages from the WWW, usually for the purpose of examining the nature of, or providing content-retrieval facilities over, web pages. Here we will describe the architecture of our web crawler that we have developed to support our experiments into WWW structure.

[44] Web Crawlers may also be known as robots, bots, spiders or gatherers. In this dissertation they will be referred to as web crawlers.

5.1.1 A Basic Crawling Algorithm

The basic algorithm executed by any web crawler begins with a list of seed URLs as its input or starting point and repeatedly executes the following steps:

1. Remove a URL from the URL queue using some predefined technique and download the corresponding document, adding the URL of the downloaded document to a list of previously seen URLs.

2. Any links contained in the downloaded document are extracted, the associated URL translated from a relative URL into an absolute URL (if necessary) and added to the list of URLs to download (the queue), provided it is not on a list of visited URLs or is not already on the queue. If required, some additional processing may take place for each document, which would involve processing the downloaded document in other ways. For example the document may have its textual content prepared (stemmed and stopwords removed) for subsequent indexing and retrieval. If the linkage data is being stored as the web crawler traverses the web (which is the case with our crawls), all links from the downloaded document when extracted will be stored as source and target document ID pairs, at the very least, for future reference.

3. Goto step 1 until the required number of documents have been downloaded or until the queue is empty.

Any web crawler must have a number of documents on the queue prior to beginning the crawling process. These URLs are referred to as seed URLs. These URLs may be a handcrafted list of URLs, chosen for a particular reason, such as having a high off-site outdegree or they may be based on URLs discovered during a previous crawl. In our experiments, all our seed URLs were handcrafted based on their out-link structure, their popularity, or in one case their language.

5.1.2 WEB CRAWLER ARCHITECTURE

An overview architecture diagram of our web crawler is shown in Figure 5.1.

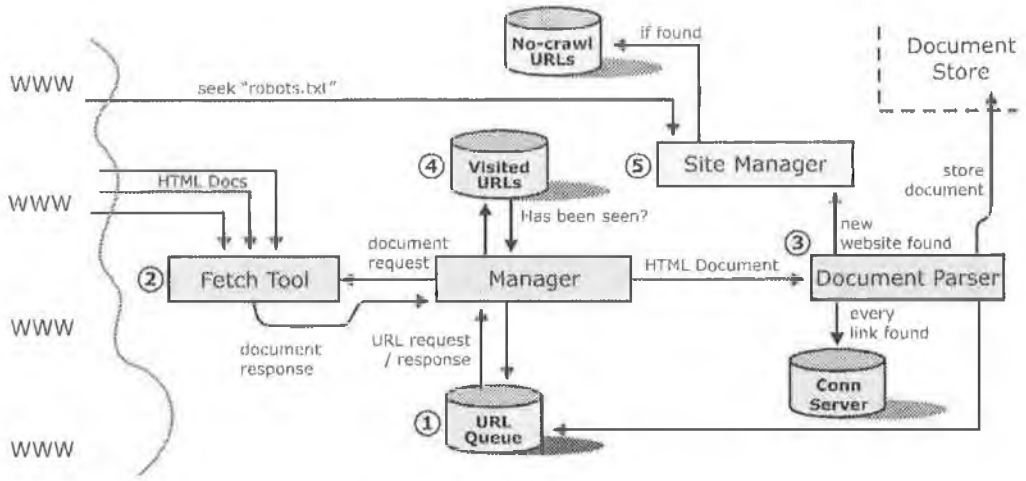


Figure 5.1 : Overview architecture of our Web Crawler

This web crawler required a number of functional components:

- a component called a 'URL Queue' for storing the list of URLs to download (1);
- a component called the 'Fetch Tool' for downloading documents using a transfer protocol (2);
- a component for extracting links from HTML documents (3). This component, the 'Document Parser', extracts other required information from each downloaded document and passes the link information to a connectivity server, it also adds any found URLs to the queue (if they have not been added once before), identifies new hosts (websites) to the 'Site Manager' and it also writes out the document (in a number of different representations) to disk for subsequent indexing (had we deemed such indexing necessary);
- a component called 'Visited URLs' for determining if a URL has already been encountered (4); and

- a component for determining whether a URL should be crawled or not based on the robots exclusion protocol (5) called the 'Site Manager'.

All these components are integrated to function together seamlessly by a 'Manager' component that passes messages and data between all other components. Our crawler was single-threaded, was written in JAVA and could process over 10,000 URLs per day, although when robots exclusion (described below) support was operational, this figure dropped to below 5,000 per day. The primary reason for this was relatively slow download rates coupled with the fact that the crawler was single threaded and consequently had to stop crawling HTML documents while it was examining websites for robots exclusion data. It was for this reason that we limited the sizes of our two large crawls to the sizes that we discuss in this chapter.

5.1.3 Robots Exclusion Protocol

The robots exclusion protocol [ROBOTS, 02] is a standard that allows website administrators to indicate to visiting crawlers which parts of their site should not be visited by the crawler. All 'good' crawlers will adhere to the standard as it is of benefit to both the crawler mangers and the website administrators in that neither will want certain sections of websites (such as user access logs) to be crawled and secondly, it is a matter of good netiquette[45]. A Website administrator may even specify different limitations for different web crawlers, and crawlers that have caused problems in the past may even be requested not to access to the site at all.

The implementation of the standard simply relies on storing all relevant data in a text file called robots.txt at the root directory of a website. However, adhering to the robots exclusion standard makes a crawler less efficient because (in our case) every new site found requires seeking out a robots.txt file, parsing it and adding the relevant sections to a list of not to be crawled URL roots. Conforming to, and abiding by, the netiquette protocol of robots exclusion is not mandatory (or enforceable) and relies upon the good citizenship of a web crawler's author. However, obeying the robots exclusion standard is for the common good of net citizens and is something that all the major search engine developers will abide by.

[45] Netiquette - "Netiquette" is network etiquette and refers to both common courtesy online and the informal "rules of the road" of cyberspace.

5.1.4 THE URL QUEUE

One of the other essential features of a web crawler is the URL queue, which stores a list of all URLs that the crawler knows about but has not yet downloaded. The URL queue is a data structure that regulates the order in which the web crawler downloads documents. All URLs parsed from downloaded documents are enqueued (pushed onto the queue), unless they have already been visited or are already on the queue, and they can be dequeued (removed from the queue) in any order. This order is determined by the queuing algorithm. The queuing algorithm is extremely important as it will determine the behavior of the web crawler, i.e. if it crawls mostly within websites or jumps across website boundaries whenever possible.

A basic queuing algorithm would work much like a FIFO queue in that the URLs are dequeued in the order they were enqueued. However, it is considered unacceptable (netiquette) to have multiple HTTP requests seeking to download web pages being sent in a sequential fashion to the same HTTP (web) server due to the potential this causes for web server load problems. This is most likely to occur when using a FIFO queue. Consequently, benefits can be gained from incorporating some form of randomness or logic rules, which influence the how URLs are dequeued.

A more advanced queuing algorithm would be a weighted queuing algorithm (sometimes referred to as a priority queue), in which weights are assigned to each URL and the top weighted URL is dequeued as required. The weights assigned to a URL may be based on any number of factors such as the number of hyperlinks pointing at the URL or even a PageRank value for a URL, or perhaps based on the linkage weight of the documents from which the URL has been parsed using a spreading activation technique.

In our web crawler, we varied the queuing algorithm between crawls depending on how we wished the crawler to behave. For example, a web crawler which was to crawl the WWW in sequential fashion without any regard for which documents to download next would use a FIFO (or an ageing) queue. However, were the crawler to be required to gather documents from as many hosts as possible, a weighted queue would be required which would allocate a high weight to a URL on the queue if the link to that URL crossed website boundaries and allocate an even higher weight if the target URL was from a host never before seen by the web crawler.

5.1.4.1 Our Crawler's URL Queue

Table 5.1 shows an extract from one of our URL queues. As we can see our queue consists of much more than simply a list of URLs.

URL ID	URL	WEIGHT	STATUS	CITATIONS
752	http://www.thedaily.com/alist.html	3	Q	1
753	http://infoseek.go.com/Topic/News	26	Q	1
754	http://www.tvguide.com	49	D	22
755	http://www.gist.com	31	D	4
756	http://www.filmland.com/boxoffice	27	Q	1
757	http://www.eonline.com	40	D	13
758	http://www.imdb.com	118	D	91
759	http://www.film.com	43	D	16
760	http://www.broadcast.com	47	D	20
761	http://www.radio-locator.com	31	D	4

Table 5.1 : Ten documents from a URL Queue associated with one of our crawls.

Each URL has the following information associated with it:

- The actual URL plus a unique identifier for the URL, which becomes the document ID once the document has been downloaded.

- A weight that regulates the order in which the URLs are dequeued. As documents are downloaded the weights of these documents may be increased, for example, if another link was found to one of the URLs on the queue or the weight may be decreased if there are problems downloading the document.

- A status flag, which identifies whether the URL is still on the queue ('Q'), dequeued and downloaded ('D') or removed from the queue due to error ('E').

- A citation count, which counts the number of citations into each document. Note that the number of citations is not the same as the weight value. The weight is based on additional factors rather than just pure citation counting.

Our URL queue was initially planned to be stored in RAM, however it soon became obvious this would consume large amounts of RAM because the queues we managed very quickly grew to hold many millions of URLs (and associated weights). Consequently, we employed a SQL Server database to manage our URL queue, which allowed us to integrate weighting logic into an SQL stored procedure, an example of which is shown in Figure 5.2.

```
CREATE PROCEDURE insertUrl
(
    @new_url  [varchar](256),
    @url_score  [int])
    AS
       DECLARE @res integer
       SELECT @res = idx FROM urlQueue WHERE url = @new_url
       if @exists is NULL
       BEGIN
         INSERT INTO [urlQueue].[dbo].[urlQueue]
         (
             [url],
             [score]
         )
         VALUES
         (
             @new_url,
             @url_score
         )
       END
       else
       BEGIN
         update urlQueue SET score = score + 1 WHERE idx = @exists
         update urlQueue SET citations = citations + 1 WHERE idx = @exists
       END
)
```

Figure 5.2 : Stored Procedure to manage URL Queue.

The operation of this stored procedure is quite straightforward. A URL (@new_url) and a score for that URL (@url_score) are firstly passed into the stored procedure as parameters. The database is queried to validate if the URL is already on the queue. If the URL is not found on the queue (at a list of visited URLs), the URL is added into the queue and this automatically generates a unique identification number (idx) for that URL as well as setting the status of that URL by default to 'Q'. If, however, the URL is found on the queue, the unique identification number for that URL is retrieved (@exists) and the score and citation count for that URL (identified by the unique identifier) is incremented.

5.1.5 Development Issues

Developing, testing and running a web crawler such as the one we have described is certainly not a trivial task. Aside altogether from the problems of writing code capable of processing mal-formed web data (which we have not discussed), there are matters of netiquette involved also. Consequently, we integrated some rules into our crawler, as follows:

- A crawler must never request large numbers of documents from the same host sequentially. Every effort must be made to change the target host as often as is feasible.
- A crawler must never (for whatever reason) repeatedly request the same document. If a document is unavailable, its position in the queue must be penalised to such an extent that it no longer resides at (or very near to) the top of the queue. This obviously requires a weighted queue such as the one shown in Table 5.1. Repeated failures must be taken into account and the document flagged as unavailable and taken off the queue, i.e. given a status of 'E'.
- A crawler must respect a web site manager's wishes as expressed using the robots exclusion protocol. We have discussed robots exclusion previously.

5.1.6 Web Crawler Output

Although our crawler's primary function was to investigate the linkage structure of the WWW, we did store information from each document as it was downloaded. This data is outlined in Figure 5.3.

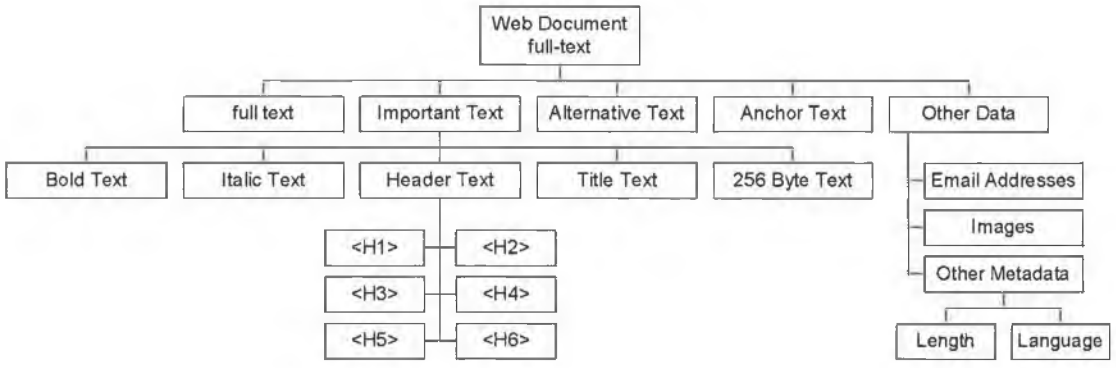


Figure 5.3 : Document subsections of crawled data

Storing this much information required that it be stored as part of eight different disk files for each web page downloaded. For example, any web page will be represented by the following files:

- The web document itself.
- A version of the HTML file with a cache header and <base href=> tag included for evaluation purposes and to act as a cache copy facility, were we to have required this.
- A cleaned document file containing a single space delimited listing of all non-HTML mark-up words in the web page with all stopwords removed.
- A bold text file, which was a space delimited listing of all **bold** text extracted from a document.
- An italic text file, which was a space delimited listing of all *italic* text extracted from a document.
- A header text file, a space delimited listing of all header text, separated into H1-H6.
- An alternative image text file, a space delimited listing of all alt text[46] contained within a document.
- And finally an 'important text' file containing a space delimited concatenation of each of the previous four files to support searching of only 'important text', were we to have required this.

In addition, we thought it prudent to store a listing of all images and email addresses found on these pages in case we were to implement a Google style image-search facility or an e-mail address search facility.

5.2 Examining the Data generated from our Crawls

Our goal in crawling live WWW data was to generate three different datasets using three different crawling strategies and to compare the structure of WT_Connected to each of the crawled datasets. Two crawls were conventional crawls of WWW data, each of which implemented a different queuing algorithm and one other crawl which was limited to only gathering documents that were

[46] Alt text refers to the alternative description for an image which HTML supports and will display is the image does not load or while the image is loading. This alternative description becomes important if one is using a web browser with image support turned-off or if network latency means long waits for images to download.

identified as being written in the Irish language. The reason for the Irish language crawl was because we felt documents written in a minority language would be more densely inter-linked than would be a more conventional subsection of the web and therefore, may provide an alternative to other (more conventional) web data when constructing a dataset to support experiments into linkage-based retrieval.

We begin by examining the results of the Irish language crawl before we turn our attention to the two other crawls.

5.2.1 GAEILGE, THE IRISH LANGUAGE CRAWL

As mentioned earlier, the Irish language crawl would allow us to examine the density of links between documents written in a minority language on the WWW. We will refer to this crawl as Gaeilge[47] throughout the remainder of this dissertation.

Crawling data in just one language requires a language identification tool, which we developed. This tool operated based on the frequency of occurrence of the most commonly occurring short terms and trigrams [Grefenstette, 97]. Evaluation of the performance of our language identifier over 100 pages randomly chosen from the dataset shows that it correctly guessed the language of web pages 98% of the time. While we are aware that this figure seems very high, the crawler only followed links from a page that was identified as being Irish and this would undoubtedly have helped the reduce the rate of error. In addition, many Irish language documents contain text written in both Irish and English, which will further aid the identification process because even a document containing a small proportion of Irish will be accepted as containing Irish. We have included a brief description of our language identification process in Appendix A.

5.2.1.1 QUEUE DETAILS

The queue that was implemented for generating the language-specific dataset was an ageing weighted queue in which each document on the queue has an associated score, a score that is used to rank documents on the queue in order to identify the next document to be dequeued. The ageing aspect of the queue refers to the fact that each time a document is dequeued, all previously existing

[47] Gaeilge is the Irish language word for 'Irish'

documents on the queue are aged by one unit, and the oldest document on the queue is the next to be dequeued, thereby ensuring that no document remains on the queue indefinitely.

In addition there was a distinction made between documents from previously unseen websites and websites from which documents have already been enqueued. If a document was from a previously unvisited website then that document is given a higher score than a document from a previously visited website. In addition, measures based in the addition of a small random value were taken to avoid the scenario where a number of documents from one site were requested from the server in sequence. Finally, if a URL that is on the queue is cited by a newly downloaded document, then this URL has its score increased so that it stands a higher chance of being downloaded earlier.

Assuming a downloaded document was identified as being in Irish, it was parsed and then all links found were enqueued. However, if a downloaded document was identified as being English or some other language, it was discarded and the next top weighted document on the queue was dequeued and crawled.

5.2.1.2 Seed URLs

As we have seen earlier, each web crawler must have a set of seed URLs from which to begin its crawl. Prior to crawling, a number of handpicked Irish language pages (twenty) were chosen to act as the seed URLs, but it soon became apparent that these seed URLs were inadequate. Given the fact that the crawler only followed links from Irish language documents and discarded all documents in other languages, the crawler quickly ran out of documents on the queue having only downloaded a few thousand documents. A larger start-set was required. We generated this by following the process outlined in these three steps:

- Select a number of (20) unique to the language frequently occurring terms.
- For each term, query a search engine with wide coverage (Google) and parse out all the documents from the top 1,000 results and add to a candidate queue.
- Remove duplicate URLs from the candidate queue and examine the remaining URLs to remove obvious erroneous URLs in order to form a starting set of URLs.

Using this approach, we were able to generate a starting set of 8,322 Irish language documents, each of which was enqueued with a high weighting. It is possible that some non-Irish web

pages would have slipped through as a result of mis-spelling or the presence of the language specific commonly occurring terms in other languages, but the language identification tool should identify such a page as not Irish and it will not form part of the dataset. For example, the Irish term for 'and' is 'agus' and this is also a frequently occurring term on Italian web pages.

5.2.1.3 Statistics of Irish Language Crawl

This crawl, which was run in July 2001, generated 26,998 documents that were successfully downloaded and classified as being Irish. We estimate that there are at least the same number of documents on a mailing list archive that the crawler found, but we were disallowed access to these documents based on robots exclusion and there are likely many documents that our crawler didn't find because our crawler only parsed links from Irish language pages. The documents that comprise the 26,998-document set came from 846 unique web sites and these sites from 29 top-level domains. Table 5.2 presents a summary of the crawled data.

Number of Documents	26,798
Number of Servers	846
Average Documents per Server	32
Number of Documents left on Queue	0
Number of Servers left on Queue	0
Average number of images per Document	13
Average number of terms per Document	554
Average length (in bytes)of text in each Document	4,021
Average length (in bytes) of each Document	13,663
Number of Seed URLs	8,322

Table 5.2 : Statistics of the Irish language crawl

Interestingly more .com domains featured than .ie domains in this dataset (see Figure 5.4). This has issues for improved language focused web crawlers. Even were we in a position to have the resources to gather even more Irish language pages by downloading all .ie websites and then processing all documents to identify only Irish documents, we have illustrated that this is not a viable option as we will not be able to gather all (or even a large percentage of) Irish language content.

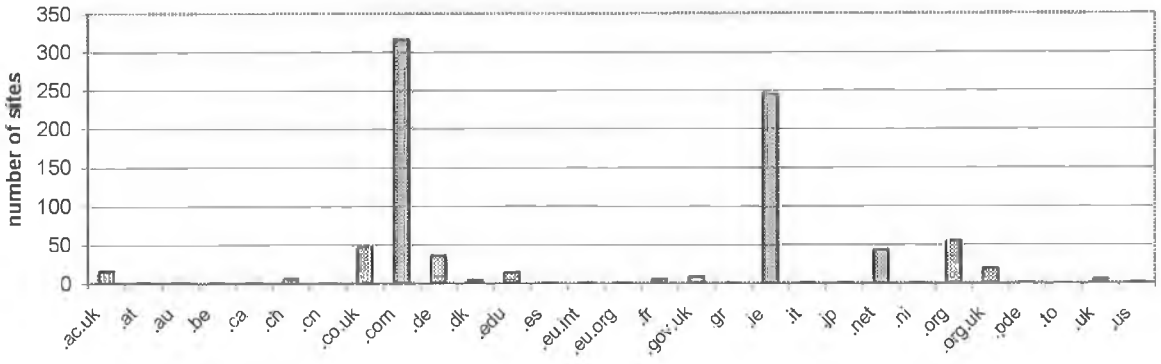


Figure 5.4 : Top level domain distribution of Gaeilge

The fact that there were no URLs left on the queue when the crawler stopped means that the queuing algorithm that was used for this crawl became irrelevant to the final dataset and therefore the nature of and structure of the dataset was solely reliant on the fact that we only crawled the Irish language web by following documents which contained at least a minimal amount of Irish text.

5.2.1.4 HYPERLINK STRUCTURE OF THE GAEILGE CRAWLED DATA.

Upon completion of the crawl, we were in a position to examine the hyperlink structure of the documents that our crawler discovered. Table 5.3 details the linkage structure of the Gaeilge crawled data.

Number of links in dataset	264,794
Number of off-site links	41,590
Number of on-site links	223,204
Average off-site indegree	1.55
Documents with a non-zero off-site in-degree	2,685
Documents with a non-zero on-site in-degree	23,115
Average off-site in-degree for non-zero documents	15.49
Ratio of off-site : on-site links	1 : 5.37

Table 5.3 : Linkage Structure of the Gaeilge crawl

A preliminary evaluation of the link structure of the Gaeilge crawl indicates a total of 41,590 off-site links within the dataset. However, in total, only 2,685 documents contain off-site in-links. This

shows than only 10% of documents have off-site in-links, but compared to the 5% of WT10g this is a doubling of the number of off-site in-links, however this does fall short of WT_Connected which is at 26%. Given the large size of the set of seed URLs, the density of off-site in-links into these documents was an issue. Figure 5.5 illustrates the number of documents with non-zero off-site indegree distributions grouped into buckets of 1,000, ordered in the order that the documents were downloaded.

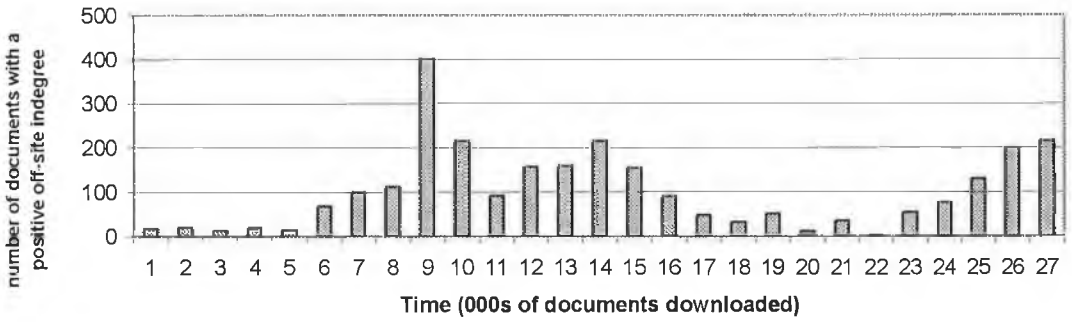


Figure 5.5: The distribution of documents with a non-zero off-site indegree within the Irish Language crawl

From Figure 5.5 we can see a limited number of documents containing non-zero off-site citations with documents that the crawler found early on (up to 6,000) as well as documents in the 17,000 to 24,000 range. Examining the reason for this, it appears (as a direct result of the queuing algorithm) that the large number of documents that comprise the seed URLs for the crawler are represented (with very few, exceptions) in the first 8,322 URLs in the graph. Any exceptions would be documents with off-site citations that got to the top of the queue ahead of some of the documents that comprise the seed URLs. After this point, we can see a marked increase in the number of documents with non-zero off-site citations. This leaves 2,307 documents containing off-site citations from the remaining 18,476 documents, or 12.5% of documents in total. In fact, only 4.3% of documents associated with the 8,322 seed URLs contain off-site in-links.

This is a problem and seems to illustrate the fact that our crawling based on language limits the possibilities of our crawler finding documents that are a source of off-site in-links into the seed URL documents. Intuitively, we would not expect that there would be a significant difference between the number of pages with a non-zero off-site indegree between the seed URL documents and the documents that were downloaded that did not comprise the seed URLs. This is especially

surprising because Google, the source of the seed URLs, incorporates linkage analysis (in the form of PageRank) into its ranking and based on our findings with WT_Connected and research undertaken at AT&T Research labs, NJ [Amento et al., 00], the difference in retrieval performance between using raw off-site indegree and PageRank as a source of linkage information is not huge. So therefore, we conclude from this crawl that the average off-site indegree value for each document is actually being underestimated and that there are more off-site in-links than we have discovered by following only Irish language pages.

The other irregularity in the graph is the dip in the number of documents with non-zero off-site indegree between 17,000 and 24,000 documents, which is present because a large number of web pages contained deep within the hierarchical structure of web sites were queued around this point.

So let us compare WT_Connected to the Gaeilge crawl.

	WT_CONN	GAEILGE
Number of documents	120,494	26,798
Number of off-site links	171,740	41,590
Average off-site indegree	1.43	1.55

Table 5.4 : Comparing off-site in-degree statistics of WT_Connected and Gaeilge

Comparing Gaeilge to WT_Connected we can see that although the dataset sizes are very different, the average off-site indegree of each document is very similar for both datasets. This is promising and seems to lend weight to the findings of our experiments on WT_Connected, but we have identified a possible irregularity with regard to the distribution of documents with a non-zero off-site indegree and consequently we are interested to see if we find similar figures from the other two (larger) conventional crawls.

5.2.1.5 CONCLUSIONS FROM THE GAEILGE CRAWL

Our first conclusion that we can draw from the generation of this dataset regards the fact that the average off-site indegree of each document is very similar for both datasets, although we have identified a problem with the number of documents containing a non-zero off-site indegree, which suggests that the crawl underestimates the true density of off-site links.

Our second conclusion regarding the crawling of Irish language specific data leads us to believe that it is not possible to accurately crawl a large percentage of documents in the Irish language relying on following the link structure of the documents alone. Whether this is due to the link structure of the documents themselves, or due to the bi-lingual nature of the documents which would cause the set of Irish documents gathered by our crawler not to reach an acceptable level of connectivity, we do not know. This did cause problems as illustrated in Figure 5.5 where the distribution of off-site indegree across documents could in no way be considered to be uniform, which is a result of problems with using a large seed URL set and producing a small dataset.

Recall that we generated eight representations of each downloaded document to facilitate the provision of content-retrieval facilities of the datasets if we so wished. With the Gaeilge dataset, we did index all downloaded Irish language documents and provided content retrieval facilities over the documents using a version of our search engine described in the previous chapter. This search engine called Focail[48] implemented our scarcity-abundance technique for regulating the influence of linkage analysis and was the first Irish language search engine that we are aware of, but access is restricted to inside the University only. A screenshot of the search engine is shown in Figure 5.6.

[48] Focail is an obvious name for an Irish language search engine as it is the Irish word for 'word'

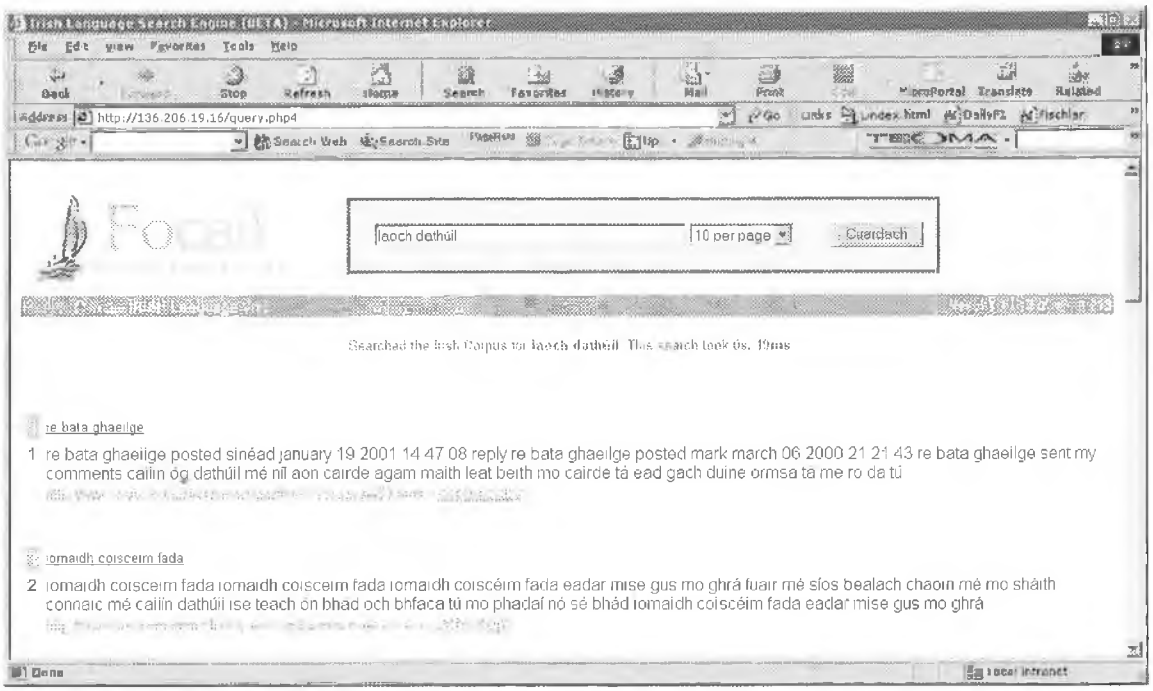


Figure 5.6 : The Focail Search Engine

5.2.2 Conventional Web Crawls

In addition to the Gaeilge crawl, we made two conventional (not language dependent) web crawls implementing two different queuing algorithms, one of which was based on an ageing queue with highly weighted URLs linked via off-site links and the other crawl was designed to maximize the number of websites that the crawler gathered documents from. We will refer to these two conventional crawls as 'ageing-crawl' and 'website-crawl' respectively.

5.2.2.1 Ageing Crawl : The Second Conventional Web Crawl

This crawl, the first of our conventional crawls, produced a WT2g sized dataset, which comprised 253,922 HTML documents. Total links within the dataset are 9,081,632. When we stopped the crawler, another 3,116,690 URLs had been identified and remained on the queue.

5.2.2.1.1 Queue Details

The queue that was implemented for this crawl was an ageing queue. All seed URLs were weighted highly prior to crawling. Once the crawling process was underway each new URL being enqueued was firstly classified as being on-site or off-site with URLs associated with off-site links

being allocated a higher weight than on-site linked URLs. This was done so as to ensure that documents linked to via off-site links would be included in the dataset. In addition, a small random value was used to influence weights to keep too many pages from a single site being dequeued together. Since the queue was an ageing queue, after each document was downloaded, all documents on the queue were aged by one age unit.

5.2.2.1.2 Starting Set of Documents on the URL Queue

As we have seen, each web crawler must have a set of seed URLs from which to begin its crawl. Prior to crawling, a number of handpicked pages (twenty) were chosen to act as the seed URLs. These URLs comprised the top eighteen URLs as ranked by our SiteRank experiment at AT&T (see Appendix B) and two local interest URLs of our own choice.

5.2.2.1.3 Statistics of the Ageing Crawl

As previously mentioned, this crawl produced 253,922 documents of conventional web data using the crawling and queuing approach described in the previous pages. These web documents originated from 26,730 different web sites. Table 5.5 illustrates the nature of the dataset and provides statistics on the documents themselves and the queue.

Number of Documents	253,922
Number of Servers	26,730
Average Documents per Server	9.5
Number of Documents left on Queue	3,116,690
Number of Servers left on Queue	125,374
Number of unseen Servers left on Queue	108,397
Average number of images per Document	32
Average number of terms per Document	620
Average length (in bytes)of text in each Document	4575
Average length (in bytes) of each Document	24,306
Number of Seed URLs	20

Table 5.5 : Statistics of the first conventional web crawl (ageing crawl)

Owing to the fact that we stopped the crawler after it had downloaded 253,922 documents, the queuing algorithm would have been influential in the final composition of the full crawled dataset.

5.2.2.1.4 HYPERLINK STRUCTURE OF THE DATASET

Table 5.6 summarises the linkage structure of the ageing crawl dataset. These figures relate to the downloaded documents only and do not include any documents that were still on the queue.

Number of links within dataset	9,081,632
Number of off-site links	4,120,718
Number of on-site links	4,960,914
Average off-site indegree	16
Documents with a non-zero off-site in-degree	80,204
Documents with a non-zero on-site in-degree	215,414
Average off-site in-degree for non-zero documents	51
Ratio of off-site : on-site links	1 : 1.2

Table 5.6 : Linkage Structure of the first conventional crawl (ageing crawl)

A preliminary evaluation of the link structure of the ageing crawl dataset indicates a total of 4,120,718 off-site links between documents in the dataset. However, in total, only 80,204 documents actually contain off-site in-links. This illustrates that with this crawl 32% of the documents have off-site in-links as opposed to the 10% with the Irish Language crawl and 26% WT_Conn. One interesting aspect of this crawl is that the number of off-site links associated with downloaded documents is almost as large as the number of on-site links and the average off-site indegree of each document is 16. Anecdotally, this figure seems to be too large, but we needed to run more experiments with live WWW data (see next chapter) to validate our belief that this figure is too large.

	WT_Conn	Ageing Crawl
Number of documents	120,494	253,922
Number of off-site links	171,740	4,120,718
Average off-site indegree	1.43	16

Table 5.7 : Comparing off-site in-degree statistics between WT_Connected and the first conventional crawl (ageing crawl)

The ageing crawl is compared to WT_Connected in Table 5.7 and we can see that the dataset sizes are very different and the average off-site indegree of the ageing crawl is over 11 times that of WT_Connected. This is a huge difference and upon closer examination we discovered a reason for this large average off-site indegree figure. This reason is that there is a strongly connected component within the 253,922 documents which increases the number of off-site links between documents so that the average indegree figure for each document becomes artificially high. This strongly connected component is comprised of web pages from the popular "about.com" network of web sites and the very presence of this strongly connected component is dependent on the method we used to identify the domain to which a web page belongs.

Our method of identifying websites was based on examining the lowest level of the URL string, which is an acceptable and obvious method of doing this. For example, the lowest level of the URL 'http://research.microsoft.com/pubs/' is 'research.microsoft.com' and the lowest level of 'http://www.microsoft.com/servers/' is 'www.microsoft.com'. Both 'research.microsoft.com' and 'www.microsoft.com' could not be considered to be from the same domain (although they both represent a single organisation) and we believe that it would be a mistake to do so. Yet, the strongly connected component of the 253,922 documents is strongly connected precisely because we do view the entire lowest level of the URL string as being the domain.

The 'about.com' network is comprised many hundreds of websites referred to as 'Guide' sites which are organized in a hierarchical structure, each of which links back up its hierarchical path to the root page. For example 'http://home.about.com' links to 'http://history.about.com/' which links to 'http://europeanhistory.about.com/' each of which links back up the hierarchical path back to all sites along the path and also onto many other sites within the 'about.com' network. In all, 44,880 documents originate from the 'about.com' network with a further 271,356 documents on the queue. The average off-site in-degree of each of these 44,880 documents is 39 and if we remove them from the dataset we are left with a dataset of size 209,042 documents and the revised linkage statistics are shown in Table 5.8.

Number of links within dataset	5,192,350
Number of off-site links	2,099,387
Number of on-site links	3,439,193
Average off-site indegree	10
Documents with a non-zero off-site in-degree	74,040
Documents with a non-zero on-site in-degree	176,570
Average off-site in-degree for non-zero documents	28
Ratio of off-site : on-site links	1 : 1.6

Table 5.8 : Revised linkage statistics for the first conventional crawl (ageing crawl)

As can be seen, when this strongly connected component is removed from the dataset, the average off-site indegree of each document drops from 16 to 10 and the ratio of off-site : on-site links drops from 1 : 1.2 down to 1 : 1.6. However, the number of off-site links into each document averaging at 10 still seems to be very high. This we feel is still overestimating the number of off-site links on the WWW.

	WT_CONN	REVISED AGEING CRAWL
Number of documents	120,494	209,042
Number of off-site links	171,740	1,753,157
Average off-site indegree	1.43	10

Table 5.9 : Comparing off-site in-degree statistics between WT_Connected and revised figures for the first conventional crawl (ageing crawl) with the strongly connected component removed.

Comparing these revised figures to WT_Connected in Table 5.9, we can see that although the dataset sizes are still quite different, the average off-site indegree of each document is closer to WT_Connected than was the case previously. Can we assume that an average off-site indegree figure of 10 is more representative of the WWW than the 1.43 figure of WT_Connected and the 1.55 figure of Gaeilge? Once again, we could not come to any definite conclusion and additional research is required.

5.2.2.1.5 Conclusions from this Crawl

This crawl does suggest that the off-site indegree density within both WT_Connected and the Gaeilge crawl underestimates the true linkage structure of the WWW, which intuitively we felt was the case after examining the linkage structure of the Gaeilge crawl. Even when we extracted the strongly linked component associated with the popular "about.com" network of web sites, the average off-site indegree figure still remains at 10, which we believe is too high. We ran the second conventional web crawl to see if we could replicate any of the findings from within either the Gaeilge crawl or this crawl.

5.2.3 WebSite Crawl : The Second Conventional Web Crawl

In addition to the previously described crawl, we ran the crawler again this time with a different queuing algorithm, which is described below. This dataset produced by the second crawl consisted of 126,996 HTML documents. Total links within the dataset were 4,401,017 and another 3,087,859 other documents were identified and remained on the queue. So once again, the queuing algorithm was influential in the final construction of the dataset.

5.2.3.1 Queue Details

The queue that was implemented differed from the queue implemented for the previous conventional web crawl in that much more emphasis was placed on gathering documents from as many websites as possible. Where a URL was found from a website that had not yet been visited by the crawler, that URL was weighted so that it would be near the top of the queue. This will inevitably have an effect on the average documents per server that this crawl found.

5.2.3.2 STATISTICS OF THE SECOND CONVENTIONAL CRAWL

Our seed URLs for this crawl were generated in a manner similar to the generation method for the previous conventional crawl (ageing crawl). This crawl produced 126,996 documents of conventional web data. This is equivalent in size to the WT_Connected dataset described in the previous chapter, so we should find that the linkage structure is not that dissimilar to WT_Connected, were WT_Connected to accurately reflect the true linkage structure of documents on the WWW. These 126,996 documents originated from 117,312 unique domains, which clearly illustrates the success of the queuing algorithm. Table 5.10 summarises the nature of the dataset with figures from the ageing crawl (Table 5.5) also included.

	WEBSITE CRAWL	AGEING CRAWL
Number of Documents	126,996	253,922
Number of Servers	117,312	26,730
Average Documents per Server	1.1	9.5
Number of Documents left on Queue	3,087,859	3,116,690
Number of Servers left on Queue	332,554	125,374
Number of unseen Servers left on Queue	218,673	108,397
Average number of images per Document	24	32
Average number of terms per Document	564	620
Average length (in bytes) of text in each Document	3,817	4575
Average length (in bytes) of each Document	21,875	24,306
Number of Seed URLs	20	20

Table 5.10 : Statistics of the second conventional crawl (website crawl).

The most striking figure from this crawl is the average number of documents per server, which is 1.1, especially when compared to 9.5 for the ageing crawl. This clearly illustrates the fact that the crawler did indeed hit a large proportion of websites. Another major difference is the number of unseen servers left on queue. Although the website crawl has only downloaded about half the number of documents as the ageing crawl, we see that there are 218,673 unseen servers left on the queue as opposed to 108,397 unseen servers left on the queue of the ageing crawl. This clearly shows that the queuing algorithm not only affects what documents are downloaded, but by favouring documents on new (unseen) servers we also influence the documents on the queue as well.

5.2.3.4 HyperLink Structure of the Dataset

Table 5.11 illustrates the linkage structure of the dataset.

	WEBSITE CRAWL	AGEING CRAWL
Number of links within dataset	4,401,017	5,192,350
Number of off-site links	2,647,701	2,099,387
Number of on-site links	1,773,316	3,439,193
Average off-site indegree	21	16
Documents with a non-zero off-site in-degree	124693	74,040
Documents with a non-zero on-site in-degree	46440	176,570
Average off-site in-degree for non-zero documents	21	28
Ratio of off-site : on-site links	1 : 1.48	1 : 1.6

Table 5.11 : Linkage Structure of the website crawl and the ageing crawl

Once again (as was the case with the ageing crawl) the average off-site indegree of each document seems too large, this time the figure is 21. Examining the link structure of the second crawl we find that only 412 documents from the 'about.com' network [ABOUT, 02] exist within the 126,996 documents and they only have 6,288 off-site in-links associated with them so their removal would have no noticeable effect of the statistics presented above.

	WT_Conn	Website Crawl	Ageing Crawl
Number of documents	120,494	126,996	253,922
Number of off-site links	171,740	2,627,701	4,120,718
Average off-site indegree	1.43	21	16

Table 5.12 : Comparing off-site in-degree statistics between WT_Connected and the website and ageing crawls

When compared to WT_Connected, the dataset size of the website crawl is roughly equivalent but there is almost a fifteen-fold increase in the average off-site indegree figure for each document, which once again suggests that WT_Connected (and the Gaeilge crawl) is underestimating the true linkage structure of documents on the WWW. Our belief is that the figure of 21 is far too high to represent the real WWW average off-site indegree, and we have identified some issues with this crawl (like the previous crawl) that do pose cause for concern and would influence the average

off-site indegree figure. Figure 5.7 illustrates the composition of the top 100 documents from the crawl, as ranked in order of off-site indegree using our own rough classification. As can be seen 46% of the documents all originate from one highly connected group of web pages all based around the 'deviantart.com'[49] [DEVIANTART, 02] domain. This problem is similar to the 'about.com' problem from the last crawl. Upon closer examination, we found there to be 5,121 documents from this domain in the 126,996-document dataset (with another 95,043 on the queue). These 5,121 documents are the target of 340,732 in-links, which would reduce the number of off-site in-links by 13% to 2,286,969.

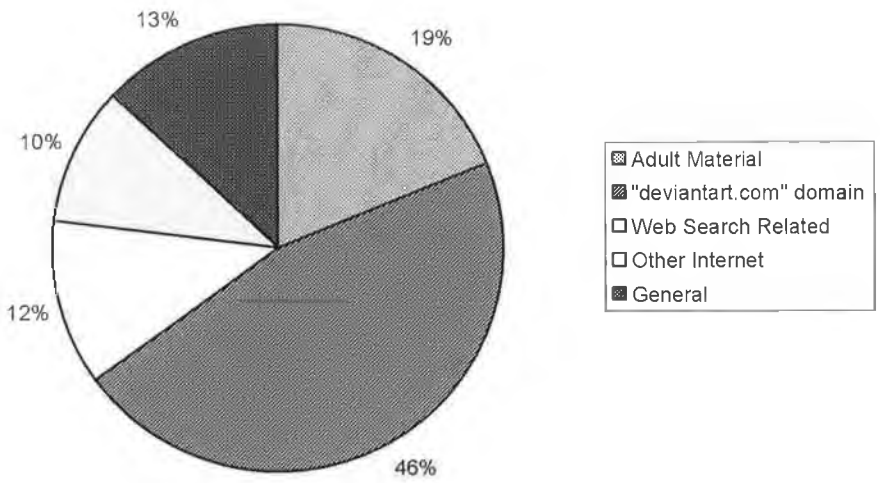


Figure 5.7 : Examining what documents comprise the top 100 indegree documents from the second crawl

In addition, the second largest grouping of web pages is from sites containing adult material. Examining the link structure of these sites suggests that they are densely interlinked and form a tightly connected community[50] and that this will further skew the indegree figures for the dataset as a whole.

Regarding the density of adult content within the 126,996 documents, we ran a small experiment to identify how many of the top 10,000 documents ranked by off-site indegree actually

[49] DeviantArt is a popular online art community

[50] Primarily we believe that commercial reasons prevail here and commercial reasons dictate that these websites containing adult content will be densely interlinked.

contained adult content and what was the average indegree of these documents. Our estimate is that 50% of the top 10,000 ranked documents are related to adult content and that each of these documents has an average off-site indegree of 162. These figures are based on an examination of 100 documents chosen at random from the top 10,000. This figure of 50% would result in 810,000 off-site links existing into these pages from only the top 10,000 documents which, when combined with the previous reduction, would reduce the off-site indegree figure to 1.48 million and consequently the average off-site indegree figure is reduced to 12.6. Further detailed examination of the dataset will most probably find more of these sites and thus reduce the indegree figures further.

This issue arose because of the crawling algorithm that we implemented which aims to reach as many websites as possible. Consequently, when the crawler finds a densely interlinked network of websites it will likely visit each website and, depending on the link structure of this network of web pages, the average indegree count could rise dramatically, as was the case with this crawl. Indeed the higher the indegree count of these web pages, the higher the score they will have on the URL queue. Based on the extraction of adult content from the top 10,000 pages and the removal of the 'deviantart.com' network we present revised linkage figures for the website-crawl in Table 5.13. Note that these figures would be subject to further downward revision were we to remove all adult content from the dataset and not just content from the top 10,000 pages.

	WT_CONN	REVISED WEBSITE CRAWL
Number of documents	120,494	116,875
Number of off-site links	171,740	1,476,969
Average off-site indegree	1.43	12.6

Table 5.13 : Comparing off-site in-degree statistics between WT_Connected and the revised figures for the website crawl

After examining the results of the three crawls of web data that we made we are only in a position to conclude that we are not sure as to the actual link structure of the WWW and if indeed simply gathering a dataset of documents will be sufficient to accurately recreate the linkage structure of the WWW.

5.3 COMPARING ALL FIVE DATASETS

These comparisons have been carried out on the original datasets that we crawled, and not on the revised figures that we have presented after examining the crawled data. Table 5.14 shows a comparison of document and server figures for the three crawls, WT10g and WT_Connected.

	WT10G	WT_CONN	GAEILGE	AGEING	WEBSITE
Number of Documents	1,692,096	120,494	26,798	253,922	126,996
Number of Servers	11,680	11,611	846	26,730	117,312
Average Documents per Server	144.8	10.4	32	9.5	1.1

Table 5.14 : Comparing the number of documents and servers across all five datasets

The most striking difference between the five datasets is the average number of documents per server. We have plotted these figures graphically in Figure 5.8.

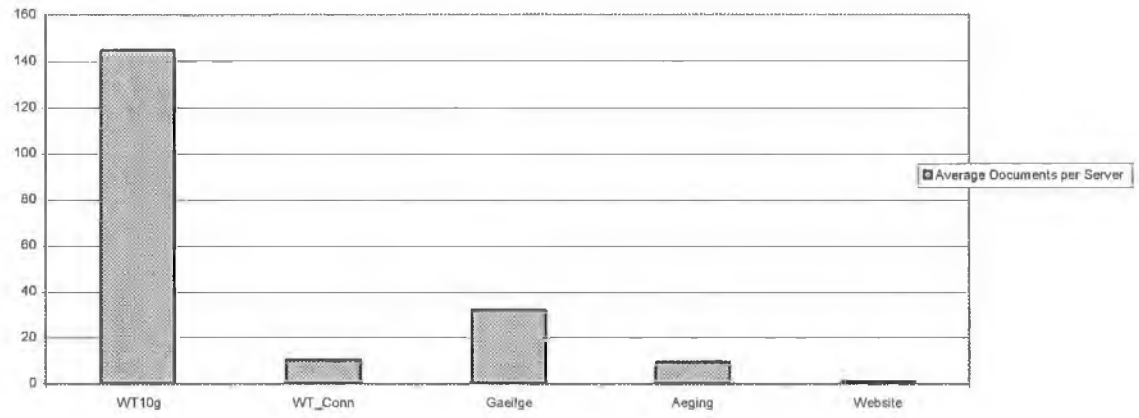


Figure 5.8 : The average number of documents per server for WT10g, WT_Connected and each of the three crawls

The standard deviation of the average number of documents per server value is 59.93, which is quite large and illustrates that these experiments could not identify the required average number of documents per server to accurately reflect the WWW on a small scale. If we remove WT10g from the calculation, the standard deviation figure is 13.18. Large differences exist between the figures in the three web crawls as well, but the reason for these is the fact that the Irish language

crawl was completed, thus removing all documents from the queue while both the ageing-crawl and the website-crawl were both halted with a large number of documents remaining on the queue. This would affect the number of servers crawled as a percentage of downloaded documents because both queues weighted URLs associated with off-site links higher than URLs associated with on-site links, so URLs which are linked across website boundaries would be at the top of the queue. The second conventional crawl (the website-crawl) has an even lower average number of documents per server value than the ageing-crawl because (as we have mentioned previously) the website-crawl imposes an added weight to URLs from websites that have never before been seen by the crawler. This discrepancy is more evident from Figure 5.9.

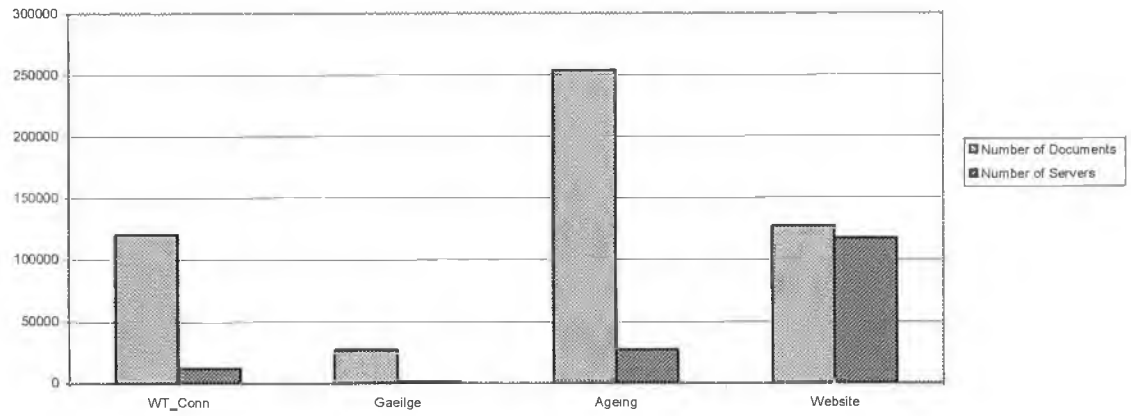


Figure 5.9 : Comparing the number of documents and the number of documents per server for WT_Connected and each of the three crawls

Here, when we compare the number of documents to the number of servers within the datasets for all but WT10g[51] we can clearly see the benefit that can be gained from the website-crawl queuing algorithm if the aim of the crawler is to maximise the number of web servers reached. This is in contrast to the ageing-crawl, which implements a more conventional queuing algorithm than the website-crawl, which shows a very similar value (9.5) to that of WT_Connected (10.4). However, the results are still inconclusive. We cannot generate a definite figure for the average number of

[51] The reason WT10g is not used is due to the fact that the size of WT10g would make the examination of the number of servers difficult as the graph would not be able to illustrate the figures as a proportion of WT10g documents.

documents per server required when crawling the WWW or if WT_Connected is representative of live WWW data.

We will now examine the sizes of the crawls when compared to WT10g and WT_Connected.

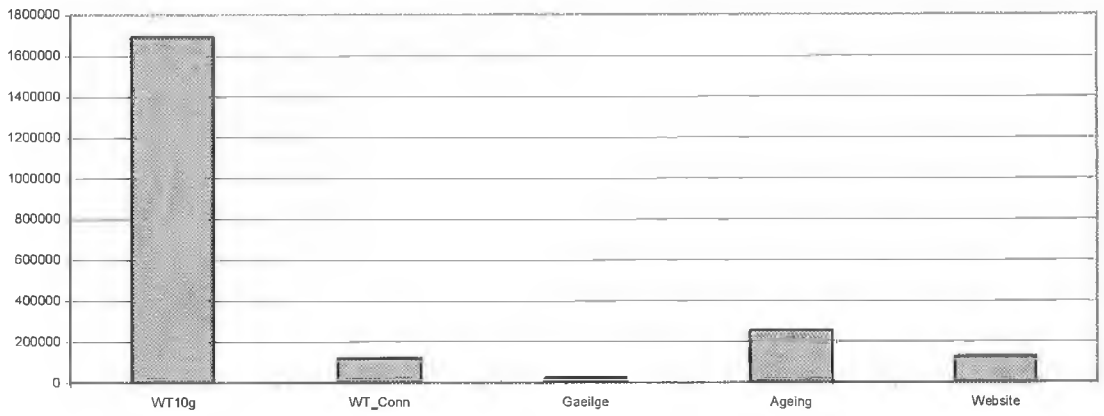


Figure 5.10 : Illustrating the number of documents in WT10g, WT_Connected and each of the three crawls.

Figure 5.10 clearly illustrates the differences in volume of documents between the five datasets. The ageing-crawl is the largest of the datasets that we have generated, but it is only 15% of the size of WT10g. While the smallest dataset, the Gaeilge crawl is only 1.6% of the size of WT10g. Our feelings are that WT10g is the accepted standard dataset for both ad-hoc and web based retrieval experiments at TREC and that any new dataset to support experiments into linkage analysis should, at least, aim to replicate the size of WT10g.

Aside from the number of documents and servers within the datasets, the linkage issue is also of vital importance. In order to accurately evaluate which linkage algorithms will operate most effectively in the real world we must be able to generate a dataset with the correct density of off-site links and to a lesser extent on-site links. Table 5.15 shows the average off-site indegree and the percentage of documents with a non-zero off-site in-degree for both TREC (WT) datasets and all three crawls.

	WT10G	WT_CONN	GAEILGE	AGEING	WEBSITE
Average off-site indegree	0.1	1.4	1.5	16	21
Ratio of off-site : on-site links	1: 45.95	1 : 2.20	1 : 5.37	1 : 1.20	1 : 1.48
Percentage of documents with a non-zero off-site in-degree	1.8	25.9	10.0	31.5	98.2

Table 5.15 : Examining the linkage structure of the crawls

The most striking result from Table 5.15 is the average off-site indegree between the datasets. Figure 5.11 illustrates this more clearly. WT10g, as we have already identified, has a linkage structure that is incapable of supporting experiments into linkage-based retrieval, hence the construction of the more densely interconnected WT_Connected, which did show improvements in retrieval performance. However, we were not sure of how accurate the off-site links structure of WT_Connected was, therefore we built the crawler and ran the experiments that were outlined in this chapter. We were hoping to find some indication that either WT_Connected was representative, or not, of the WWW as a whole, but we were not able to do this. A huge discrepancy exists between the ageing-crawl and the website-crawl, and both of these crawls when compared to the Gaeilge crawl and the WT datasets. The results of our three crawls clearly illustrate that we were unable to come to any conclusions about WT_Connected by running these crawls. We still do now know what the average off-site indegree of documents on the WWW is and how this should be replicated in a dataset.

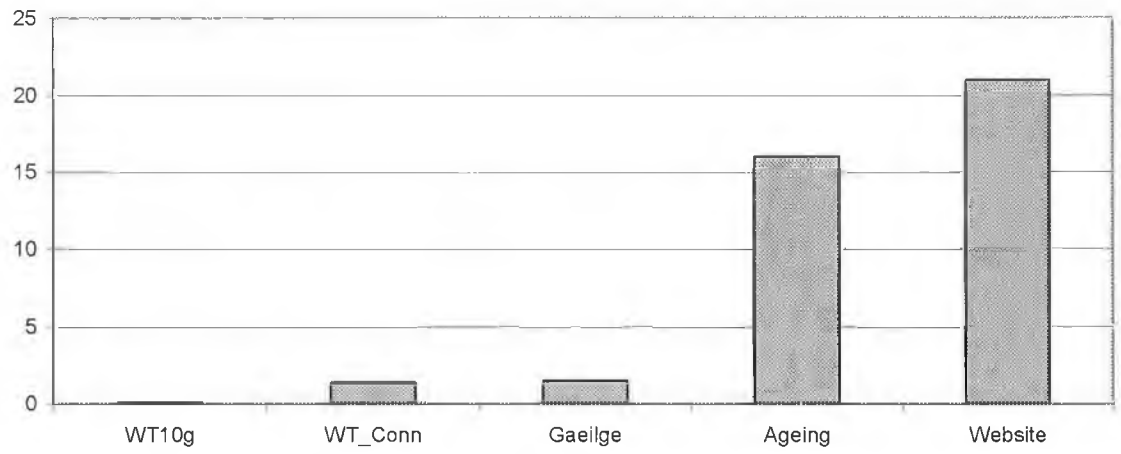


Figure 5.11 : The average off-site indegree of each crawl and the WT datasets

If we examine the standard deviation of the off-site indegree figures between these crawls and the TREC datasets, we can see that it stands at 6.68 with the average at 5.4. We have identified some problems with the crawled data in both the ageing-crawl and the website-crawl, which would influence these figures. We have shown that a more accurate figure for the ageing-crawl is 10 while a revised figure for the website-crawl stands at 12.6 and this is illustrated in Figure 5.12.

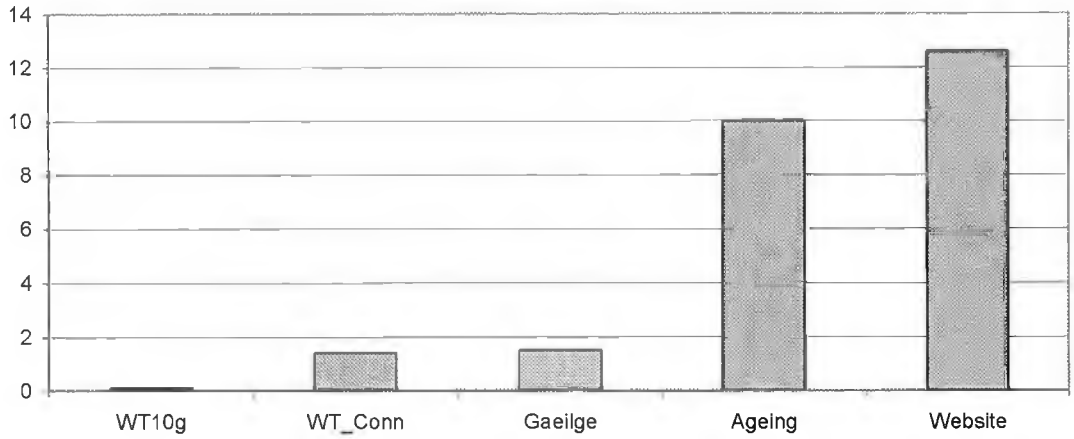


Figure 5.12 : The average off-site indegree of each crawl and the WT datasets using revised figures for the ageing and website crawls

We can reduce the average off-site indegree figure for the website crawl further by carrying out a more exhaustive examination of the dataset, yet there would still be differences between the crawls. How then can we assume that any of these represent the ideal off-site indegree for each document? Anecdotally, the figure of 10 off-site in-links for each document seems too large to be accurate.

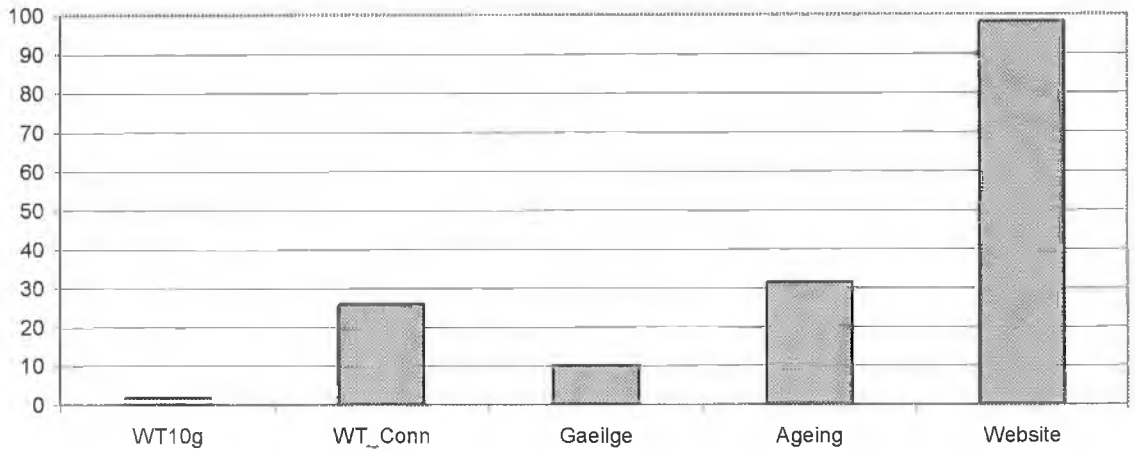


Figure 5.13 : Illustrating the percentage of documents with a non-zero off-site indegree.

The difference between both the percentage of documents with a non-zero off-site in-degree on both the ageing-crawl and the website-crawl as shown in Figure 5.13, is striking. By varying the queuing algorithm, the structure of the final dataset can be changed enormously, with more than a threefold increase in the percentage of documents with a non-zero off-site indegree over the next best crawl. Clearly then it is impossible to simply develop web crawler and crawl a dataset of the required size in one process. We can summarise the reasons thus:

- We are still not sure of what the linkage structure of the WWW is; and
- We have shown that the queuing algorithm influences the nature of the crawled dataset (unless all documents from the queue are removed) and it is vital that the queuing algorithm is correct, but without knowing the nature of the linkage structure of the WWW we are not in a position to design the correct queuing algorithm.

We could keep varying our queuing algorithms and sending out web crawlers, but without having an exact goal for the crawler we would never know at what point the queuing algorithm is successful.

5.4 Conclusion

As a result of running these experimental web crawls it was obvious to us that in order to adequately evaluate the benefits of Linkage Analysis, a fully representative dataset would be required, but representing what? How do we know what a representative dataset should look like if no such dataset exists? It is intuitive that the larger a proportion of the web that we crawl and download, the richer (and more accurate) the linkage structure will be. Google, for examples indexes over 2 Billion web pages and therefore the linkage structure of its index would be more complete and better support linkage algorithms than would WT10g or probably even WT_Connected.

Consequently, any linkage analysis technique will benefit from a larger index and this should be reflected when building a dataset for experiments, yet a dataset that is too large will discourage participation among participants as the resources required would be too great. We should attempt to accurately represent the rich linkage structure of the web in a small dataset but how can this be done? We will look into this in the next chapter. What we do now know, however is that sending out a web crawler to generate a dataset is not the answer in that we are very unlikely to stumble across the correct linkage structure by crawling a dataset. Therefore, we must examine the linkage structure of the WWW to see what clues we can extract to identify what are the requirements of an ideal dataset to support faithful experiments into linkage-based retrieval.

The reason why we stopped the crawling process with a large number of documents remaining on the queue for both the ageing and website crawls was primarily because it was taking large amounts of time to download the documents. The average network latency in simply downloading each document was in the order of 2.5 seconds[52]. In addition, the time taken to parse each document, generate disk files and add link data into a crawl database, update the URL queue, select the next URL and adhere to the robots exclusion standard clearly illustrates that these web crawls were lengthy processes. Although each crawl was completed with minimal interruption, the crawls (excluding the smaller Irish language crawl) would have taken 3 and 5 weeks to complete.

Clearly, had we developed a crawler specifically to support more efficient downloading of large numbers of web pages we could have generated much larger crawls in a reasonable time period,

[52] This figure of 2.5 seconds is based on a random sample of 200 documents from unique URLs extracted from the URL queue of the WebSite Crawl. This does seem to be a very long time and secondary observations from the experiment shows that the vast majority of documents are downloaded in under a second, with a small number taking considerably longer.

however we were primarily interested in examining the behaviour of web crawlers and their ability to crawl a dataset to support faithful experiments into linkage-based web retrieval.

5.5 SUMMARY

Web Crawlers are complex software tools that traverse the WWW gathering web pages. Web crawlers can not know of the existence of all documents in the WWW, rather, given a small set of seed URLs, a crawler will extract links from downloaded documents and using a queue of URLs will store the URLs of documents that it has not yet seen. The prioritisation of URLs to download is based on a queuing algorithm.

We made three crawls of live WWW data for these experiments. The first crawl was an Irish language specific crawl which downloaded 26,798 documents in the Irish language, each of which had an average off-site indegree of 1.55 which was quite similar to the average off-site indegree figure of WT_Connected which was encouraging for the results of our experiments in the previous chapter, yet our intuition suggested that the distribution of off-site indegrees across the 26,798 documents was not natural and reflected the choice of seed URLs that we made. Our belief was that the true off-site indegree of documents was actually being underestimated by the Irish language crawl.

The second and third crawls were more conventional crawls of WWW data, in that they were not language specific and no restrictions were placed on the crawler's movements when gathering documents, save any restrictions from the crawler's adherence to the robots exclusion standard. The findings of these two crawls (264,794 and 126,996 documents in size) suggests that the figure of 1.55 off-site links into each document as found by the Irish language crawl was indeed an underestimation of the real nature of WWW data. These crawls produced average off-site indegree figures of 16 and 21 respectively, however, when these figures are adjusted to take into account irregularities we found in the link structure of the crawled data the figures became 10 and 12.6 respectively. However, neither of these figures was similar to the figures for both the Irish language crawl and WT_Connected. We refer the reader to Figure 5.10, 5.11 and 5.12 as well as Table 5.15 for a comparison of the findings from the web crawls.

Given that the three crawls produced different average off-site indegree figures, we were not in a position to declare that WT_Connected is actually representative of true WWW linkage structure. Rather, more experimentation was needed and this is presented in the following chapter.

Chapter 6

PROPERTIES OF AN IDEAL TEST COLLECTION TO SUPPORT FAITHFUL EXPERIMENTS INTO LINKAGE-BASED RETRIEVAL

In this chapter, we discuss our random sample of 5,000 web pages. We examined the linkage structure of web pages so that we can identify the requirements of any future test collection that aims to support faithful evaluation of linkage-based techniques. After discussing the random sample, we examine the distribution of off-site indegrees from the 5,000 web pages and show that they follow a power-law distribution. Finally, we develop a set of requirements for any future test collections that aim to support linkage-based experimentation.

6.1 INTRODUCTION

When building a test collection to support experiments into retrieval of documents from the WWW there are a number of requirements that should be adhered to [Bailey et al., 01]. These requirements are to:

- Model real web search, by means of
  - A sufficiently large dataset
  - Representative web queries
  - Sufficiently complete relevance judgments.
- Include the required and appropriate type of link density to enable meaningful experiments into linkage-analysis based methods.
- Support algorithms for experimentation into distributed IR which allows merging of results from different types of retrieval systems and methodologies.
- Be of a high enough quality so as not to discourage people from using it. High (in this case) means that all reasonable attempts must be made to remove binary and unclean data from the dataset.

We have already seen that WT10g (and its predecessor WT2g) does not support truly investigative experiments into linkage-based retrieval, findings which all other participants in the TREC web track agree upon. Our experiments using WT_Connected do show modest improvements in retrieval performance when linkage-based approaches are compared to conventional content-only approaches. Consequently, to support further research in the field of WWW IR, we must identify what key components should comprise an ideal test collection that is capable of supporting experiments into linkage-based retrieval. Given that the results of our crawls presented in the previous chapter were inconclusive in answering the question of how to create such an ideal test collection, we must examine the structure of live WWW data in order to correctly identify and characterize its linkage structure for replication in a test collection that is to faithfully support linkage-based retrieval experiments.

6.2 Examining WWW Structure

After TREC-8 (1999) it was apparent to most of the participants in the web track that the one of the primary reasons for the failure to improve retrieval performance of any linkage-analysis techniques was due to the dataset [Hawking, 01], [Bailey et al., 01]. The WT2g dataset consisted of 247,491 HTML pages but the number of closed off-site links was hopelessly inadequate as can be seen in Figure 6.1 which illustrates the difference in average off-site indegree between the TREC test collections, WT_Connected and our crawls of web data (using the revised figures) as discussed in the preceeding chapter. WT10g, which followed from WT2g was not much better, even though it had a tenfold increase in the number of off-site links, and as we have seen, no group has been able to show improvements in retrieval performance for typical WWW queries using WT10g and link-based retrieval.

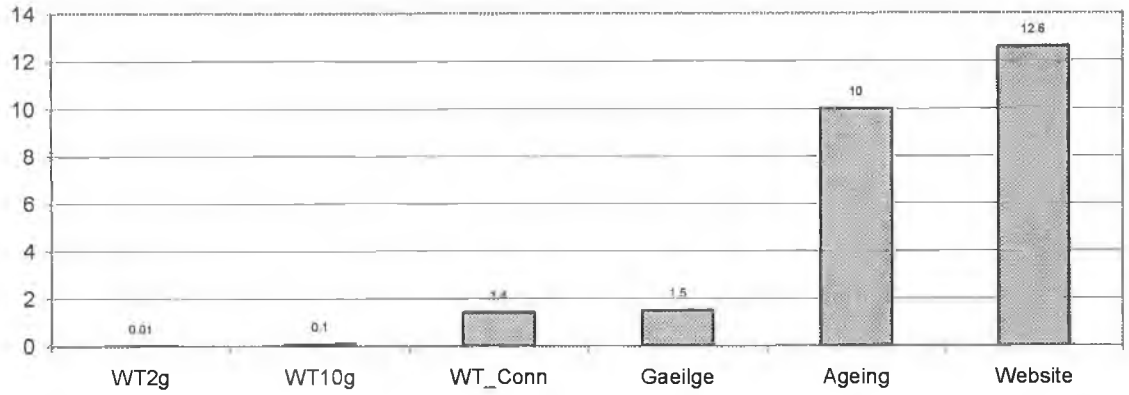


Figure 6.1 : Average document off-site indegree from a number of source

As can be seen from the diagram above, the average document off-site indegree values vary enormously and even our own crawls of WWW data were inconclusive in terms of capturing a faithful representation of the WWW, with each reporting different off-site indegrees. Consequently, we were not to know how representative any of the figures are of real WWW off-site indegrees. In Chapter 4, we have shown that modest improvements in retrieval performance are possible when incorporating linkage-based retrieval techniques into conventional IR ranking algorithms when using the WT_Connected dataset. Therefore, we have undertaken to examine the linkage structure of the WWW so that we are in a position to:

- Clarify, if indeed, WT_Connected accurately recreated the off-site indegree structure of the WWW, which we did not think was the case and;

- Identify the indegree structure of the WWW for both off-site and on-site links so that we are in a position to identify requirements for any new dataset that is being generated to support experiments into linkage-based retrieval.

-

6.2.1 Details of real-world experiments

We have seen that increasing the density of off-site links between documents within the test collection can lead to modest improvements in precision (as was the case with WT_Connected), so therefore the ideal course of action would be to increase the number of off-site links within a test

collection until we meet the average indegree, or ore specifically, the average on-site and off-site indegree of documents on the WWW. Our web crawls to try and determine these figures were inconclusive in that all three returned different figures for average document off-site and on-site indegrees. It is intuitive that the larger a proportion of the web that we index, the richer and more representative the linkage structure will be. Consequently, the results of any experiments into linkage analysis techniques will be more accurate and benefit from a larger index, which by its very nature would be more representative of the WWW. This makes sense, if we take Google as an example, it indexes 2,073,418,204 web pages (as of July 2002) and we believe that its PageRank algorithm does indeed improve retrieval performance.

However, many of the research groups around the world will be unable to process large amounts of data, but would be still interested in running linkage-based experiments. This is why the TREC web track keeps the size of its dataset at 10GB (with the larger 100GB large web task dataset for participants who have the resources to process this much data). This aspect should be considered when building a dataset for experiments; it must not be too large so as to discourage participation, but must accurately represent the rich linkage structure of the web. TREC web test collections have kept the datasets small in size, but have been unable to accurately recreate the linkage structure of the WWW and this is borne out by the poor retrieval performance of linkage algorithms in TREC experiments over the past three years.

In order to identify the requirements for a small-scale test collection, which faithfully models the linkage structure of the WWW, we must to examine in detail the structure of the WWW itself.

6.2.1.2 VIEWING THE WWW AS A GRAPH

From graph theory, we know that an edge connects two nodes together in a graph. If the graph is a directed graph, each edge will have a source and a target node associated with it. Any outgoing edge from one node is an incoming edge on another node, so we can view each edge as being an in-edge from one node and an out-edge from another node. On the WWW, which is a huge directed graph, we generally assume that a link connects two documents together and the same observation applies to links as to edges in a graph. We can assume that an out-link from one document will be an in-link into another document[53]. It follows trivially that the number of in-links is

[53] We accept that this is not always the case, as hyperlinks do actually exist which do not actually point to another web page because the destination page has been deleted or removed, or the link itself is incorrect.

equal to the number of out-links and this should allow us to identify the web page indegree requirements based on outdegrees of web pages that we will observe.

6.2.1.3 How to identify In-links

Since the WWW is a directed graph and each out-link is also an in-link, were we to find by observation the average outdegree of each document on the WWW, then we can also identify the average indegree of every document. However, is this really the case? It will be clear to anyone that has spent more than a few minutes browsing the hyperlinked structure of the WWW that links can point to documents that no-longer exist. These are 'broken links' and normally result in the user seeing an 'HTTP 404 error: file not found'.

This is a problem that is caused by the very open nature of the WWW. Since anyone can publish and link, this results in the WWW being chaotic in nature in that no-one individual or organization is in control and the author of a web page can remove it, rename it, move it or alter its content at any time the author desires. Once the target document of a link is moved, unless the author of the document that contains the link knows of the change, then the link becomes and remains broken. It is this very chaotic nature of the WWW that we will employ later to identify the distribution of indegrees across web pages in a test collection.

However, if we can identify what proportion of links on the WWW are broken we can adjust the average outdegree figure to take into account the percentage of links that we found to be broken and let the new figure represent the average indegree of web pages, as in the following formula where N is the number of documents from which the links were examined and $error_n$ is the number of broken links found on a page n.

$$avgOutDeg = \left(\frac{\sum_{n \in N} outDeg_n}{N} - \frac{\sum_{n \in N} error_n}{N} \right) \qquad (6.1)$$

However, we need to know more than the average outdegree of each web page if we are to identify the characteristics of the WWW. We also need to know the average number of off-site out-links and the average number of on-site out-links which can be discovered by observation.

6.2.2 Surveying the Linkage Structure of the WWW

In order to correctly identify the average in-degree (both off-site and on-site) of web pages we must carry out our own survey of the linkage structure of the WWW as it is in 2002. Our chosen technique was to sample web pages at random. Previous work had been carried out in this area, an example being the SOWS survey of web structure, which also involved the random sampling of WWW pages. In addition, Cyveillance [Cyveillance, 00] have carried out experiments two years ago [Murray & Moore, 00] in order to size the Internet and they have found that the average page contains 23 on-site out-links and 5.6 off-site out-links, however they did limit their processing of web pages to pages under 200KB in size. In addition we were interested in links specifically to web pages based on examining in detail the links from pages, while SOWS and Cyveillance were simply interested in the number of links from a page, so we felt it best to carry out our own survey and detailed evaluation.

6.2.2.1 SOWS III

The latest SOWS survey [SOWSIII, 99] of the WWW (the third in a series) was carried out in 1999. The target population for the survey was (obviously) all documents from the WWW. Their chosen method of generating random URLs was to firstly generate two lists, each containing 45 randomly chosen terms. For each run, words from each list were paired randomly and used as query terms to the AltaVista search engine. To further avoid bias, a computer generated random number was used to select the starting point for extracting the results from AltaVista's query response, i.e. for each query AltaVista generated a ranked list of web pages but the extracted results were not always the top scored pages.

The survey was based on fetching and examining 200 URLs chosen randomly from a pool of a possible 1,964, the pool being chosen using the random URL generation method outlined above. One requirement for each of the chosen 200 URLs was that the document associated with the URL contained at least 4 out-links. The 200 documents were downloaded, examined and found to contain 4,851 links, 261 of them (5.7%) were dead-links and 28.5% of the 200 pages contained at least one dead link. We have summarized the findings of SOWS III in the following table.

Sample size (number of documents)	200
Number of links	4,581
Average number of links per document	22.9
Number of dead-links	261
Average number of dead-links per document	1.3
Percentage of documents containing dead-links	28.5%

Table 6.1 : Summary of the findings of SOWS III

However, no information was provided on whether these dead links were mostly off-site or on-site and this is information that we required. A basic statistical review of the accuracy of SOWS III is shown in Table 6.2.

Confidence level	95%	99%
Population Size	Unknown	Unknown
Sample size	200	200
Confidence Interval	6.93%	9.12%

Table 6.2 : Statistical review of the accuracy of SOWS III

As can be seen, at a 99% confidence level[54] the confidence interval[55] is 9.12%, which states with a probability of 99% that the sample is plus or minus 9.12% of the stated values shown for the sample. At the more commonly used 95% confidence level the confidence interval drops to 6.93% which (like the 99% confidence level) is quite high. Population size is ignored, as is often the case when a population is large or unknown. We felt that for our random sample of web pages, it would be beneficial to increase the sample size so that we could reduce the confidence interval.

6.2.2.2 Our Survey of 5,000 random web pages

We took a different approach to generating random URLs than that taken in the SOWS III survey. We developed a JAVA application that selected random URLs by using a web accessible random URL generator [UROULETTE, 02]. A handful of such services exist on the web, such as the

[54] The Confidence Level of a survey is expressed as a percentage and illustrates how certain one can be about the result of a survey, e.g. a 95% confidence level means that you are 95% certain of the result.

[55] The Confidence Interval of a survey is the plus-or-minus figure that illustrates how accurate the result of the survey is, e.g. a confidence interval of 5% means that any value taken from the survey can be + or – 5% of that value.

random page generator run by Yahoo [YAHOO, 02] which one must assume works over their index, so we avoided using this random page generator as we felt the Yahoo index would be more likely to contain high quality documents because it is a humanly generated index. Upon request URouLette will generate a random URL. Our sample size was 5,000 and the accuracy statistics for the survey are shown in Table 6.3. The 95% and 99% confidence intervals are both much smaller than the SOWS sample, which indicates that our sample could be classified as being more statistically reliable.

Confidence level	95%	99%
Population Size	Unknown	unknown
Sample size	5000	5000
Confidence Interval	1.39%	1.82%

Table 6.3 : Statistical review of the accuracy of our experiment

One caveat with our experiments is that the above figures assume truly random sampling and since we relied on URouLette for our random URLs, we are not sure how random our sample is and from how large a list of candidate URLs the random URLs are chosen. All random sampling techniques (even SOWS) that rely on choosing a document at random from a web crawl are only random with respect to the crawled data, which will be a fraction of the WWW, and thus can not be classidied as truly random. Truly random sampling would involve some form of random IP address generation and subsequent file selection. However considering this, we will refer to our sample as being a 'random' sample, although the degree of randomness is in question.

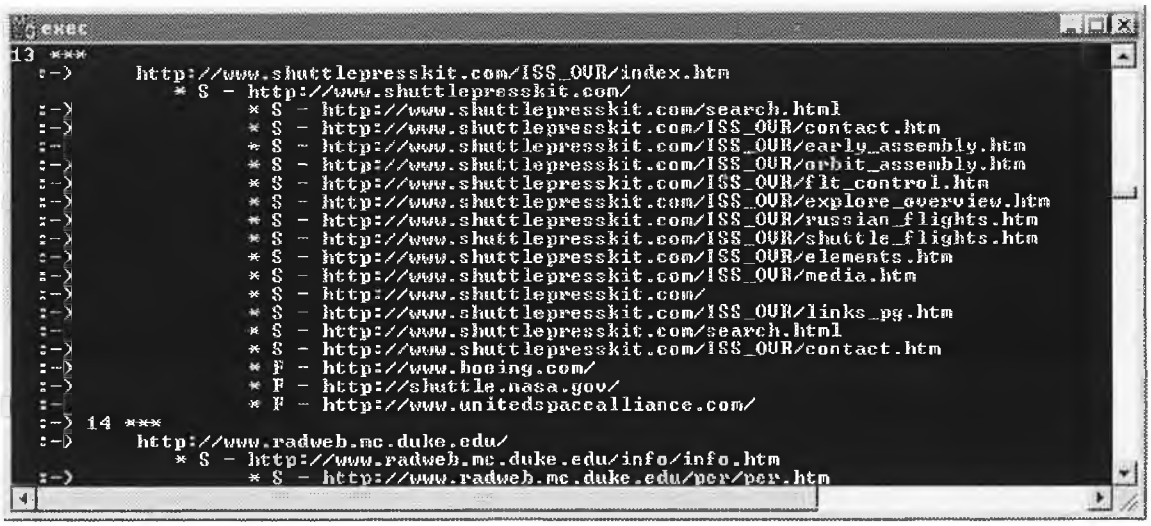


Figure 6.2 : The Random URL crawler showing the 13th URL and its out-links

For each of the 5,000 URLs that were sampled, that page was immediately downloaded, all embedded links were extracted and using the cache facility of our web crawler described in Chapter 5, saved to disk. An example of a document being downloaded is shown in Figure 6.2 and one of the downloaded documents is shown in Figure 6.3. In cases where the URLs associated with embedded links were represented by a relative URL, these were resolved into an absolute URL. Only URLs that used the HTTP protocol[56] were accepted, all other protocols were ignored. Each URL was then identified as being either off-site or on-site in nature and our application downloaded the associated document and stored this document on disk for subsequent validation that the document is actually content-bearing and not just a HTML encoded 'HTTP 404 : file not found' error. URLs that no longer existed and whose servers didn't return a HTTP 404 type error message were flagged as such and considered to be broken URLs.

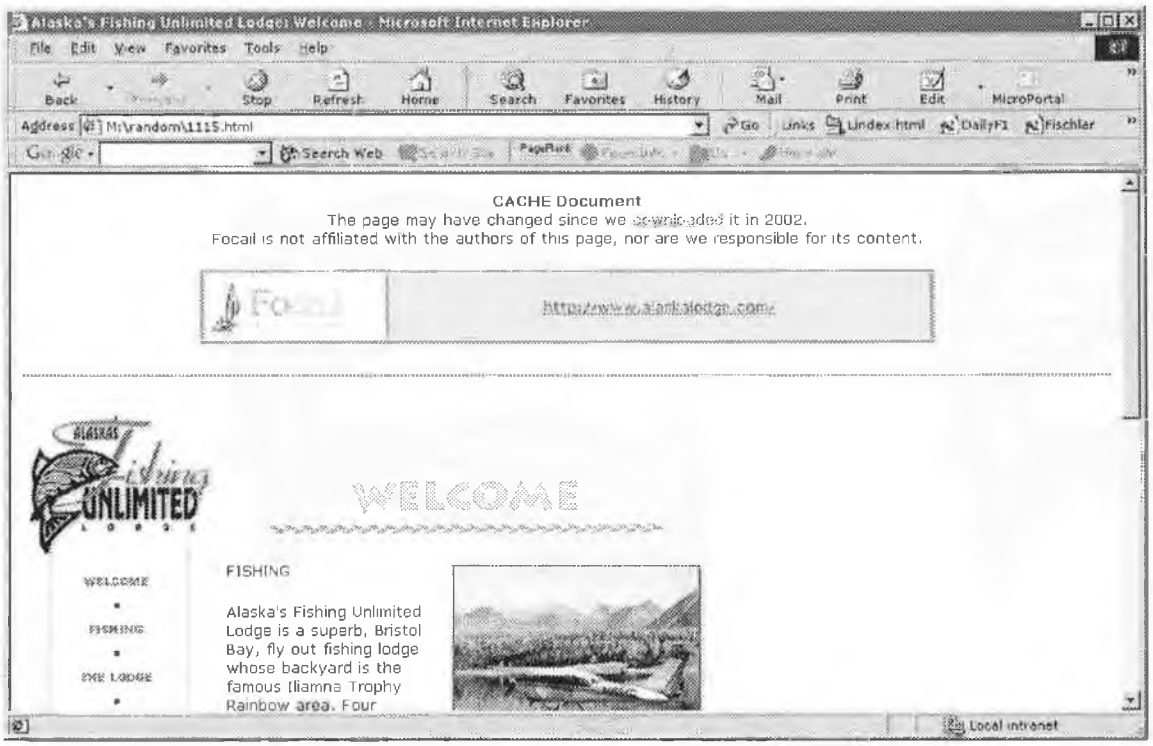


Figure 6.3 : A cached document from our random sample of WWW documents

We repeated this process 5,000 times over an eight-day period. The reason for the lengthy time period was that we did not wish to send many simultaneous requests to URouLette as well as the additional requirement that we downloaded all linked pages, which takes time.

[56] We didn't process any embedded links using MAILTO or GOPHER or FTP protocols.

The information that was stored about each downloadable linked document was as follows:

- Source and target URLs
- Source and target hosts
- Type of link (off-site or on-site)[57]
- The associated anchor text
- Flag identifying if the URL is dead or not, however our application could only identify if a URL didn't exist. In many cases an HTTP 404 type error message was returned by a web server, necessitating our saving each target URL to disk and inspecting each suspect candidate URL (based on the text within the document) whose content was based around explaining that the document was no-longer available. An example of one such page is shown in Figure 6.4.

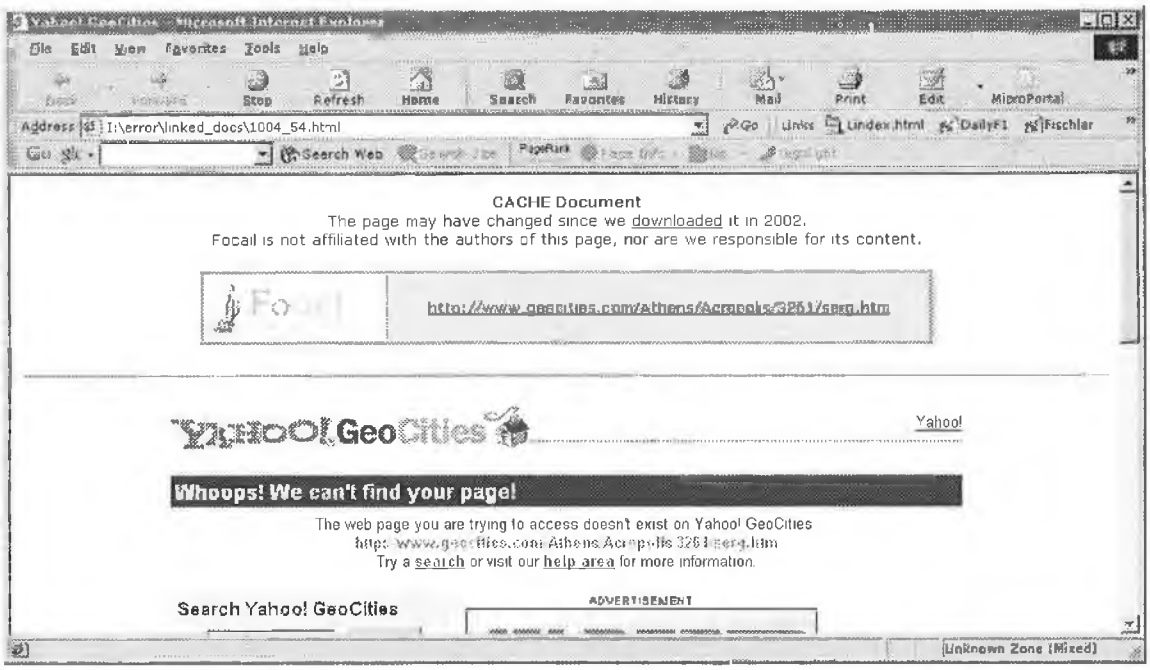


Figure 6.4 : An example of a downloaded page that no longer exists.

[57] Self-links are assumed to be on-site links in this case.

6.2.2.3 PRELIMINARY OBSERVATIONS FROM THE RANDOM SAMPLE (PRELIM)

Based on our experiments we are in a position to present our findings based on our random sample of 5,000 web pages. Let us start by looking at statistics for the number of documents downloaded.

Sample size (number of documents)	5,000
Number of unique documents downloaded	64,125
Number of documents identified	64,483
Total URLs discovered from out-links	60,945

Table 6.4 : Unique document statistics from our random sample of WWW documents

As can be seen from Table 6.4 our survey identified 64,483 web pages, but by extracting just the out-links from these documents fewer documents were found, which illustrates the number of URLs identified as out-link target URLs that were actually represented in the random sample. So let us examine the link structure of these 5,000 documents.

Documents that contain out-links	3,940
Documents with no out-links	1,060
Documents with off-site out-links	2,706
Documents with on-site out-links	3,571

Table 6.5 : Basic linkage structure of documents from the random sample

From Table 6.5 we can see that 78.8% of the sample documents (3,940) contain at least one out-link, leaving only 1,060 documents (21.2%) without even one out-link. Examining the nature of these links illustrates that 54.1% of documents have a positive off-site outdegree while 71.4% of documents have a positive on-site outdegree. However, this information is still not enough to identify the indegree structure of the full WWW. For this we need to examine the number of, and nature of, the links parsed from the 5,000-document sample, which is shown in Table 6.6.

Total number of links found	98,579
Number of which are off-site	25,813
Number of which are on-site	72,766

Table 6.6 : Examining the types of links found

As can be seen, a total of 25,813 off-site links and 72,766 on-site links were parsed from the 5,000-document sample. Based on the findings shown in Table 6.6 from our sample, we are in a position to state average outdegree figures for each document, as shown in Table 6.7.

Average off-site out-degree for each document	5.2
Average on-site out-degree for each document	14.6
Average out-degree for each document	19.8

Table 6.7 : Average document outdegrees

Plotted on a graph (Figure 6.5) we can see the percentage of out-links from each document that fall into each of the two categories of links (off-site and on-site). As would be expected, the number of navigational (on-site) links far outweighs the number of judgement bearing (off-site) links, by a figure of almost 3 to 1.

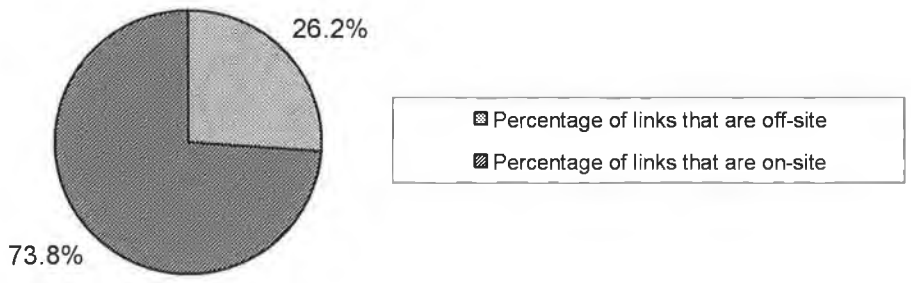


Figure 6.5 : Percentage of both on-site and off-site links

Recall from Chapter 2 that the HVV (Hyperlink Vector Voting) technique, which allows the anchor text of an in-link to a document to describe the document's content, requires the existence

of anchor text descriptions to be successful. Anchor text statistics from the random sample of documents is shown in Table 6.8.

Number of links with anchor text	79,372
Number of links without anchor text	19,207
Number of off-site links with anchor text	20,021
Number of on-site links with anchor text	59,351

Table 6.8 : Anchor Text Statistics for the random sample

As can be seen 80.5% of all links include anchor text descriptions with the remaining 19.5% lacking them (these are likely represented by images). Off-site links contain a slightly lower percentage of anchor text descriptions at 77.6% of the total while on-site links show an 81.6% positive anchor text presence.

6.2.2.4 IDENTIFYING AND REMOVING BROKEN LINKS FROM THE CALCULATION

These experiments on the 5,000-document sample provide us with our first indication of the true linkage structure of the WWW. Recall our assumption that each-and-every link on the WWW is both an out-link from one document and an in-link into another document. However, the one caveat that restricts us from simply estimating the average indegree of WWW documents to be 19.8, comprised of 5.2 off-site links and 14.6 on-site links is that we have ignored how many of these links are in fact linking to non-existent documents (broken links). We will review our findings presented above, in light of the fact that our examination of the downloaded documents identified 3,136 of the linked URLs to be broken links, in that they point at documents that do not exist. The reason for the broken links is not of interest to us, rather the fact that 3.2% of the links we parsed from the document sample were broken and these links originated from 842 separate documents. We have not included in these figures another 1,271 links that we were unable to download or process documents for, either due to not using the HTTP protocol or because of serious errors in the HTML of the web page and these links were not included in the original statistics. Were we to have included these links, we would find that a 4.4% of links were broken, not 3.2%.

Considering this information (based on the 3.2% figure) we must review our findings and present new figures (referred to as 'Revised 1') below in Table 6.9, which displays the figures for links that are working (i.e. not broken). The number of links found drops by 3,136 from our preliminary figures ('Prelim') with a corresponding drop in the number of off-site and on-site links.

	PRELIM	REVISION 1
Total number of links found	98,579	95,443
Number of which are off-site	25,813	24,580
Number of which are on-site	72,766	70,863

Table 6.9 : Examining the types of links found, excluding broken links and comparing the results to the preliminary figures

This results in an expected drop in the average outdegree from 19.8 to 19.1 (see Table 6.10) with the average off-site outdegree dropping by from 5.2 to 4.9 and the average on-site outdegree dropping from 14.6 to 14.2. Off-site links show a higher percentage drop than on-site links and this is intuitive because we would assume that the probability of a link being broken is higher for off-site links than it would be for on-site links.

	PRELIM	REVISION 1
Average off-site out-degree for each document	5.2	4.9
Average on-site out-degree for each document	14.6	14.2
Average out-degree for each document	19.8	19.1

Table 6.10 : Revised average document outdegree figures compared to the preliminary figures

We also note no notable change in the ratio of links that are off-site and on-site as shown in Figure 6.6.

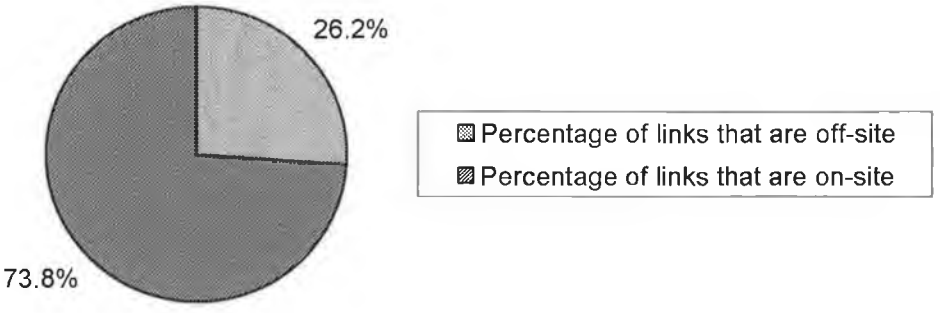


Figure 6.6 : Revised percentage of both on-site and off-site links

Our findings presented over the last number of pages based on the random sample clearly illustrate that WT_Connected seriously underestimates the link density of the WWW. WT10g has a lower link density than WT_Connected so the problems with WT10g are even more acute. Assuming that all out-links (with the exception of broken-links) also act as in-links, and the off-site / on-site separation between link types does not change after link inversion[58] then we can compare the average off-site indegree of WT_Connexted to that of the random sample (after a process of link inversion) as illustrated in Figure 6.7.

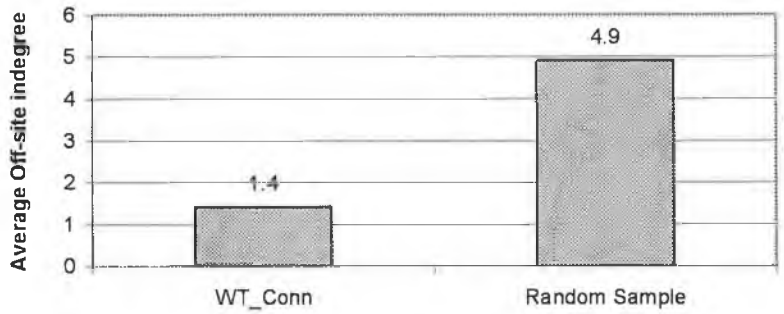


Figure 6.7 : Comparing WT_Conn to the Random Sample

6.2.2.5 FURTHER OBSERVATIONS FROM OUR RANDOM SAMPLE

During the course of our examination of the link structure of the sample set of documents, we did notice that a large number of links point to advertisement web pages, which are paid links whereby the owner of a web page earns a small amount of money each time a user follows a particular paid link. In all we found 935 (non-broken) URLs that link to advertisement web pages. An example of such a paid link is shown in Figure 6.8 where the paid link is highlighted. In all, 1% of the total number of links were paid links.

[58] Link inversion is simply the name we give to the process of altering our view of links from out-links originating from a set of documents A into in-links to a set of documents B

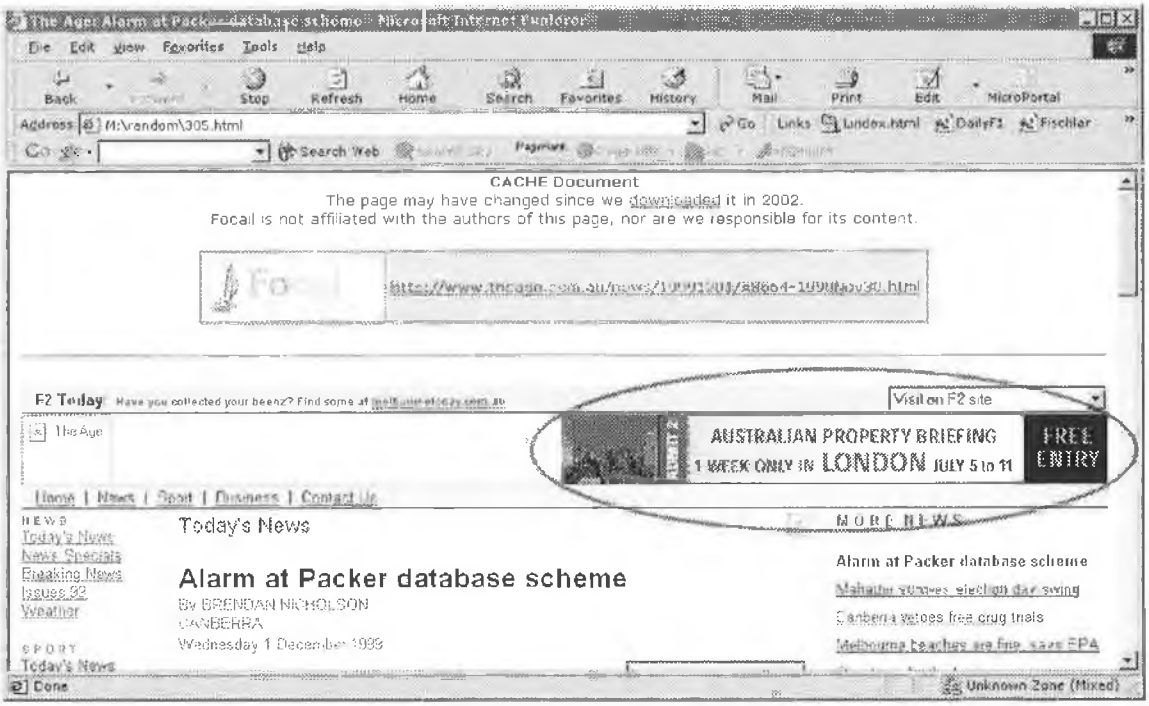


Figure 6.8 : An example of an advertisement link.

Although these paid links are valid and there is no reason for them to be removed from the link count experiments, were we to remove both the advertisement and broken links we get new linkage figures as displayed in Table 6.11. In all, 4,071 URLs were removed (4.1% of the total) which were linked to by 1009 distinct documents from the sample.

	PRELIM	REVISION 2
Total number of links found	98,579	94,508
Number of which are off-site	25,813	23,650
Number of which are on-site	72,766	70,858

Table 6.11 : Revised examination of link types when both advertising and broken links are removed, compared to our preliminary figures

This effects the average outdegree calculations as shown in Table 6.12.

	PRELIM	REVISION 2
Average off-site out-degree for each document	5.2	4.7
Average on-site out-degree for each document	14.6	14.2
Average out-degree for each document	19.8	18.9

Table 6.12 : Revised average document outdegree figures when both advertising and broken links are removed compared to the preliminary figures

6.2.3 Our Findings

A comparison between our preliminary findings and those of the SOWS III survey are shown in Table 6.13. As can be seen there is no notable difference between the results of both findings, which adds weight to the validity of our chosen sampling method. The average outdegree of each document is higher for SOWS, but the requirement of at least four out-links per document would affect these results. Our survey found that less links are broken with less than one broken link found per document as opposed to 1.3 by SOWS III. One likely explanation for this is the widespread availability of link validation tools such as CyberSpyder Link Test [CYBERSPIDER, 02] or improved web authoring tools such as Macromedia Dreamweaver [DREAMWEAVER, 02] that include built-in FTP facilities that restrict the opportunity for error.

	SOWS III	OUR SAMPLE
Sample size (number of documents)	200	5,000
Number of links	4,581	98,579
Average number of links per document	22.9	19.8
Number of dead-links	261	4407
Average number of dead-links per document	1.3	0.9
Percentage of documents containing dead-links	28.5%	16.8%

Table 6.13 : Comparing our findings to the SOWS III survey findings.

Comparing our findings to those of Cyveillance we do find a difference in the average on-site outdegree of each document, but the average off-site outdegree is very similar as can be seen from Table 6.14 which is encouraging once again for the validity of our sample. This suggests that people are creating roughly the same number of off-site links from documents now as was the case two years ago when Cyveillance published their results even though improved search engines and recent issues

regarding the legality of deep linking[59] [Wired, 02] could have been expected to have resulted in a decrease in the number of off-site links being created.

	CYVEILLANCE	PRELIM
Average off-site out-degree for each document	5.6	5.2
Average on-site out-degree for each document	23	14.6
Average out-degree for each document	28.6	19.8

Table 6.14 : Comparing our findings to Cyveillance

However, in order to extract indegree figures from this experiment, we must remove broken links from the process. If we do so, then based on our findings from the first revised figures presented in Table 6.9 and Table 6.10, we can identify that the average out-degree of a page (excluding broken links only) is 19.1, with 4.9 off-site out-site links and 14.2 on-site outLinks. Therefore, after inversion, we can state that the average document on the WWW has an off-site indegree of 4.9 and an on-site indegree of 14.2. Paid links, although not usually considered to be judgement bearing, do point to documents that do exist and thus are valid links and were not removed from the calculations.

While this may seem like too many links, one must remember that indegrees of web pages are not uniform over all web pages, rather the combined indegree of all pages are distributed unevenly over web pages, meaning that not all pages will have an off-site indegree of (in the region of) 4.9. This is clearly the case and can be seen if one takes ten popular web sites and calculates the indegree of the root page of each site (Table 6.15). We used the Google 'link:' query[60] to calculate these figures, so given that Google does not index the entire WWW it is to be expected that these figures underestimate the true indegree of these web pages.

[59] Deep Links are off-site links that bypass another site's front page, leading users directly to specific content within the site.

[60] Google supports querying the webpages that link into a particular URL by prefixing the URL with 'link:' and sending this as a query. In this way we are able to compute a lower bound on the indegree of any web page.

WEBSITE	INDEGREE
www.google.com	210,000
www.microsoft.com	132,000
www.yahoo.com	623,000
www.msn.com	143,000
www.apple.com	90,600
www.dmoz.org	595,000
www.nasa.gov	89,100
www.adobe.com	113,000
www.cnn.com	103,000
www.sun.com	87,000

Table 6.15 : The root page indegree of ten popular root web sites

The average indegree of each of these web pages is 218,570, which is many times higher than the average indegree that we have just identified by experimentation. The distribution of indegrees (and outdegrees) for web pages has been shown by experimentation and observation to approximate a power-law distribution and we will now examine if the outdegree distribution of our sample of 5,000 pages approximates a power law distribution. If so, then this adds further weight to the validity of our sample.

6.3 Distribution of Indegree amongst Documents

It has been discovered that the distribution of web page indegrees follows closely to a power-law distribution [Broder et al., 00], yet we have recently seen further evidence that the distribution does not follow a pure power-law [Pennock et al., 02], rather it can be said to approximate a power-law distribution. We are told that the "distribution of inbound links on the web as a whole is closest to a pure power-law" while "category specific distributions exhibit very large derivations from power-law scaling". This raises issues for the generation of test collections because any attempt to influence the documents comprising a dataset in order to include some category specificity will not correctly represent the natural WWW link structure. However, given that we are looking at a non category-specific collection of web pages when building a test collection we can be satisfied that the distributions of document indegrees should approximate a power-law distribution.

6.3.1 Power-law Distributions

Power-laws are used in mathematics when one wishes to relate one quantity to the power of another and can be written thus:

$$x = y^z \qquad (6.2)$$

where x is equal to the value of y to the power of z (z is the exponent of the power-law). A power-law implies that small occurrences are extremely common whereas large occurrences are extremely rare, so if applied to web page indegrees or outdegrees this means that the vast majority of web pages have a very small number of in-links (or out-links) and a few pages have a large or enormous number of in-links.

Power-law distributions are not just used to describe the indegrees of web pages (and computer science problems in general), rather they are common to both man made and naturally occurring phenomena [Mitzenmacher, 01]. From computer science we see power-law distributions in web page indegrees [Broder et al., 00], [Barabasi et al, 00] (as we have seen above), outdegrees [Faloutsos et al., 99], in the number of pages on websites [Adamic & Huberman, 01], in models for Internet growth [Mitzenmacher, 01], and even in the distributions of word frequencies in language [Zipf, 32], [Miller, 96], [Adamic, 00].

The characteristic signature of data that follows a power-law is that when the data is plotted on a log-log scale, the distribution shows itself to be linear (with the slope based on the exponent)

which is how we will identify power-law distributions. A sample synthetic plot of the (pure power-law) distribution of visitors to websites is shown in Figure 6.9. Given that this is a plot of synthetic data we would expect the correlation co-efficient to be 1.0. In out case this value was 0.9998.

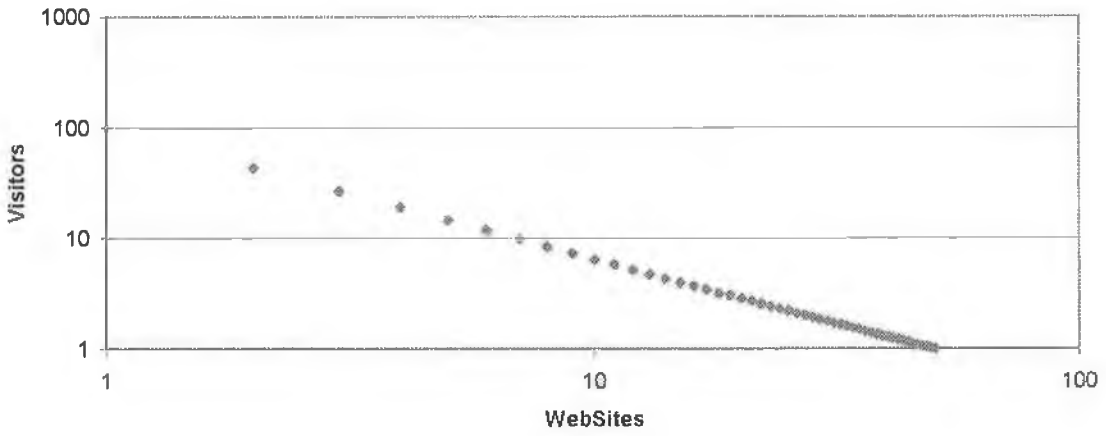


Figure 6.9 : A power-law distribution plotted on a log-log scale

If however the data is plotted on conventional (linear) scale axes then the curve of data that follows a power-law distribution would be an L shape, as shown in Figure 6.10 and is seen to hug the axes of the diagram. This is using the same synthetic data that we used in Figure 6.9.

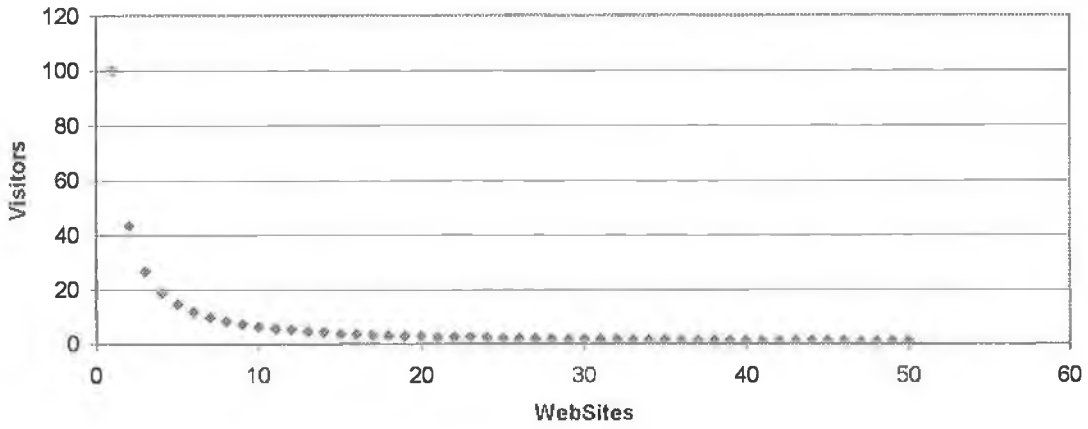


Figure 6.10 : A power-law distribution plotted on a linear scale

A simple description of a power-law distribution is that the data has:

- a few elements that score very high
- a medium number of elements with average scores
- a huge number of elements with very low scores

If we plot the outdegree distribution of our random sample of 5,000 web pages on axes of linear scale the result should be an L shaped graph as would be expected were the data to approximate a power-law distribution. This graph is shown in Figure 6.11 and as can be seen, the graph is L shaped and hugs the axes, which is the signature of data that follows a power-law distribution when plotted on linear axes.

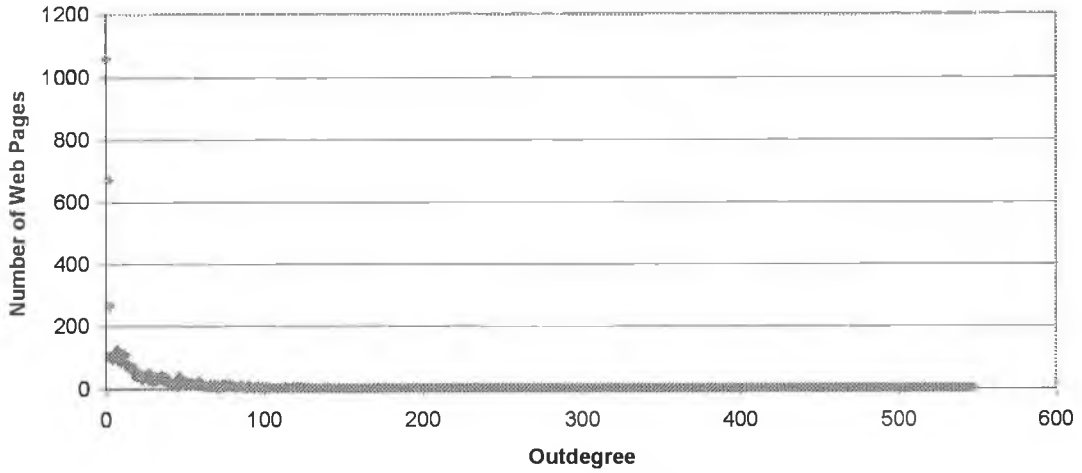


Figure 6.11 : The outdegree distribution of our random sample plotted on a linear scale

However, if we plot the outdegree distribution of our random sample on axes of log-log scale the result is shown in Figure 6.12. It is clearly visible (by examination) that this result approximates a power-law distribution (albeit with slight deviations) and this is as we would expect were our sample valid and be representative of true WWW link distributions. The included trendline has a correlation co-efficient of 0.8692.

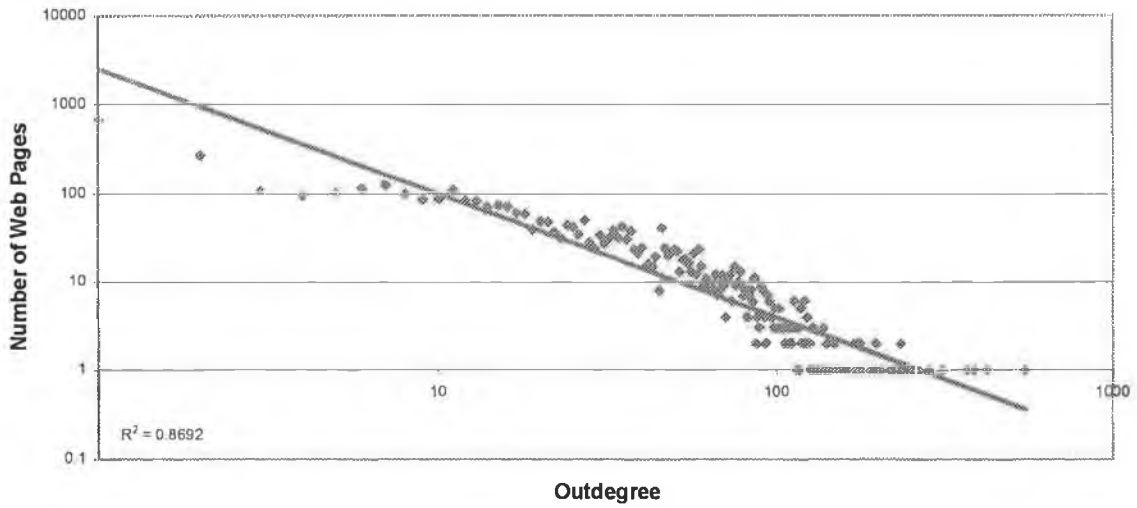


Figure 6.12 : The outdegree distribution of our random sample plotted on a log-log scale including trendline (correlation co-efficient is 0.8692)

This plot in Figure 6.12 ignores the fact that many of the links that we examined were broken links and Figure 6.13 shows a plot of the out-degree distribution for the random sample of web pages, excluding broken links. Once again, the graph is plotted on a log-log scale and we can see that the result is similar to Figure 6.12 and that the non-broken outdegree of the random sample of pages also approximates a power law.

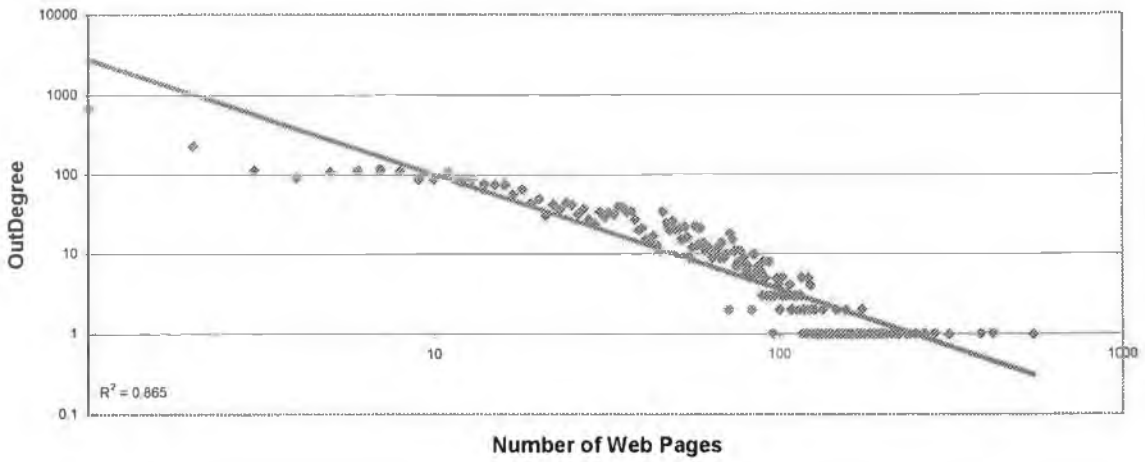


Figure 6.13 : The outdegree distribution of our random sample excluding all broken links plotted on a log-log scale, with trendline (correlation co-efficient 0.865)

Recall that we are interested in both off-site and on-site out-links and if we examine the distribution of off-site out-links in isolation[61] (ignoring that many broken links exist) we can see that this too approximates a power-law (to a greater extent) as can be seen in Figure 6.14.

[61] 'In isolation' refers to the fact that just on-site or off-site links are displayed

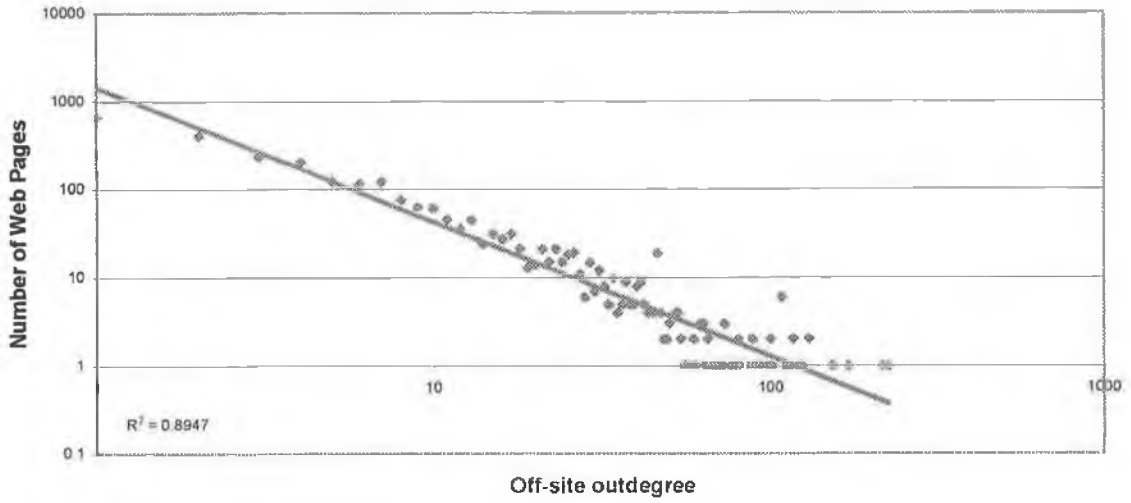


Figure 6.14 : The off-site outdegree distribution of our random sample plotted on a log-log scale with trendline (correlation co-efficient 0.8947)

Examining the on-site outdegree distribution (in isolation also) of on-site links (once again including broken-links) we can see that this also approximates a power-law distribution and is plotted in Figure 6.15.

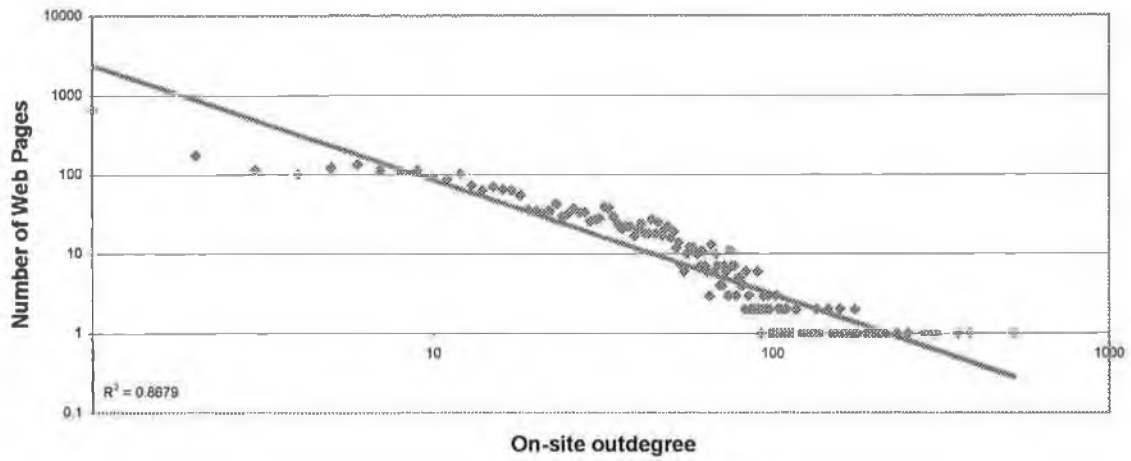


Figure 6.15 : The on-site outdegree distribution of our random sample plotted on a log-log scale with trendline (correlation co-efficient = 0.8679)

If we plot both graphs again this time removing all broken-links the findings are as expected and differ little from the plots in which all links were included. Figure 6.16 illustrates the log-log plot of the distribution of off-site outdegrees of of-site links in isolation.

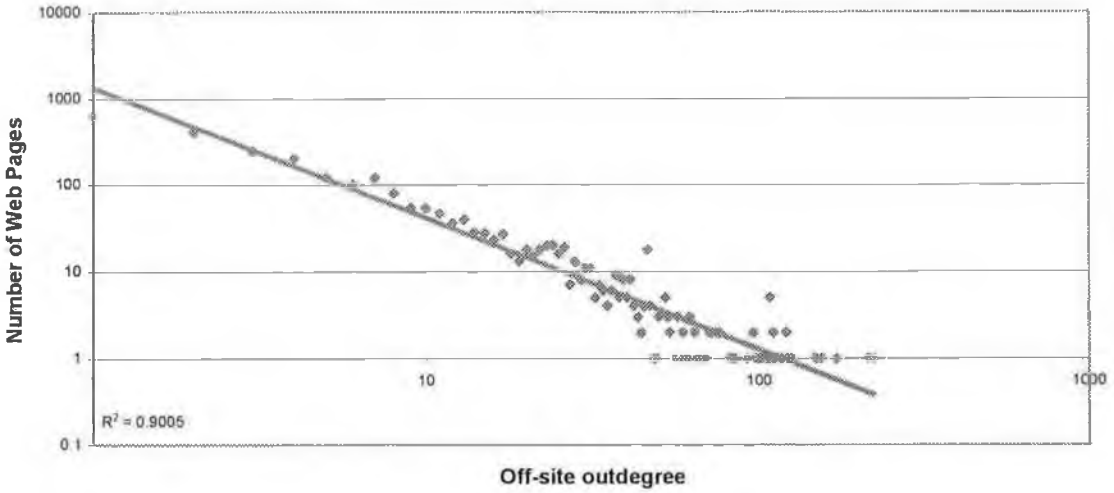


Figure 6.16 : The off-site outdegree distribution of our random sample plotted on a log-log scale with broken links removed, including trendline (correlation co-efficient = 0.9005)

Finally, the results of the distribution of on-site outdegrees of non-broken on-site links are plotted in isolation on Figure 6.17 with the expected results.

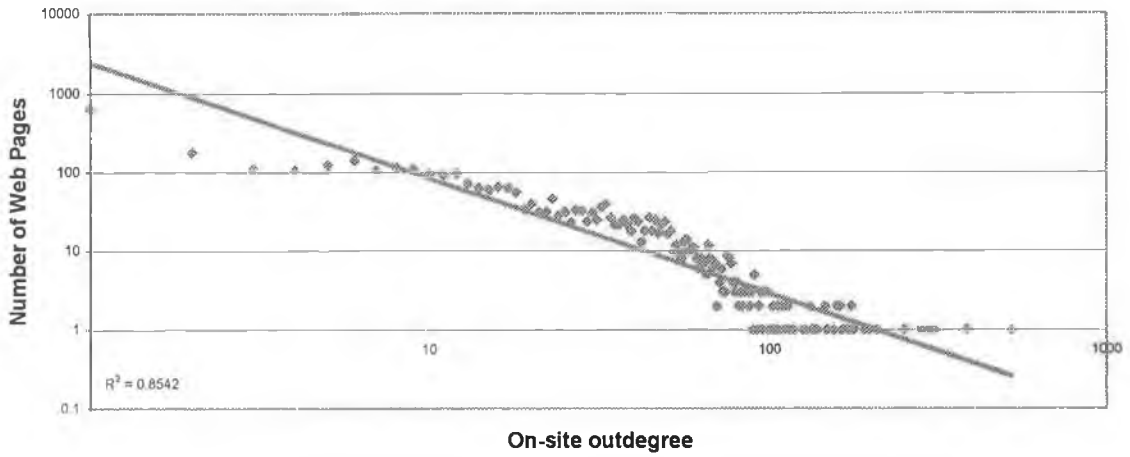


Figure 6.17 : The on-site outdegree distribution of our random sample plotted on a log-log scale with broken links removed, including trendline (correlation co-efficient = 0.8542)

By examining the distribution of outdegrees from our random sample, we have illustrated that the distribution approximates a power-law and that this is precisely what we would expect to find if our sample adequately reflected the WWW.

We know [Broder et al., 00] that document indegrees (including off-site indegrees) also approximate (or follow) a power-law distribution. Consequently, when building a dataset to support faithful experiments into linkage-based retrieval of web pages, the indegree distribution of links within the dataset should approximate a power-law distribution and our average outdegree figures indicate the link density that we would expect to find in such a dataset. We will now examine the requirements for such a dataset.

6.4 Specification of A Dataset to faithfully support Linkage-based Retrieval Experiments

As a result of our random sample of 5,000 web pages and the work of the TREC web track organisers into methods of constructing a test collection [Bailey et al., 01] and identifying that indegree distributions of web pages follow or approximate a power law we can identify the requirements for a test collection that is to support linkage-based retrieval experiments. This test collection should:

- Model real web search, by means of
    - A sufficiently large dataset [Bailey et al., 01].
    - Representative web queries [Bailey et al., 01].
    - Sufficiently complete relevance judgments [Bailey et al., 01].
    - Sufficiently high generality of the dataset so as to clearly illustrate any benefit which linkage-based retrieval techniques bring to web retrieval.
- Include the required link structure to accurately reflect the true link structure of the WWW and in so doing enable meaningful experiments into linkage-based retrieval methods. This structure can be thus summarized:
    - Must have an average off-site indegree of (or near) 4.9 based on our findings from our random sample of 5,000 web pages after we had removed all broken links.
    - Must have an average on-site indegree of (or near) 14.2.
    - The indegree distributions (both off-site and on-site) must approximate a power-law distribution with exponents (and the slope of the data on a log-log plot) capable of producing the two figures mentioned above.

We will examine each of the requirements now, starting with the dataset.

Dataset

The concept of how large is a sufficiently large dataset is an issue that we will not examine in this dissertation, except to mirror the TREC web track organisers [Bailey et al., 01] when they say that "the desire for representativness of the general web argues for a large collection, but the requirement for 'sufficiently complete' relevance judgements argues against it". Our belief is that the compromise decided upon by the TREC Web track organizers for a WT10g size dataset (1.69 million HTML documents) is sufficient and this belief is validated by the fact that the TREC 2002 Web Track test collection (the .gov[62] collection), the successor to WT10g, is no larger than WT10g. In fact, the total number of documents in the .gov collection is 1,247,753 [GOV, 02], which is actually somewhat smaller that WT10g. We have not carried out an examination of the .gov collection and we await the findings of the TREC participants who are using the .gov collection for TREC-2002.

Queries

Representative web queries are easily generated by extracting queries at random from a web query log, as was the case with TREC queries from recent years. We do however suggest that cognisance is taken of the distribution of query lengths as we have done in our experiments in Chapter 4 and as has been discussed in [Silverstein et al., 98]. We believe that it would be a mistake to simply choose web queries that are more likely to support linkage-based web retrieval (such as short or broad focus queries) as this would serve to devalue the results of the experiments. Rather, choosing the correct distribution of web queries based on their length will illustrate the benefit of techniques for combining content and linkage evidence (based on the query) such as the scarcity-abundance technique we have described in Chapter 4 and may even illustrate the need for clever methods of combining linkage and content scores which are optimized for the particular requirements of a query.

Relevance Judgements

The requirement for sufficiently complete relevance judgements is the primary issue that prohibits us from generating such an ideal dataset and running linkage-based evaluation experiments ourselves. An organization such as NIST as part of the TREC series of conferences has the resources to manually generate relevance judgements over a text collection such as WT10g. However, not even NIST have the resources to generate complete relevance judgements, rather they use the pooling technique to drastically reduce (to 4% of the full dataset) the number of documents that require human inspection. The pooling technique requires numerous diverse retrieval algorithms to generate

[62] The .gov collection has been generated from a crawl of .gov web pages, presumably restricted to .gov domains due to legal issues

the pool of candidate documents and then human inspection and judgement is required to evaluate each document in the pool. Even the pooling technique is far beyond the capabilities of our resources. For example, in TREC-9 (web track) 70,071 documents required human inspection in order to build incomplete relevance judgements. This would take a single individual evaluating 60 documents per hour, working 40 hours per week, over 29 weeks to complete the evaluation process. Had the WT10g dataset been larger, there would have been an even larger number of documents requiring human relevance judgements and this is another reason why WT10g was no larger than 1.69 million documents.

Generality

Generality is an issue that is of vital importance if we are to see significant benefits being gained by any one retrieval technique over another. Generality affects the queries chosen because we must ensure that relevant documents for each query exist in the test collection, otherwise the queries would be useless for experimentation. A certain minimal level of generality is required and we see no reason to deviate to any great extent from the WT10g level which was 0.00155 which infers that 0.155% of all documents in the dataset have been (manually) found relevant (using the pooling technique) to any one of the fifty web queries that comprised part of the test collection. It is also of utmost importance that when generating a dataset that one does not choose documents for the dataset based on document content being related to some particular topic. In other words, the dataset must dictate possible topics as opposed to the topics dictating what documents comprise the dataset. Recall that we are told that the 'category specific (indegree) distributions exhibit very large derivations from power-law scaling' from [Pennock et al., 02], therefore, in order to accurately evaluate linkage-based retrieval we must accurately recreate the linkage structure of the WWW, and category specific datasets would not approximate a natural power-law distribution of link densities.

Link Structure

The second and final set of requirements for a faithful representation of the WWW is that we must accurately recreate the link structure of the WWW, which we have discussed at length in this and the previous two chapters. Previous datasets such as WT10g have failed in this requirement and we can see that accurately recreating the linkage structure of the WWW was not one of the top priorities when generating the WT10g dataset as we are told that they (the TREC web track organizers) wished to construct a corpus which "contained many inter-server links (to permit meaningful experimentation with hyperlinks)" [Bailey et al., 01] as opposed to aiming to accurately recreate the linkage structure of the WWW. Our findings have shown that documents within the

dataset must have an average off-site indegree of (or near) 4.9, an average on-site indegree of (or near) 14.2 and that the indegree distributions (both off-site and on-site) must approximate a power-law distribution. We believe that linkage-based experiments on such a dataset will clearly illustrate the benefit to be gained from linkage-based retrieval techniques such as SiteRank over conventional content-only techniques.

6.4.1 How to Generate a Faithful Test Collection

We have mentioned earlier that test collection queries can be extracted from query logs to match length distribution figures, but the question of how to generate a dataset of documents of suitable size with a representative link structure for use in WWW information retrieval experiments is not as straightforward. So let us examine a few options:

- If we crawl all the WWW then we will obviously mirror exactly the link structure, but this is not an option as not all documents can be reached by following the linkage structure of the WWW and in any case, this would be far beyond the resources of most research groups.

- If we randomly sample 1.5 million pages then we will get pages with a representative outdegree structure, but unless the pages that are linked to are also in the dataset (chosen randomly also) then this out-degree structure is meaningless so this is unlikely to succeed. This does assume a source of random URLs, although the technique used by SOWS III integrated with the Google search engine would probably suffice for this purpose.

- Simply randomly sampling a number of pages again (123,000[63] will suffice) and adding in all documents referenced by these documents will produce a dataset of almost 1.5 million documents in size. However, the linkage structure would only be based on the 123,000 randomly chosen documents (which will mostly have an off-site indegree of 0, as we have seen from the Gaeilge crawl) and our findings presented in the previous chapters would indicate that the majority of the other documents (at least 63.8%) would only have an indegree of 1. We would still need to remove some documents from the dataset to produce

[63] 123,000 web pages will contain non-broken links to 1,498,853 unique documents based on the findings of our random sample, which shows that each document (on average) contains 19.1 non-broken links to other documents, but we have taken into account problems caused by duplication of linked URLs. From our random sample, over 35% of links did not identify new (never before seen) URLs.

1.5 million documents, but care must be taken to retain the representative link structure at each removal (or addition) of a document from the dataset, which is by no means a trivial task.

- We could send out a web crawler to download 1.5 million pages from the WWW. In Chapter 5 we have illustrated that simply sending out a web crawler using a queuing algorithm that weights off-site links higher than on-site links will overestimate the number of off-site links and for that reason a test collection based on this dataset will not be representative, however a subset of it might. One could send out a simple FIFO queue (breadth first search) and gather web data this way. Once again, however, only out-links will be extracted and the problem with the distribution of in-links of the previous point applies once again here. In addition, by using a web crawler (based on some queuing algorithm) one is more likely to find oneself stuck in local small highly connected components. If one kept crawling the WWW, periodically examining the link structure of the crawled data until the desired link structure is found, then all control over the size of the final dataset is relinquished.

The only realistic alternative to crawling the WWW and stopping when one has attained the desired linkage structure and a minimum number of documents is to follow the approach taken by the TREC Web track organizers when fabricating WT10g. As we have discussed in Chapter 3 (Figure 3.10), the WT10g corpus is a subset of the 100GB VLC dataset, which is itself a subset of a 300 GB Internet Archive crawl completed in early 1997. WT10g, we are told [Bailey et al., 00] "was constructed by selecting from a superset of documents in such a way that desirable corpus properties were preserved or optimised". We have already discussed our desirable corpus properties above.

The TREC web track organisers employed a four-phase process [Bailey et al., 01] when generating the WT10g dataset. These phases were:

1. Choice of superset, which was the 100GB VLC2 subset of an Internet Archive crawl from 1997, chosen due to time constraints and the "time consuming nature" of certain phases of the analysis process. This superset was not chosen due to any particular linkage structure of the documents contained therein, rather it was readily available.

2. Rejection of unwanted data, based on removing non-English documents, eliminating duplicate documents (based on checksums) and removal of documents whose URL did not end with a .html (and variants) or .txt.

3. Characterisation of servers, which involved the selection of a set of requirements requirements that servers must meet in order to be ranked for selection for the final dataset.

4. Selection of WT10g servers and pages from servers based on server characterisation measures discovered in the previous phase which include; server size distribution, in-link and out-link densities and presence of homepages and relevance to 10,000 large web task queries.

We recommend a similar phased process, but in our case, we foresee only three phases to be necessary given that we already know the required characterisation of the dataset from our experiments presented previously. Our suggested phases are as follows:

1. Generate a superset of documents, most probably by a process of crawling the WWW, although were an alternative source of data of suitable size with a rich link structure to be found then this may be sufficient. This superset must contain a sufficient quantity of links between documents to allow the extraction of a subset of 1.5 million documents with their associated linkage structure (7.23 million off-site links and 21.3 million on-site links) distributed over all documents so that a power-law distribution is approximated. This will allow for the extraction of a component of the superset that will accurately recreate the link structure of the WWW. The exact size of this superset is dependent on the queuing algorithm employed by the web crawler as this will dictate the link density of the superset. As we have seen, if phase 1 produces a superset (for example, the 100 GB collection) that is incorrect (with respect to our requirements) then the whole process will fail to produce a representative subset. We feel that this area is worthy of additional research.

2. A certain amount of unwanted data will have to be rejected such as the elimination of binary or duplicate documents. This may influence the size of the superset created in phase 1, so one must consider this and be prepared to prune the superset prior to executing phase 3 (below).

3. Finally, generate a subset of the required size (we suggest 1.5 million documents), extracted from the superset, along with the links between these documents, which should be 7.23 million off-site links and 21.3 million on-site links. These links should be distributed to approximate a power-law distribution. All candidate documents must be weighted for selection with respect to their value within the subset and therefore these values must be calculated based on the documents that already have been selected for the dataset. Taking

such an approach, documents are judiciously added from the superset to the subset until the required properties of the subset have been realised. We refer the reader to [Bailey et al., 01] for a description of a similar a process, which realised different goals.

We cannot hope to develop the dataset with the representative link structure ourselves and subsequently develop the incomplete relevance judgements to support meaningful experiments. Even generating the pool of relevant documents in order to generate incomplete relevance judgements requires many diverse retrieval algorithms and techniques, each of which produces ranked lists of highly scored documents for each query. In TREC-9, a total of 23 research groups participated in the Web track [Voorhees & Harman, 00] and these groups produced 59 different sets of results that were added to the pool. This means that possibly 59 diverse retrieval algorithms produced these results, which one single research group would be unable to replicate within any reasonable resource constraints. The diverse nature of these retrieval algorithms is clearly highlighted by the fact that only the top 100 ranked documents of each algorithm were added to the pool, yet the mean actual pool size for each query [Hawking, 01] was 1,401. In addition, it has been found [Zobel, 98] that the quality of the pools used (based on the diversity of systems contributing and the number of scored documents taken from each experiment) do affect the quality of the resulting test collection. Therefore were one single research group to attempt to generate pools of documents required for incomplete relevance judgements then many diverse retrieval systems would need to be designed, developed and deployed in order to generate a quality pool of documents from which incomplete relevance judgements can be distilled. This would be beyond the resources of most research groups.

Recall from earlier that one of the requirements when building a test collection to support experiments into WWW information retrieval is that the test collection must include the required and appropriate type of link density to enable meaningful experiments into linkage-based methods. We have shown that the TREC test collections do not faithfully reproduce the actual linkage structure of the WWW and have identified the linkage requirements that any such test collection should adhere to. However, it is our belief that in order to generate such a dataset (of adequate size) would require crawling a dataset many times larger and extracting a densely linked subset approximating a power law distribution which is beyond the scope of both our resources. We simply do not have the available resources to crawl a large dataset, which accurately recreates the linkage structure of the WWW (in order to support faithful experiments into linkage-bases retrieval), and generate (even) incomplete relevance judgements. Even the TREC web track organizers only processed the 100GB VLC2

collection when generating WT10g and not the (readily available) 320GB superset of the VLC2 due to the "time consuming nature of certain phases of the analysis".

It is for reasons like this that the TREC series of conferences is so valuable to the research community as it takes the tedious, time-consuming and very expensive work out of the process of evaluating retrieval performance. While it would be impossible for a participating group such as ourselves to build the dataset and gather queries with relevance judgements, by operating under the umbrella of TREC, we are given an opportunity to evaluate algorithms in an environment that models the real world (albeit in a limited way sometimes) to which we would otherwise not have the resources to replicate ourselves.

6.5 SUMMARY

A test collection to support experiments into retrieval of documents from the WWW has an associated list of requirements, which include:

- to model real WWW search by means of a sufficiently large dataset, representative web queries and sufficiently complete relevance judgements.
- to include the required and appropriate type of link density.

In order to examine the required link density and the appropriate type of link density that would be required to fulfill the above requirements, we sampled 5,000 web pages chosen at random from the WWW and examined the out-link structure of these web pages. Our findings suggested that the average web page has an average off-site outdegree of 5.2 and an average on-site outdegree of 14.6. However, when we revised these figures to remove broken links from the sample we found that the average off-site outdegree dropped to 4.9 with a corresponding decrease in average on-site outdegree to 14.2. This clearly illustrated that the actual link densities and type distribution on the WWW are not reflected in either WT_Connected or WT10g, although WT_Connected is far closer to the real WWW.

Based on these figures we could state that the average web page has an off-site indegree of 4.9 and on on-site indegree of 14.2. However, these averages are not distributed uniformly over all web pages. It has been found that the indegree and outdegree distribution of web pages approximates

a power-law distribution and our random sample is no exception. Therefore, we were in a position to state that a test collection to faithfully support linkage-based retrieval experiments must have:

- an average indegree figures as just mentioned above.
- this indegree distributed according to an approximate power-law distribution.
- a dataset of suitable size (1.5 million documents seems sufficient).
- representative web queries, perhaps from a query log, which mirror the discovered distribution of query lengths.
- relevance judgements which are sufficiently complete for the dataset and queries.

We have shown that link densities on the WWW are not reflected in either WT_Connected or WT10g, although WT_Connected is far closer to the real WWW and we have developed a set of requirements for a test collection, which could faithfully support linkage-based retrieval experiments.

We have also discussed some methods of generating such a test collection, but concluded that the only sure method was to gather a large superset of documents and extract a subset, which becomes the dataset and fulfills the requirements outlined above. However, this is as far as we can take this research. We simply do not have the available resources to crawl a large dataset (and likely extract a subset), which accurately recreates the linkage structure of the WWW (in order to support faithful experiments into linkage-bases retrieval), and generate (even) incomplete relevance judgements.

Besides simply confirming our inability to create a TREC-like collection, we have provided the research community with a path to follow, which we firmly believe is the next logical step in experimenting with linkage analysis techniques. With WT_Connected we have shown that increasing linkage densities can improve retrieval performance, albeit only slightly. For too long the linkage-based IR field has been metaphorically stuck on the starting line, trying to evaluate algorithms with much potential, but on test collections that were always going to cause experiments to fail. With the findings of the research presented in this thesis, we have dropped the flag and hopefully now the race can commence, a race to develop and successfully evaluate new and better algorithms than PageRank, algorithms to power new and even better search services than Google.

Chapter 7

CONCLUSIONS AND FUTURE RESEARCH OPPORTUNITIES

7.1 CONCLUSIONS FROM THIS RESEARCH

In recent years, the widespread use of the WWW has brought information retrieval systems right into the homes of many people. We are a very fortunate generation, in that we have access to many billions of documents (web pages) and have (free-of-charge) access to powerful, fast and highly efficient search facilities over these documents provided by search engines such as Google [GOOGLE, 02] or Teoma [TEOMA, 02]. In the early days of publicly accessible search engines (all of 9 years ago), search engines were designed using conventional term weighting strategies and were based on directly computing the similarity between a query and the text appearing in a web page. The term weighting strategies implemented were likely based on the Vector and Probabilistic models of IR, which we have discussed in Chapter 1. While these initial "first generation" of web search engines addressed the engineering problems of web spidering and efficient searching for large numbers of both users and documents, they did not innovate much in the approaches taken to searching.

However, the last five years have shown that we can extract additional latent information from web documents, which we believe can be used to aid retrieval performance of web search engines. This latent information is mined from the ubiquitous hyperlink and the exploitation of this latent information is called linkage analysis. Anecdotally, linkage analysis appears to have improved retrieval effectiveness of web search, yet there is little scientific evidence in support of the claims for better quality retrieval, which is surprising. Participants in the three most recent TREC conferences (1999, 2000 and 2001) have been invited to perform benchmarking of information retrieval systems on web data and have had the option of using linkage information as part of their retrieval strategies. The general consensus from the experiments of these participants is that linkage information has not yet been successfully incorporated into conventional retrieval strategies.

The goal of the TREC series of conferences is to foster research into the information retrieval and encourage participants to take part in retrieval benchmarking experiments in a spirit of openness and knowledge sharing. The evaluation methodologies employed for the TREC conference

experiments are the standard measures of Precision and Recall which are used in conjunction with test collections. Test collections have a long tradition of use in IR and they comprise documents, queries and relevance judgements and are used in large-scale retrieval experiments. TREC has supported experiments into WWW IR for the TREC-8, TREC-9 and TREC-2001 web tracks (with 2002 not having taken place yet). The aim of these web tracks were to provide a framework within which participating research groups could come together and evaluate their retrieval techniques, especially linkage-based techniques on a test collection of over 1.69 million web documents (for TREC-9 and TREC-2001).

There are three generally accessible test collections for the evaluation of retrieval of Web documents (WT2g, WT10g and VLC2) and these were available to participants in the TREC series of conferences. We have presented in this thesis experiments based on our own linkage-based retrieval strategies which have been evaluated on both WT2g and WT10g. Our TREC-8 experiments [Gurrin & Smeaton, 99] into linkage–based retrieval were based around citation ranking and were unsuccessful at improving retrieval performance (over conventional content retrieval) using the ¼ million document WT2g dataset. This finding was mirrored by all other participants at TREC-8, which was a surprise to many as anecdotally it was felt that linkage analysis improved retrieval performance over conventional content-only retrieval.

For the TREC-9 conference, we developed more advanced algorithms based on citation ranking, spreading activation and co-citation analysis incorporating spreading activation [Gurrin & Smeaton, 00]. The test collection provided by TREC was a much larger test collection called WT10g which comprised 1.69 million documents, which we have examined in detail. Our findings from our TREC-9 experiments showed that linkage-based retrieval was again of no benefit to retrieval performance and these findings (once again) were shared by other participating groups. However, our belief was that the test collection was not capable of faithfully supporting linkage-based retrieval experiments.

As part of a research internship in AT&T research labs we developed the SiteRank linkage-based retrieval algorithm to a specified requirement for an algorithm based on PageRank, but which propagated rank between web pages by taking a website-centric view of the process. With the conventional PageRank algorithm, it is possible to artificially increase the rank of a web page by creating a clever synthetic linkage structure surrounding that page. The SiteRank algorithm that we developed and present as part of this research helps eliminate this problem by limiting the influence

of web pages from any one web site, thus making it more difficult and more expensive to artificially increase a web page's rank in this manner. In our experiments (on a different test collection described below), SiteRank slightly outperforms PageRank, but SiteRank's main augmentation to PageRank is that it was developed to help combat the problem of search engine persuasion and due to the nature of the underlying dataset, we were not in a position to specifically test SiteRank's search engine persuasion defeating properties.

Given our belief that neither WT2g nor WT10g (both of which included incomplete relevance judgements) were capable of supporting linkage-based retrieval techniques (including SiteRank) we extracted a densely linked subset from WT10g which we called WT_Connected to support our experiments. WT_Connected maximised the number of and density of off-site links, as much as we could given that WT10g was our source. A number of new experiments based on our previous TREC experiments (for both TREC-8 and TREC 9), SiteRank and PageRank were executed on this new dataset. In addition, we evaluated a new method of combining linkage and content evidence together to produce a final ranking. Prior to this the most widely used method was to incorporate best guess parameters into the process, or an alternative technique which regulated linkage influence based on the number of terms in the query. Our technique was based on the size of the result-set of highly scored documents and operated on the assumption that a larger result-set signifies a broader query and this requires higher linkage influence and vice-versa. However, any improvements shown by this technique are not significant and additional experiments would be beneficial on a new test collection.

Our findings from these experiments illustrate that it is possible to gain moderate improvements in retrieval performance when running experiments using standard TREC evaluation procedures and measurements on WT_Connected as opposed to WT10g. These findings are discussed in Chapter 4. Hence, we can conclude that it is possible to show retrieval performance improvements using conventional TREC evaluation methodologies if the underlying test collection is better capable of supporting experiments into linkage-based retrieval. By this, we mean that the test collection must contain a dataset with a suitable density and type of links between documents. The question remained, what is a suitable link density and what types of links are suitable and in what quantities? In an attempt to answer these questions we made three crawls of live WWW data.

The first crawl was an Irish language specific crawl which downloaded 26,798 documents in the Irish language, each of which had an average off-site indegree of 1.55 which was quite similar to

the average off-site indegree figure of WT_Connected which was encouraging for the results of our experiments as presented in the previous chapter. However, our intuition suggested that the distribution of off-site indegrees across the 26,798 documents was not uniform and reflected the choice (and quantity) of seed URLs that we made. Our belief was that the true off-site indegree of documents was actually being underestimated by the Irish language crawl, and by WT_Connected as well.

The second and third crawls were more conventional crawls of WWW data, in that they were not language specific and no restrictions were placed on the crawler's movements when gathering documents, save the queuing algorithm employed and any restrictions from the crawler's adherence to the robots exclusion standard. The findings of these two crawls (264,794 and 126,996 documents in size) suggests that the figure of 1.55 off-site links into each document as found by the Irish language crawl was indeed an underestimation of the real nature of WWW data. These crawls produced average off-site indegree figures of 16 and 21 respectively, however, when these figures are adjusted to take into account irregularities we found in the link structure of the crawled data the figures dropped to 10 and 12.6 respectively. Neither of these figures were similar to the figures for both the Irish language crawl and WT_Connected. We refer the reader to Figure 5.10, 5.11 and 5.12 as well as Table 5.15 for a comparison of the findings from the web crawls.

Given that the three crawls produced different average off-site indegree figures, we were not in a position to conclude that WT_Connected is actually representative of true WWW linkage structure. Rather, more experimentation was needed so we undertook to generate a random sample of web pages in order to identify the actual link structure of the WWW.

We sampled 5,000 web pages chosen at random from the WWW, and examined the out-link structure of these web pages. Our findings suggested that the average web page has an average off-site outdegree of 5.2 and an average on-site outdegree of 14.6. However, when we revised these figures to remove broken links from the sample we found that the average (operational) off-site outdegree dropped to 4.9 with a corresponding decrease in average on-site outdegree to 14.2. This clearly illustrated that the actual link densities and type distribution on the WWW are not reflected in either WT_Connected or WT10g, although WT_Connected is a far closer to the real WWW.

Based on these figures and since each link has both a source and a target document, we concluded that the average web page has an off-site indegree of 4.9 and on on-site indegree of 14.2. This clearly illustrated that the actual link densities and type distribution on the WWW are not

reflected in either WT_Connected or WT10g, although WT_Connected is far closer to the real WWW. However, these average web page indegree figures are not distributed uniformly over all web pages. It has been confirmed that the indegree and outdegree distribution of web pages follows (or approximates) a power-law distribution and our random sample is no exception.

As a result of our random sample of web pages we have developed a set of requirements for any test collection which is expected to support linkage-based retrieval experiments. Such a test collection must have:

- an average off-site indegree of 4.9 and an average on-site indegree of 14.2.
- these indegrees distributed according to (or approximating) power-law distributions.
- a dataset of suitable size (1.5 million documents seems sufficient).
- representative web queries, perhaps from a query log, which mirror the actual distribution of search engine query lengths.
- relevance judgements which are sufficiently complete for the dataset and queries.

We have also discussed some methods of generating such a faithful test collection, but concluded that the only sure method was to gather a large superset of documents and extract a subset, which becomes the dataset and fulfills the requirements outlined above. However, this is as far as we can take this research. The next step, which is the creation of the dataset and relevance judgements, will be up to an organization such as NIST because generating a test collection is an extremely resource hungry activity and it requires the resources of organisations such as NIST to co-ordinate large-scale retrieval experiments using test collections as is the case with the annual TREC series of conferences. For example, to generate a test collection as described above one would have to crawl a large superset of documents and distill from this superset a smaller set of documents that eventually becomes the dataset. In addition, relevance judgements must be constructed and we have illustrated that this is beyond the resource capabilities of a small research group as we estimate that it would require (assuming a number of sources of documents for pooling) 29 man weeks to generate incomplete relevance judgements for 50 queries on a test collection similar in size to WT10g.

Besides simply confirming our inability to create a TREC-like collection, we have provided the research community with a path to follow, which we firmly believe is the next logical

step in experimenting with linkage analysis techniques to aid retrieval performance. With WT_Connected we have shown that increasing linkage densities can improve retrieval performance, albeit only slightly.

7.2 Techniques for Combining Linkage and Content Evidence at Query Time

Although the improvements shown by our scarcity-abundance technique for regulating linkage influence outperforms best-guess parameters regulation technique at rank 5 and 10 documents, and is at least equal to the best-guess parameter values at all levels until 30 documents, the performance improvement is only just over 2% which is not large enough to prove the benefit of the scarcity-abundance technique. However, we believe that this area is worthy of additional research. Our intuition suggests that a two-phase process would be required to provide effective retrieval facilities over web data. The first phase would be to execute the query and obtain a result-set, then based on the result set, identify an optional ranking formula for that query (incorporating a number of sources of evidence). Phase 2 would be to rank the documents from the result-set from the first phase using this optimal ranking formula. We have only examined situations in which two sources of evidence are available (content and linkage), however, the technique may be expanded to incorporate other sources of evidence (such as media metrics or click-through rates). This could be visualised as each query being mapped onto a point in an n dimensional 'query space', where n is the number of sources of evidence available, and the point inferring the ranking formula.

There are other likely techniques for combining linkage and content evidence as well. For example, the best-guess parameter figures we have outlined would require tuning, but all experiments must be executed on a test collection that faithfully models the WWW and is capable of supporting such experiments.

References

[ABOUT, 02] The about.com WebSite. Available online at URL: http://www.about.com/. (last visited July 2002).

[Achlioptas et al., 01] Achlioptas, D., Fiat, A., Karlin, A., McSherry, F. "Web Search Via Hub Synthesis", IEEE Symposium on Foundations of Computer Science (accepted subject to revision), 2001.

[Adamic & Huberman, 01] Adamic, L. and Humberman B. "The Web's Hidden Order". Communications of the ACM, Vol. 44, No. 9, September 2001.

[Adamic, 00] Adamic, L. "Zipf, Power-laws, and Pareto - a ranking tutorial", Available online at URL: http://www.hpl.hp.com/shl/papers/ranking/. (last Visited July 2002).

[Agosti, 00] Agosti, M. "Information Retrieval on the Web". 3rd European Summer School in Information Retrieval, Italy, 2000. Also published as Lectures on Information Retrieval, p.242-285, Springer 2000.

[Aitchison & Browne, 54] Aitchison, J. and Brown, J. "On Criteria for Description of Income Distribution", Metroeconomica, 1988.

[Albert et al., 99] Albert, R., Jeong H. and Barabasi, A. "Diameter of the world wide web", Nature 401, 1999, 130-131.

[ALLTHEWEB, 02] The AllTheWeb Search Engine Available online at URL: http://www.alltheweb.com/. (last visited July 02).

[ALTAVISTA 02] The AltaVista Search Engine Available online at URL: http://www.altavista.com/. (last visited July 02).

[Amento et al, 00] Amento, B., Terveen, L. and Hill, W. "Does 'Authority' mean quality? Predicting Expert Quality Ratings of Web Document", Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in IR, 2000.

[APACHE] The Apache Open Source HTTP Server, Available online at URL: http://www.apache.org/ (last visited July 2002)

[AT&T, 02] AT&T Research Lab Website, Available online at URL: http://research.att.com) last visited July 2002)

[Baeza-Yates & Ribeiro-Neto, 99] Ribeiro-Neto, B. and Baeza-Yates, R. "Modern Information Retrieval". Addison-Wesley. 1st edition, 1999

[Bailey et al., 01] Bailey, P., Craswell, N. and Hawking, D. "Engineering a multi-purpose test collection for Web retrieval experiments (DRAFT)". Accepted, subject to revision, by Information Processing and Management, 2001.

[Barabasi et al, 00] Barabasi, A,. Albert, R., Jeong, H. and Bianconi, G., "Power-Law Distribution of the World Wide Web". SCIENCE, Vol 287, 2000.

[Berry & Browne, 99] Berry, M. and Browne, M. "Understanding Search Engines Mathematical Modeling and Text Retrieval" 1st edition. Society for Industrial and Applied Mathematics, Philadelphia, 1999. pg 40-41.

[Bharat & Henzinger, 98] Bharat K. and Henzinger M. "Improved Algorithms for Topic Distillation in a Hyperlinked Environment". Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in IR, 1998.

[Bharat et al., 98] Bharat K., Broder A., Henzinger M., Kumar P. and Venkatasubramanian S., "The Connectivity Server: fast access to linkage information on the web". Proceedings of the 7th International WWW Conference, 1998.

[Botafogo et al, 92] Botafogo, R.A., Rivlin, E., Shneiderman, B. "Structural Analysis of Hypertext: identifying hierarchies and useful metrics". ACM Transactions on Information Systems, 10(2), 1992.

[Boughanem et al., 99] Boughanem M., Julien C., Mother J. and Soule-Dupny C., "Mercure at trec8: Adhoc, Web, CLIR and Filtering tasks". Proceedings of the 8th Annual TREC Conference, pg 431-444. November 16-19, 1999.

[Brin & Page, 98] Brin S. and Page L., "The Anatomy of a Large-Scale Hypertextual Web Search Engine", Proceedings of the 7th International WWW Conference, 1998

[Broder t al., 00] Broder A., Kumar R., Maghoul, F., Raghavan P., Rajagopalan S., Stata R., Tomkins A. and Weiner J. "Graph Structure in the Web". Proceedings of the 9th International WWW Conference, 2000.

[Bush, 45] Bush V. "As We May Think". The Atlantic Monthly, July, 1945, Volume 176, No. 1, pg 101-108.

[Chakrabarti et al, 98] Chakrabarti S., Dom B., Gibson D., Kleinberg J., Raghavan P. and Rajagopalan S. "Automatic resource compilation by analysing hyperlink structure and associated text". Proceedings of the 7th Annual WWW Conference, 1998.

[Chakrabarti et al, 99] Chakrabarti S., van den Berg M. and Dom B. "Focused crawling: a new approach to topic-specific Web resource discovery" Proceedings of the 8th International WWW Conference, 1999.

[Cho et al., 98] Cho J., Garcia-Molina H. and Page L., "Efficient crawling through url ordering". Proceedings of the 7th Annual WWW Conference, 1998.

[Clarke & Kelly, 98] Clarke, D. and Kelly P. "Automatic Language Identification" Available online at URL: http://www.computing.dcu.ie/Projects/1998/dclark.ca4/ (last visited July 2002).

[Cleverdon at al., 66] Cleverdon C., Mills J. and Keen E. "Factors determining the Performance of Indexing Systems". Aslib-Cranfield research project, Cranfield, 1966.

[Codd, 70] Codd E. "A Relational Model of Data for Large Shared Data Banks". Communications of the ACM, Vol. 13, No. 6, June 1970, pg 377-387.

[COMAP, 96] "The Principles and Practice of Mathematics", 1st edition. Springer Verlag, New York, 1997, pg 304-331.

[Cooper, 88] Cooper, W. "Getting Beyond Boole" Information Processing & Management, 24, pg 243-248.

[CYBERSPIDER, 02] CyberSpider Link Test software, Available online at URL: http://www.cyberspyder.com/cslnkts1.html. (last visited July 2002).

[Cyveillance, 00] Cyveillance Corporation WebSite. Available online at URL: http://www.cyveillance.com. (last visited July 2002)

[DEVIANTART, 02] The DeviantArt Website. Available online at URL: http://www.deviantart.com. (last visited July 2002).

[DREAMWEAVER, 02] MacroMedia DreamWeaver WebPage Authoring Software WebSite, Available online at URL: http://www.macromedia.com/software/dreamweaver/. (last visited July 2002).

[Duff et al., 89] Duff I., Grimes R. and Lewis J. "Sparse matrix test problems". ACM Transactions on Mathematical Software, pg 1-14, 1989.

[EXCITE, 02] The Excite Search Engine. Available online at URL: http://www.excite.com. (last visited July 2002).

[Fabrikant et al., 02] Fabrikant A., Koutsoupias E. and Papadimitriou C. "Heuristically Optimized Trade-offs: A New Paradigm for Power Laws in the Internet" (Extended Abstract for STOC 02).

[Faloutsos et al., 99] Faloutsos, M., Faloutsos, P., Faloutsos, C. "On Power-Law Relationships of the Internet Topology". Proceedings of ACM SIGCOMM 99, 1999.

[Faloutsos & Christodoulakis, 87] Faloutsos C. and Christodoulakis S. "Description and Performance Analysis of Signature file Methods for Office Filing". AAM Transactions on Office Information Systems, Vol. 5, No. 23, July 1987, pg 237-257.

[Fox, 83] Fox, E., "Characterization of two new experimental collections in computer and information science containing textual and bibliographic concepts". Technical Report 83-561, Cornell University, 1983.

[FRONTPAGE] Microsoft FrontPage WebPage creation software. Available online at URL: http://www.microsoft.com/frontpage/ (last Visited July 2002)

[Fujita, 98] Fujita S., "Reflections on 'Aboutness' TREC-9 Evaluation Experiments at Justsystem", Proceedings of the 9th Annual TREC Conference", November 16-19, 2000.

[Fuller et al., 99] Fuller M., Kaszkiel M., Kimberley S., Zobel J., Ng C., Wilkinson R. and Wu M. "The RMIT/CSIRO Ad Hoc, Q&A, Web, Interactive, and Speech Experiments at TREC 8" Proceedings of the 8th Annual TREC Conference, November 16-19, 1999.

[Garfield, 01] Garfield E. "History of Citation Indexing, The ISI® Essays, 2001, Available online at URL: http://www.isinet.com/isi/hot/essays/ (last Visited July 2002).

[Garfield, 64] Garfield E. "Citation Indexes for Science: A New Dimension in Documentation through Association of Ideas". Readings in Information Retrieval, pg 261-274. The Scarecrow Press, Inc. 1964. Reprinted in Essays of an Informaiton Scientist, Vol 6, pg 468-471, 1983.

[Garfield, 94] Garfield E. "Using the Impact Factor, Current Contents". July, 1994. Available online at URL: http://www.isinet.com/isi/hot/essays/journalcitationreports/8.html.

[Garfield_2 01] Garfield E. "From Bibliographic Coupling to Co-Citation Analysis via Algorithmic Historio-Bibliography". Presented at Drexel University, Philadelphia, PA., November 27, 2001. Available online at URL: http://garfield.library.upenn.edu/papers/drexelbelvergriffith92001.pdf.

[GEOCITIES, 02] WebSite Hosting Facility. Available online at URL: http://www.geocities.com. (last visited July 2002).

[Goldszmidt & Sahami, 98] Goldszmidt, M. and Sahami, M. "A Probabilistic Approach to Full-Text Document Clustering". Tech. Report ITAD-433-MS-98-044, SRI International, 1998.

[GOOGLE, 02] Google Search Engine. Available online at URL:
http://www.google.com/press/highlights.html. (last visited July 2002)

[GOV, 02] The .gov test collection, Available online at URL:
http://www.ted.cmis.csiro.au/TRECWeb/govinfo.html (last Visited July 2002)

[Grefensette, 97]. Grefensette, G. "Short Query Linguistic Expansion Techniques: Palliating One-Word Queries by Providing Intermediate Structure to Text", Information Extraction, International Summer School, SCIE 97.

[Grefenstette & Nioche, 00] Grefenstette G. and Nioche J. "Estimation of English and non-English Language Use on the WWW". Proceedings of RIAO'2000.

[Gurrin & Smeaton, 99] Gurrin, C. and Smeaton, A.F. "Connectivity Analysis Approaches to Increasing Precision in Retrieval from Hyperlinked Documents". Proceedings of the 8th Annual TREC Conference", November 16-19, 1999.

[Gurrin & Smeaton, 00] Gurrin, C. and Smeaton, A.F. "Dublin City University Experiments in Connectivity Analysis for TREC-9". Proceedings of the 9th Annual TREC Conference", November 16-19, 2000.

[Harman, 92] Harman D. "Overview of the First Text REtrieval Conference (TREC-1)". Proceedings of the 1st Annual TREC Conference, November 4-6, 1992.

[Harmandas et al., 97] Harmandas, V., Sanderson, M., Dunlop, M. "Image retrieval by hypertext links". Proceedings of the 19th ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'97), 1997.

[Hawking & Craswell, 01] Hawking D. and Craswell N. "Overview of the TREC 2001 Web Track (DRAFT)". Draft Proceedings of the 10th Annual TREC Conference, November 13-15, 2001.

[Hawking et al., 99] Hawking D., Voorhees E., Craswell N. and Bailey P. "Overview of the TREC-8 Web Track". Proceedings of the 8th Annual TREC Conference", November 16-19, 1999.

[Hawking, 01] Hawking D., - "Overview of the TREC-9 Web Track". Proceedings of the 9th Annual TREC Conference", November 16-19, 2000.

[Henzinger, 01] Henzinger M. "Hyperlink Analysis for the Web". IEEE Internet Computing, pg 45-50, January-February 2001.

[HighSearchEngineRanking, 02] Search Engine Optimisation Website. Available online at URL: http://www.high-search-engine-ranking.com/default.htm. (last Visited July 2002).

[HOTDOG] HTML HotDog from Sausage Software, Available online at URL: http://www.sausage.com. (last Visited July 2002).

[Huang, 00] Huang, L. "A Survey on Web Information Retrieval Technologies". RPE report, January 2000.

[INDEX SERVER] Microsoft Index Server Software Application. Available online at URL: http://www.microsoft.com/windows2000/technologies/web/default.asp. (last visited July 2002).

[Ingwersen, 98] Ingwersen P. "The Calculation of Web Impact Factors". Journal of Documentation, 54(2), pg 236-243, 1998.

[INTERNET ARCHIVE, 02] The Internet Archive, Available online at URL: http://www.archive.org. (last Visited July 2002).

[IONAUT, 02] The Ionaut Question Answering System, Available online at URL: http://www.ionaut.com:8400/. (last Visited July 2002).

[Josuttis, 99] Josuttis N. "The C++ Standard Library: A Tutorial and Reference". Addison-Wesley Pub Co., 1st edition, August 1999.

[Kessler, 63] Kessler, M. "Bibliographic Coupling between Scientific Papers". American Documentation, Vol.14, No.1, pg 10--25, 1963.

[Kraaij & Westerveld, 00] Kraaij W. and Westerveld T., "TNO-UT at TREC-9: How Different are Web Documents?". Proceedings of the 9th Annual TREC Conference, November 16-19, 2000.

[Kwok et al., 00] Kwok K., Grunfeld L., Dinstl N. and Chan M., "TREC-9 Cross Language, Web and Question-Answering Track Experiments using PIRCS". Proceedings of the 9th Annual TREC Conference, November 16-19, 2000.

[Li, 98] Li Y. "Toward a more Qualitative Search Engine". IEEE Internet Computing, Vol 2 No. 4, pg 24-29, Jul-Aug 1998.

[Luhn, 58] Luhn, H.P. "The automatic creation of literature abstracts". IBM Journal of Research and Development (159-165), 1958.

[LYCOS, 02] The Lycos Search Engine : Available online at URL: http://www.lycos.com. (last visited July 2002).

[McBryan, 94] McBryan O. "GENVL and WWWW: Tools for taming the Web", Proceedings of the 1st WWW Conference, May, 1994.

[McNamee et al., 00] McNamee P., Mayfield J. and Piatko C., "The HAIRCUT System at TREC-9". Proceedings of the 9th Annual TREC Conference, November 16-19, 2000.

[Mechkour et al., 98] Mechkour M., Harper D.J. and Muresan G. "The WebCluster Project. Using Clustering for Mediating Access to the World Wide Web". Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in IR, 1998.

[Miller, 96] George A. Miller "The Science of Words". W H Freeman & Co Publishers, April 1996.

[Mitzenmacher, 01] Mitzenmacher M. "A Brief History of Generative Models for Power Law and Lognormal Distributions". Allerton 2001.

[Murray & Moore, 00] Murray B. and Moore A., "Sizing the Internet - A White Paper", White paper, Cyveillance, Inc., 2000. Available online at URL:
http://www.cyveillance.com/web/corporate/white_papers.htm. (last Visited July 2002).

[OPENDIRECTORY, 02] The OpenDirectory web Directory : Available online at URL:
http://www.opendirectory.org. (last visited July 2002).

[Page et al., 98] Page L., Brin S., Motwani R. and Winograd T. "The PageRank Citation Ranking: Bringing Order to the Web", Stanford Digital Libraries working paper, 1997-0072.

[PANDIA, 02] Pandia Search World. Available online at URL:
http://www.pandia.com/searchworld. (last visited July 2002).

[Pennock et al., 02] Pennock, D., Flake, G., Lawrence, S., Glover, E. and Giles, C. "Winners don't take all: Characterising the competition for links on the web". Proceedings of the National Academy of Sciences, Volume 99, Issue 8, pg 5207-5211, April 2002.

[Powell, 98] Powell, T. "HTML: The Complete Reference". McGraw-Hill, 1998.

[RANKDEX, 02] The Rankdex Prototype Search Engine. Available online at
UTL:http://rankdex.gari.com. (last visited April 2002).

[Robertson & Sparck Jones, 76] Robertson, S. and Sparck Jonesr, K. "Relevance weighting of search terms". Journal of the American Society for Information Science, volume 27, pg 129-146, 1976.

[Robertson, 77] Robertson, S. "The probability ranking principle in IR". Journal of Documentation, 33(4) pg 294 - 304, 1977.

[ROBOTS, 02] The Robots Exclusion Standard, Available online at URL: http://www.robotstxt.org/wc/eval.html (last visited July 2002).

[Rocchio, 65] Rocchio, J.J. "Relevance Feedback in Information Retrieval", Scientific Report ISR-9, Section 23, August 1965.

[Salton & McGill, 83] Salton, G. and McGill, M.J., "The SMART and SIRE Experimental Retrieval Systems". Readings in Information Retrieval (ed. K S Jones, P Willett), pg 381-399, Morgan Kaufman Publishers, 1997.

[Salton et al., 75] Salton, G., Wong, A. and Yang, C. "A vector space model for automatic indexing". Communications of the ACM 18, pg. 613-620, 1975.

[Savoy & Picard, 99] Savoy, J. and Picard, J. "Report on the TREC-8 Experiment: Searching on the Web and in Distributed Collections" Proceedings of the 8th Annual TREC Conference, pg 229- 239. November 16-19, 1999.

[Savoy & Rasolofo, 00] Savoy, J. and Rasolofo, Y. "Report on the TREC-9 Experiment: Link-based Retrieval an Distributed Collections". Proceedings of the 9th Annual TREC Conference, November 16-19, 2000.

[SEARCHDAY11, 01] Sherman, C. "Search Day, May 21, 2001: The Once and Future P2P". Available online at URL: http://www.searchenginewatch.com/searchday/01/sd0521-p2p.html. (last Visited July 2002).

[SEARCHDAY291, 02] Sherman, C. "Search Day, June 17th 2002". Available online at URL: http://searchenginewatch.com/searchday/02/sd0617-update.html. (last visited July 2002).

[SEARCHDAY73, 01] Sherman, C. "Search Day, August 15, 2001: SearchDay Reader Tips 1". Available online at URL: http://searchenginewatch.com/searchday/01/sd0815-sdtips1.html. (last Visited July 2002).

[SearchEngineSerious, 02] Search Engine Serious WebSite. Available online at URL: http://www.searchengineserious.co.uk. (last Visited July 2002).

[Shin et al., 99] Shin, D., Kim, Y. and Kim, S., "SCAI TREC-8 Experiments". Proceedings of the 8th Annual TREC Conference, pg 583- 590. November 16-19, 1999.

[Silverstein et al., 98] Silverstein, C., Henzinger, M., Marais, H. and Moricz, M. "Analysis of a very large AltaVista Query Log", SRC Technical Note, Oct 26, 1998.

[Singhal & Kaszkiel, 00] Singhal, A. and Kaszkiel, M. "AT&T at TREC-9". Proceedings of the 9th Annual TREC Conference, November 16-19, 2000.

[Singhal et al., 96] Singhal, A., Buckley, C., Mitra, M. "Pivoted document length normalization". Proceedings of the 19th ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'96), 1996.

[Singhal et al., 98] Singhal, A., Choi, J., Hindle, D., Lewis, D. and Pereira F. "AT&T at TREC-7". Proceedings of the 7th Annual TREC Conference, November 19-22, 1998.

[SOWSIII, 99] The third State of the Web Survey, Available online at URL: http://www.pantos.org/atw/35654-a.html. (last visited July 2002).

[Sparck Jones & Webster, 79] Sparck Jones, K. and Webster, C., "Research in Relevance Weighting". British Library Research and Development Report 5553, Computer Laboratory, University of Cambridge., 1979.

[Spertus, 97] Spertus E. "Parasite: Mining Structural Information on the Web" Proceedings of the 6th International WWW Conference, 1997.

[SQL SERVER] Microsoft SQL Server, Available online at URL: http://www.microsoft.com/sql/default.asp. (last visited July 2002).

[Sullivan, 02] Sullivan, D. "Search Engine Watch NewsLetter". Available online at URL: http://searchenginewatch.com/sereport/00/11-liv.html. (last visited July 2002).

[NETSIZER, 02] The Telcordia NetSizer WebSite. Available online at URL: http://www.netsizer.com. (last visited July 2002).

[TEOMA, 02] The Teoma Search Engine. Available online at URL: http://www.teoma.com. (last visited July 2002).

[TREC, 02] The TREC Text REtrieval Conference WebSite. Available online at URL: http://trec.nist.gov. (last visited July 2002).

[UROULETTE, 02] URouLette Random Web Page Generator, Available online at URL: http://www.uroulette.com. (last visited July 2002).

[van Rijsbergen, 79] van Rijsbergen, K. "Information Retrieval". 2nd edition, Butterworths, London, 1979. Also available online at URL: http://dcs.glasgow.ac.uk/Keith/Preface.html.

[Voorhees & Harman, 00] Voorhees, E. and Harman, D. "Overview of the Ninth Text REtrieval Conference (TREC-9)". Proceedings of the 9th Annual TREC Conference, November 13-15, 2001.

[Voorhees & Harman, 01] Voorhees, E. and Harman, D. "Overview of TREC-2001". Draft Proceedings of the 10th Annual TREC Conference, November 13-15, 2001.

[Voorhees & Harman, 99] Voorhees, E. and Harman, D. "Overview of the Eighth Text Retrieval Conference (TREC-8)". Proceedings of the 8th Annual TREC Conference", November 16-19, 1999.

[Voorhees, 01] Voorhees, E. "Overview of TREC-2001", Draft Proceedings of the 10th Annual TREC Conference, November 13-15, 2001.

[Voorhees_2, 01] Voorhees E., Presentation: "Overview of TREC-2001" Draft Proceedings of the 10th Annual TREC Conference", Gaithersburg, MD, November 13-15, 2001.

[W3C, 02] The World Wide Web Consortium. Available online at URL: http://www.w3.org. (last visited July 2002).

[WEBCRAWLER, 02] The Webcrawler Search Engine. Available online at URL: http://www.webcrawler.com. (last visited July 2002).

[WebPromotion, 02] Web Promotion Inc. Available online at URL: http://www.webpage-promotions.com/index.html. (last Visited July 2002).

[Westerveld et al., 01] Westerveld, T., Kraaij, W., and Hiemstra, D. "Retrieving Web Pages using Content, Links, URLs and Anchors", Draft Proceedings of the 10th Annual TREC Conference, November 13-15, 2001.

[Wired, 02] "Deep Links Return to Surface". April 18, 2002, Available online at URL: http://www.wired.com/news/politics/0,1283,51887,00.html. (last visited July 2002).

[Walker et al., 97] Walker, S., Robertson, S., Boughanem, M., Jones, G., and Sparck Jones, K. "Okapi at TREC-6 Automatic ad hoc, VLC, routing, filtering and QSDR". Proceedings of the 6th Annual TREC Conference, November 19-21, 1997.

[YAHOO, 02] The Yahoo Search Engine/Directory. Available online at URL: http://www.opendirectory.org. (last visited July 2002).

[Zamir & Etzioni, 98] Zamir, O. and Etzioni, O. "Web Document Clustering : A Feasibility Demonstration". 1994

[Zipf, 32] Zipf, G. "Selective Studies and the Principle of Relative Frequency in Language", Harvard University Press, Cambridge, MA, 1932.

[Zobel et al, 98] Zobel, J., Moffat, A. and Ramamohanarao, K. "Inverted files Versus Signature files for Text Indexing". ACM Transactions on Database Systems, Vol. 23, No.4, December 1998, pg 453-490.

[Zobel, 98] Zobel, J. "How reliable are the results of large-scale information retrieval experiments?". Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in IR, 1998.

Appendix A

Identifying Irish Documents

In order to identify the language of a document we used a combination of most frequent short tokens per language and most frequent trigrams per language. We calculated the probabilities of a term being found next in a document using the relative frequencies of the occurrences of all terms over a very large number of observations. The top Irish Terms and their probabilities are shown below.

ar	agus	na	ag	tá	sa	ó	sé	seo	bhí
.02832	.02603	.02531	.01333	.00875	.00693	.00666	.00661	.00640	.00596

Table A.1 The top 10 Irish Terms and their probabilities

ach	ar_	na_	_ag	ch_	_na	_bh	Gus	agu	nn_
.00932	.00885	.00757	.00746	.00523	.00519	.00517	.00461	.00460	.00455

Table A.2 : The top 10 Irish Trigrams and their probabilities

The probabilities of a term occurring are calculated according to the following formula, letting $tf(T_n)$ be the term frequency of T_n in the corpus and N is the total number of words in the corpus:

$$\Pr\{T_n\} = \frac{tf(T_n)}{N}$$

It has been found by experimentation within our university [Clarke & Kelly, 98] that as little as 150KB of text is required as a corpus to select the most frequent terms and trigrams. For this

experiment, we gathered 1.5MB of Irish language text (1 MB was chosen from newspapers and 0.5 MB was chosen from websites[1].

Once we had the probabilities of all terms in a corpus calculated, ten of the top terms were selected for use in the language disambiguation task. These were not simply the top ten terms as there would have been some crossover between top tanked Irish terms and top ranked English terms. Therefore the top 10 acceptably unique terms & trigrams were chosen. Table A.3 illustrates three terms with high probabilities in both Irish and English.

TERM	IRISH PROBABILITY	ENGLISH PROBABILITY
i	0.0170186428	0.0052848242
as	0.0028423538	0.0070093457
a	0.0400058345	0.0193035157

Table A.3 : Illustrating the cross-over between commonly occurring Irish and English terms

Through the use of both term tokenisers and trigram tokenisers any incoming document was decomposed into vectors of both terms and trigrams. Based on these vectors each document was given a probability of being Irish based on the frequency of occurrences of the top 10 terms and trigrams in both Irish and English. For example the Irish Trigram score of a document would be the sum of the probabilities of each occurrence of each of the top 10 Irish Terms divided by the total number of terms. In this way we get a probability score for each document being Irish. If the document passes a threshold it is accepted as Irish, however if the document is does not pass a threshold then it is the subject of additional examination. This is because many Irish Language web pages will not be entirely Irish, even if a fraction of a document was Irish we wanted to be able to add it to the dataset. Therefore, if a document was not Irish, but the probability value for the document surpassed a lower bound, then we calculated the probability of the document being English using a similar manner to the calculation of Irish probabilities. If a document had a high English probability yet had a sufficiently high Irish term score the document was assumed Irish.

In addition to the problem of bi-lingual documents, there are other languages on the Web other than Irish and English that may influence the language detection process. Since, by the very nature of the web being linked and since crawlers traverse the web by following links, we had to allow

[1] The text was chosen from a number of different web sites: Entertainment, News, Sport, tv listings...

for the fact that a document could be added to the queue which was neither Irish nor English. To combat this problem, we selected eight common European languages and using the five most frequently occurring terms [Grefenstette, 97] checked each page for its similarity to these languages, in case the crawler was straying from Irish and English pages. Periodic examination of the URL queue allows us to remove any other problem URLs that we found. In this way we gathered 26,798 Irish language documents. Adherence to the Robots Exclusion Standard disallowed us from gathering additional documents that our crawler located.

Appendix B

SiteRank Top 100 Ranked URLs

These URLs are shown in decreasing order of SiteRank score.

RANK	URL	SCORE
0	http://www.netscape.com/	20.075829
1	http://www.microsoft.com/	17.856512
2	http://www.yahoo.com/	16.047762
3	http://home.netscape.com/	14.709599
4	http://www.microsoft.com/ie/	13.441017
5	http://www.adobe.com/	10.052464
6	http://home.netscape.com/comprod/mirror/index.html	8.126648
7	http://www.adobe.com/products/acrobat/readstep.html/	7.921433
8	http://www.excite.com/	7.804879
9	http://www.lycos.com/	6.879511
10	http://www.real.com/	6.708895
11	http://www.digits.com/	6.688189
12	http://worldwidemart.com/scripts/	6.140473
13	http://www.freeservers.com/	5.627914
14	http://www.stpt.com/	5.052332
15	http://www.altavista.com/	4.689848
16	http://home.netscape.com/download/	4.323414
17	http://home.netscape.com/comprod/mirror/client_download.html	4.308418
18	http://www.mapquest.com/	4.292669
19	http://www.apache.org/	4.046741
20	http://www.winzip.com/	3.972477
21	http://www.macromedia.com/shockwave/download/	3.966242
22	http://www.webcrawler.com/	3.945361
23	http://www.apple.com/	3.698875
24	http://www.cnn.com/	3.629958
25	http://www.addme.com/	3.437808
26	http://www.microsoft.com/windows/ie/default.htm	3.41965
27	http://home.netscape.com/computing/download/index.html	3.194697
28	http://www.webring.org/	3.13005
29	http://www.netscape.com/download/	2.984217
30	http://www.netnanny.com/	2.977879
31	http://home.about.com/	2.971298
32	http://www.realaudio.com/	2.962381
33	http://home.netscape.com/download/index.html/	2.767508
34	http://www.cyberpatrol.com/	2.760402
35	http://www.sun.com/	2.715243
36	http://www.linux.org/	2.681929
37	http://www.freefind.com/	2.631474
38	http://www.egroups.com/	2.601581
39	http://www.macromedia.com/	2.577999
40	http://www.microsoft.com/ie/logo.asp	2.546094
41	http://www.microsoft.com/frontpage/	2.510393
42	http://www.nasa.gov/	2.483305
43	http://www.apple.com/quicktime/	2.455769
44	http://www.redhat.com/	2.423023
45	http://www.microsoft.com/windows/ie/	2.387594
46	http://www.zdnet.com/downloads/altavista/	2.38757
47	http://www.microsoft.com/ie/ie.htm	2.329393
48	http://www.webjump.com/	2.309386
49	http://www.nsf.gov/	2.296819
50	http://www.worldwidemart.com/scripts/	2.267708
51	http://www.ibm.com/	2.261238
52	http://www.ncsa.uiuc.edu/General/Internet/WWW/HTMLPrimer.html	2.245173
53	http://www.tucows.com/	2.237581
54	http://www.usatoday.com/	2.15026

55	http://www.eads.com/	2.146364
56	http://www.weather.com/	2.125444
57	http://www.netscape.com/comprod/mirror/client_download.html	2.094992
58	http://www.nytimes.com/	2.090128
59	http://123counter.mycomputer.com/	2.073024
60	http://www.whitehouse.gov/	2.058892
61	http://www.hp.com/	2.055855
62	http://www.eff.org/blueribbon.html	2.038758
63	http://www.hwg.org/	2.0364
64	http://www.census.gov/	2.010182
65	http://counter.mycomputer.com/	2.008063
66	http://java.sun.com/	1.998716
67	http://www.google.com/	1.983114
68	http://www.w3.org/	1.91467
69	http://www.microsoft.com/windows/ie/download/windows.htm	1.900927
70	http://www.safesurf.com/	1.821805
71	http://www.netmind.com/	1.730375
72	http://www.yahooligans.com/	1.704588
73	http://www.netscape.com/download/index.html/	1.69374
74	http://www.nih.gov/	1.685862
75	http://www.dogpile.com/	1.682923
76	http://www.intel.com/	1.647742
77	http://www.winamp.com/	1.586767
78	http://www.ed.gov/	1.571733
79	http://www.icq.com/	1.559529
80	http://www.flycast.com/about_us/about-privacy.html	1.546888
81	http://www.guestworld.com/	1.535662
82	http://www.aol.com/	1.512522
83	http://www.northernlight.com/	1.511045
84	http://www.virtualave.net/	1.510922
85	http://www.msn.com/	1.496504
86	http://www.unitedmedia.com/comics/dilbert/	1.484103
87	http://www.onelist.com/	1.454122
88	http://www.eff.org/	1.434908
89	http://www.cdc.gov/	1.43227
90	http://www.linkstoyou.com/	1.42599
91	http://www.excite.com/navigate/	1.422774
92	http://www.netscape.com/comprod/mirror/index.html	1.417294
93	http://www.switchboard.com/	1.405176
94	http://www.ebay.com/	1.38906
95	http://www.house.gov/	1.388995
96	http://www.sgi.com/	1.375629
97	http://www.nlm.nih.gov/	1.363653
98	http://www.networksolutions.com/	1.359643
99	http://www.persiankittv.com/	1.313381

Appendix C

Results of Experiments on WT_Connected

We now present the scores obtained by all ten algorithms when executed against WT_Connected using the distilled relevance judgements. These are the total scores, averaged over all queries. All 50 queries used in our experiments are included in Table A.4 below.

OUR ID	TREC ID	QUERY
1	451	bengal cat bengals
2	452	beaver beavers habitat
3	453	hunger hunger hunger organization eradication eradicate group
4	454	parkinson parkinson's disease
5	455	jackie robinson jackie robinson first game
6	456	end world 2000 apocalypse
7	457	chevrolet chevy truck
8	458	fasting fasting religion religious
9	459	lender forclose property lender forclose property legal legally
10	460	moses moses moses israel
11	461	lava lava lava lamp lamps
12	462	realtor realtor new jersey new jersey residential real estate
13	463	tartan tartan tartan Scottish Scotland Scot
14	464	nativity scene ban states cities state city nativity ban nativity ban nativity ban
15	465	deer disease human Lyme
16	466	peer gynt Grieg
17	467	dachshund dachshund dachshund wiener dog dog
18	468	incandescent incandescent incandescent light bulb light bulb history
19	469	steinbach steinbach steinbach nutcracker nutcrackers
20	470	mistletoe beneficial benefit
21	471	mexican food mexican food mexican food popular
22	472	antique appliance restoration&20antique appliance restoration&20antique appliance restoration&20museum dealer
23	473	toronto film festival toronto film toronto film movie
24	474	e mail e mail e mail business benefit internet
25	475	zirconium zirconium properties
26	476	jennifer aniston jennifer aniston movies tv television
27	477	royal caribbean cruise royal caribbean cruise royal caribbean cruise line ships
28	478	baltimore mayor baltimore mayor baltimore mayor name
29	479	kappa alpha psi kappa alpha psi kappa alpha psi information
30	480	car traffic car traffic car traffic report washington maryland virginia
31	481	babe ruth babe ruth babe ruth baseball
32	482	pine tree pine tree growth rate
33	483	rosebowl parade rosebowl parade rosebowl parade rose bowl
34	484	skoda skoda skoda automobile car
35	485	gps gps gps clock accuracy clock

36	486	el dorado casino eldorado reno
37	487	angioplasty angioplasty angioplasty follow repeat
38	488	newport beach newport beach newport beach entertainment
39	489	calcium calcium medical benefit benefits supplements supplement
40	490	motorcycle helmet motorcycle helmet law safety
41	491	tsunami tsunami20japanese wave
42	492	savings savings savings bonds saving bond
43	493	retirement community retirement community retirement community us canada
44	494	nirvana nirvana members
45	495	roaring twenties 20s
46	496	tmj tmj tmj temporal mandible joint
47	497	orchid orchid orchids grow growing industry
48	498	hair transplant hair transplant hair transplant procedure
49	499	pool cue pool cue pool cue use development select origin
50	500	dna dna testing

Table A.4 : Weighted Queries used in our experiments

```
Queryid (Num):       Content-only
Total number of documents over all queries
    Retrieved:    34196
    Relevant:       254
    Rel_ret:        218
Interpolated Recall - Precision Averages:
    at 0.00       0.5050
    at 0.10       0.4770
    at 0.20       0.4171
    at 0.30       0.3677
    at 0.40       0.3548
    at 0.50       0.3244
    at 0.60       0.2654
    at 0.70       0.1908
    at 0.80       0.1767
    at 0.90       0.1384
    at 1.00       0.1316
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.2854
Precision:
  At    5 docs:   0.2389
  At   10 docs:   0.1833
  At   15 docs:   0.1611
  At   20 docs:   0.1500
  At   30 docs:   0.1167
  At  100 docs:   0.0444
  At  200 docs:   0.0246
  At  500 docs:   0.0114
  At 1000 docs:   0.0061
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.2777    Exact:        0.2777
```

```
Queryid (Num):       Link 1
Total number of documents over all queries
    Retrieved:    34196
    Relevant:       254
    Rel_ret:        219
Interpolated Recall - Precision Averages:
    at 0.00       0.0270
    at 0.10       0.0258
    at 0.20       0.0209
    at 0.30       0.0208
    at 0.40       0.0166
    at 0.50       0.0163
    at 0.60       0.0152
    at 0.70       0.0144
    at 0.80       0.0141
    at 0.90       0.0129
    at 1.00       0.0107
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0135
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.0056
  At   15 docs:   0.0074
  At   20 docs:   0.0083
  At   30 docs:   0.0093
  At  100 docs:   0.0072
  At  200 docs:   0.0061
  At  500 docs:   0.0085
  At 1000 docs:   0.0061
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0063
```

```
Queryid (Num):       Link 2
Total number of documents over all queries
    Retrieved:    34196
    Relevant:       254
    Rel_ret:        218
Interpolated Recall - Precision Averages:
    at 0.00       0.2362
    at 0.10       0.2325
    at 0.20       0.1966
    at 0.30       0.1550
    at 0.40       0.1430
    at 0.50       0.1388
    at 0.60       0.1005
    at 0.70       0.0541
    at 0.80       0.0520
    at 0.90       0.0370
    at 1.00       0.0327
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.1132
Precision:
  At    5 docs:   0.0944
  At   10 docs:   0.0694
  At   15 docs:   0.0537
  At   20 docs:   0.0528
  At   30 docs:   0.0537
  At  100 docs:   0.0331
  At  200 docs:   0.0222
  At  500 docs:   0.0109
  At 1000 docs:   0.0061
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0912
```

```
Queryid (Num):       Link 3
Total number of documents over all queries
    Retrieved:    34196
    Relevant:       254
    Rel_ret:        218
Interpolated Recall - Precision Averages:
    at 0.00       0.5042
    at 0.10       0.4734
    at 0.20       0.4164
    at 0.30       0.3655
    at 0.40       0.3528
    at 0.50       0.3236
    at 0.60       0.2545
    at 0.70       0.1747
    at 0.80       0.1645
    at 0.90       0.1277
    at 1.00       0.1210
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.2795
Precision:
  At    5 docs:   0.2444
  At   10 docs:   0.1833
  At   15 docs:   0.1630
  At   20 docs:   0.1472
  At   30 docs:   0.1148
  At  100 docs:   0.0444
  At  200 docs:   0.0247
  At  500 docs:   0.0114
  At 1000 docs:   0.0061
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.2610
```

```
Queryid (Num):       Link 4
Total number of documents over all queries
    Retrieved:    34196
    Relevant:       254
    Rel_ret:        218
Interpolated Recall - Precision Averages:
    at 0.00       0.5057
    at 0.10       0.4776
    at 0.20       0.4178
    at 0.30       0.3648
    at 0.40       0.3520
    at 0.50       0.3228
    at 0.60       0.2553
    at 0.70       0.1746
    at 0.80       0.1647
    at 0.90       0.1280
    at 1.00       0.1214
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.2799
Precision:
  At    5 docs:   0.2500
  At   10 docs:   0.1861
  At   15 docs:   0.1630
  At   20 docs:   0.1486
  At   30 docs:   0.1148
  At  100 docs:   0.0442
  At  200 docs:   0.0249
  At  500 docs:   0.0114
  At 1000 docs:   0.0061
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.2603
```

```
Queryid (Num):       Link 5
Total number of documents over all queries
    Retrieved:    34196
    Relevant:       254
    Rel_ret:        218
Interpolated Recall - Precision Averages:
    at 0.00       0.1857
    at 0.10       0.1672
    at 0.20       0.1470
    at 0.30       0.1219
    at 0.40       0.1178
    at 0.50       0.1106
    at 0.60       0.0973
    at 0.70       0.0707
    at 0.80       0.0644
    at 0.90       0.0452
    at 1.00       0.0377
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0903
Precision:
  At    5 docs:   0.0500
  At   10 docs:   0.0639
  At   15 docs:   0.0722
  At   20 docs:   0.0750
  At   30 docs:   0.0787
  At  100 docs:   0.0406
  At  200 docs:   0.0244
  At  500 docs:   0.0114
  At 1000 docs:   0.0061
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0492
```

```
Queryid (Num):       PageRank Parameter values
Total number of documents over all queries
    Retrieved:    34196
    Relevant:       254
    Rel_ret:        218
Interpolated Recall - Precision Averages:
    at 0.00       0.5037
    at 0.10       0.4755
    at 0.20       0.4161
    at 0.30       0.3620
    at 0.40       0.3526
    at 0.50       0.3226
    at 0.60       0.2546
    at 0.70       0.1773
    at 0.80       0.1636
    at 0.90       0.1272
    at 1.00       0.1205
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.2788
Precision:
  At    5 docs:   0.2444
  At   10 docs:   0.1833
  At   15 docs:   0.1630
  At   20 docs:   0.1486
  At   30 docs:   0.1130
  At  100 docs:   0.0444
  At  200 docs:   0.0246
  At  500 docs:   0.0114
  At 1000 docs:   0.0061
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.2604
```

```
Queryid (Num):       PageRank S-A
Total number of documents over all queries
    Retrieved:    34196
    Relevant:       254
    Rel_ret:        218
Interpolated Recall - Precision Averages:
    at 0.00       0.5036
    at 0.10       0.4759
    at 0.20       0.4161
    at 0.30       0.3634
    at 0.40       0.3538
    at 0.50       0.3238
    at 0.60       0.2558
    at 0.70       0.1768
    at 0.80       0.1629
    at 0.90       0.1272
    at 1.00       0.1206
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.2793
Precision:
  At    5 docs:   0.2444
  At   10 docs:   0.1806
  At   15 docs:   0.1593
  At   20 docs:   0.1486
  At   30 docs:   0.1139
  At  100 docs:   0.0444
  At  200 docs:   0.0246
  At  500 docs:   0.0114
  At 1000 docs:   0.0061
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.2615
```

```
Queryid (Num):       SiteRank Parameter Values
Total number of documents over all queries
    Retrieved:    34196
    Relevant:       254
    Rel_ret:        219
Interpolated Recall - Precision Averages:
    at 0.00       0.4994
    at 0.10       0.4597
    at 0.20       0.4218
    at 0.30       0.3599
    at 0.40       0.3467
    at 0.50       0.3224
    at 0.60       0.2665
    at 0.70       0.1838
    at 0.80       0.1699
    at 0.90       0.1347
    at 1.00       0.1290
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.2820
Precision:
  At    5 docs:   0.2556
  At   10 docs:   0.1778
  At   15 docs:   0.1537
  At   20 docs:   0.1431
  At   30 docs:   0.1120
  At  100 docs:   0.0447
  At  200 docs:   0.0250
  At  500 docs:   0.0113
  At 1000 docs:   0.0061
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.2574
```

```
Queryid (Num):       SiteRank S-A
Total number of documents over all queries
    Retrieved:    34196
    Relevant:       254
    Rel_ret:        219
Interpolated Recall - Precision Averages:
    at 0.00       0.5050
    at 0.10       0.4635
    at 0.20       0.4274
    at 0.30       0.3676
    at 0.40       0.3501
    at 0.50       0.3269
    at 0.60       0.2669
    at 0.70       0.1860
    at 0.80       0.1699
    at 0.90       0.1342
    at 1.00       0.1285
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.2853
Precision:
  At    5 docs:   0.2556
  At   10 docs:   0.1806
  At   15 docs:   0.1574
  At   20 docs:   0.1417
  At   30 docs:   0.1139
  At  100 docs:   0.0450
  At  200 docs:   0.0249
  At  500 docs:   0.0113
  At 1000 docs:   0.0061
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.2690
```

We now present the scores obtained by a selection of the linkage algorithms and the content-only experiment when executed against WT_Connected using the distilled relevance judgements. Note that we have included all fifty TREC queries even though fourteen of them do not contain any relevant documents in the WT_Connected test collection.

The experiments we include here are:

- Content-only Experiment.
- Link 3 : Normalised indegree weighting, best guess combination.
- Link 4 : Normalised indegree weighting, scarcity-abundance combination.
- SiteRank using scarcity-abundance combination.

Content-only Experiment
Using the BM25 Ranking Algorithm

```
Queryid (Num):        1
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         7
    Rel_ret:          4
Interpolated Recall - Precision Averages:
    at 0.00       0.5000
    at 0.10       0.5000
    at 0.20       0.5000
    at 0.30       0.1875
    at 0.40       0.1875
    at 0.50       0.0059
    at 0.60       0.0000
    at 0.70       0.0000
    at 0.80       0.0000
    at 0.90       0.0000
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.1705
Precision:
  At    5 docs:   0.4000
  At   10 docs:   0.2000
  At   15 docs:   0.1333
  At   20 docs:   0.1500
  At   30 docs:   0.1000
  At  100 docs:   0.0300
  At  200 docs:   0.0150
  At  500 docs:   0.0060
  At 1000 docs:   0.0040
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.2857
```

```
Queryid (Num):        2
Total number of documents over all queries
    Retrieved:     1000
    Relevant:        16
    Rel_ret:         16
Interpolated Recall - Precision Averages:
    at 0.00       0.1818
    at 0.10       0.1818
    at 0.20       0.1818
    at 0.30       0.1316
    at 0.40       0.0347
    at 0.50       0.0280
    at 0.60       0.0260
    at 0.70       0.0168
    at 0.80       0.0168
    at 0.90       0.0168
    at 1.00       0.0168
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0643
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.0000
  At   15 docs:   0.1333
  At   20 docs:   0.1000
  At   30 docs:   0.1333
  At  100 docs:   0.0600
  At  200 docs:   0.0300
  At  500 docs:   0.0200
  At 1000 docs:   0.0160
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.1250
```

```
Queryid (Num):        3
Total number of documents over all queries
    Retrieved:     1000
    Relevant:        16
    Rel_ret:         15
Interpolated Recall - Precision Averages:
    at 0.00       1.0000
    at 0.10       1.0000
    at 0.20       1.0000
    at 0.30       0.8333
    at 0.40       0.6429
    at 0.50       0.6429
    at 0.60       0.5714
    at 0.70       0.5714
    at 0.80       0.2414
    at 0.90       0.1807
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.6001
Precision:
  At    5 docs:   0.8000
  At   10 docs:   0.6000
  At   15 docs:   0.6000
  At   20 docs:   0.5500
  At   30 docs:   0.4000
  At  100 docs:   0.1500
  At  200 docs:   0.0750
  At  500 docs:   0.0300
  At 1000 docs:   0.0150
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.5625
```

```
Queryid (Num):        4
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         8
    Rel_ret:          8
Interpolated Recall - Precision Averages:
    at 0.00       1.0000
    at 0.10       1.0000
    at 0.20       0.5000
    at 0.30       0.5000
    at 0.40       0.5000
    at 0.50       0.5000
    at 0.60       0.2273
    at 0.70       0.1463
    at 0.80       0.1321
    at 0.90       0.1270
    at 1.00       0.1270
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.3618
Precision:
  At    5 docs:   0.2000
  At   10 docs:   0.4000
  At   15 docs:   0.2667
  At   20 docs:   0.2000
  At   30 docs:   0.1667
  At  100 docs:   0.0800
  At  200 docs:   0.0400
  At  500 docs:   0.0160
  At 1000 docs:   0.0080
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.5000
```

```
Queryid (Num):          5
Total number of documents over all queries
    Retrieved:          0
    Relevant:           0
    Rel_ret:            0
Interpolated Recall - Precision Averages:
    at 0.00         0.0000
    at 0.10         0.0000
    at 0.20         0.0000
    at 0.30         0.0000
    at 0.40         0.0000
    at 0.50         0.0000
    at 0.60         0.0000
    at 0.70         0.0000
    at 0.80         0.0000
    at 0.90         0.0000
    at 1.00         0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                    0.0000
Precision:
  At    5 docs:     0.0000
  At   10 docs:     0.0000
  At   15 docs:     0.0000
  At   20 docs:     0.0000
  At   30 docs:     0.0000
  At  100 docs:     0.0000
  At  200 docs:     0.0000
  At  500 docs:     0.0000
  At 1000 docs:     0.0000
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:          0.0000
```

```
Queryid (Num):        6
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         2
    Rel_ret:          2
Interpolated Recall - Precision Averages:
    at 0.00       0.0588
    at 0.10       0.0588
    at 0.20       0.0588
    at 0.30       0.0588
    at 0.40       0.0588
    at 0.50       0.0588
    at 0.60       0.0095
    at 0.70       0.0095
    at 0.80       0.0095
    at 0.90       0.0095
    at 1.00       0.0095
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0342
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.0000
  At   15 docs:   0.0000
  At   20 docs:   0.0500
  At   30 docs:   0.0333
  At  100 docs:   0.0100
  At  200 docs:   0.0050
  At  500 docs:   0.0040
  At 1000 docs:   0.0020
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):          7
Total number of documents over all queries
    Retrieved:      1000
    Relevant:          6
    Rel_ret:           6
Interpolated Recall - Precision Averages:
    at 0.00        0.1000
    at 0.10        0.1000
    at 0.20        0.0645
    at 0.30        0.0645
    at 0.40        0.0645
    at 0.50        0.0645
    at 0.60        0.0645
    at 0.70        0.0450
    at 0.80        0.0450
    at 0.90        0.0297
    at 1.00        0.0297
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                   0.0541
Precision:
  At    5 docs:    0.0000
  At   10 docs:    0.1000
  At   15 docs:    0.0667
  At   20 docs:    0.0500
  At   30 docs:    0.0333
  At  100 docs:    0.0400
  At  200 docs:    0.0250
  At  500 docs:    0.0120
  At 1000 docs:    0.0060
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:         0.0000
```

```
Queryid (Num):          8
Total number of documents over all queries
    Retrieved:          0
    Relevant:           0
    Rel_ret:            0
Interpolated Recall - Precision Averages:
    at 0.00         0.0000
    at 0.10         0.0000
    at 0.20         0.0000
    at 0.30         0.0000
    at 0.40         0.0000
    at 0.50         0.0000
    at 0.60         0.0000
    at 0.70         0.0000
    at 0.80         0.0000
    at 0.90         0.0000
    at 1.00         0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                    0.0000
Precision:
  At    5 docs:     0.0000
  At   10 docs:     0.0000
  At   15 docs:     0.0000
  At   20 docs:     0.0000
  At   30 docs:     0.0000
  At  100 docs:     0.0000
  At  200 docs:     0.0000
  At  500 docs:     0.0000
  At 1000 docs:     0.0000
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:          0.0000
```

```
Queryid (Num):        9
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         1
    Rel_ret:          0
Interpolated Recall - Precision Averages:
    at 0.00       0.0000
    at 0.10       0.0000
    at 0.20       0.0000
    at 0.30       0.0000
    at 0.40       0.0000
    at 0.50       0.0000
    at 0.60       0.0000
    at 0.70       0.0000
    at 0.80       0.0000
    at 0.90       0.0000
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0000
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.0000
  At   15 docs:   0.0000
  At   20 docs:   0.0000
  At   30 docs:   0.0000
  At  100 docs:   0.0000
  At  200 docs:   0.0000
  At  500 docs:   0.0000
  At 1000 docs:   0.0000
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):          10
Total number of documents over all queries
    Retrieved:          0
    Relevant:           0
    Rel_ret:            0
Interpolated Recall - Precision Averages:
    at 0.00         0.0000
    at 0.10         0.0000
    at 0.20         0.0000
    at 0.30         0.0000
    at 0.40         0.0000
    at 0.50         0.0000
    at 0.60         0.0000
    at 0.70         0.0000
    at 0.80         0.0000
    at 0.90         0.0000
    at 1.00         0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                    0.0000
Precision:
  At    5 docs:     0.0000
  At   10 docs:     0.0000
  At   15 docs:     0.0000
  At   20 docs:     0.0000
  At   30 docs:     0.0000
  At  100 docs:     0.0000
  At  200 docs:     0.0000
  At  500 docs:     0.0000
  At 1000 docs:     0.0000
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:          0.0000
```

```
Queryid (Num):        11
Total number of documents over all queries
    Retrieved:      816
    Relevant:         1
    Rel_ret:          1
Interpolated Recall - Precision Averages:
    at 0.00       0.0238
    at 0.10       0.0238
    at 0.20       0.0238
    at 0.30       0.0238
    at 0.40       0.0238
    at 0.50       0.0238
    at 0.60       0.0238
    at 0.70       0.0238
    at 0.80       0.0238
    at 0.90       0.0238
    at 1.00       0.0238
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0238
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.0000
  At   15 docs:   0.0000
  At   20 docs:   0.0000
  At   30 docs:   0.0000
  At  100 docs:   0.0100
  At  200 docs:   0.0050
  At  500 docs:   0.0020
  At 1000 docs:   0.0010
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):        12
Total number of documents over all queries
    Retrieved:     1000
    Relevant:        11
    Rel_ret:         11
Interpolated Recall - Precision Averages:
    at 0.00       0.2500
    at 0.10       0.2500
    at 0.20       0.2500
    at 0.30       0.2500
    at 0.40       0.1667
    at 0.50       0.1429
    at 0.60       0.0762
    at 0.70       0.0762
    at 0.80       0.0476
    at 0.90       0.0388
    at 1.00       0.0381
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.1384
Precision:
  At    5 docs:   0.2000
  At   10 docs:   0.2000
  At   15 docs:   0.2000
  At   20 docs:   0.2000
  At   30 docs:   0.1667
  At  100 docs:   0.0700
  At  200 docs:   0.0450
  At  500 docs:   0.0220
  At 1000 docs:   0.0110
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.1818
```

```
Queryid (Num):        13
Total number of documents over all queries
    Retrieved:     1000
    Relevant:        16
    Rel_ret:         15
Interpolated Recall - Precision Averages:
    at 0.00       1.0000
    at 0.10       0.6667
    at 0.20       0.3684
    at 0.30       0.3684
    at 0.40       0.3684
    at 0.50       0.3214
    at 0.60       0.3030
    at 0.70       0.1165
    at 0.80       0.0867
    at 0.90       0.0413
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.2953
Precision:
  At    5 docs:   0.4000
  At   10 docs:   0.2000
  At   15 docs:   0.2667
  At   20 docs:   0.3500
  At   30 docs:   0.3000
  At  100 docs:   0.1100
  At  200 docs:   0.0650
  At  500 docs:   0.0300
  At 1000 docs:   0.0150
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.3125
```

```
Queryid (Num):        14
Total number of documents over all queries
    Retrieved:        0
    Relevant:         0
    Rel_ret:          0
Interpolated Recall - Precision Averages:
    at 0.00       0.0000
    at 0.10       0.0000
    at 0.20       0.0000
    at 0.30       0.0000
    at 0.40       0.0000
    at 0.50       0.0000
    at 0.60       0.0000
    at 0.70       0.0000
    at 0.80       0.0000
    at 0.90       0.0000
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0000
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.0000
  At   15 docs:   0.0000
  At   20 docs:   0.0000
  At   30 docs:   0.0000
  At  100 docs:   0.0000
  At  200 docs:   0.0000
  At  500 docs:   0.0000
  At 1000 docs:   0.0000
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):        15
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         2
    Rel_ret:          2
Interpolated Recall - Precision Averages:
    at 0.00       0.0588
    at 0.10       0.0588
    at 0.20       0.0588
    at 0.30       0.0588
    at 0.40       0.0588
    at 0.50       0.0588
    at 0.60       0.0588
    at 0.70       0.0588
    at 0.80       0.0588
    at 0.90       0.0588
    at 1.00       0.0588
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0502
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.0000
  At   15 docs:   0.0000
  At   20 docs:   0.0000
  At   30 docs:   0.0333
  At  100 docs:   0.0200
  At  200 docs:   0.0100
  At  500 docs:   0.0040
  At 1000 docs:   0.0020
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):        16
Total number of documents over all queries
    Retrieved:        0
    Relevant:         0
    Rel_ret:          0
Interpolated Recall - Precision Averages:
    at 0.00       0.0000
    at 0.10       0.0000
    at 0.20       0.0000
    at 0.30       0.0000
    at 0.40       0.0000
    at 0.50       0.0000
    at 0.60       0.0000
    at 0.70       0.0000
    at 0.80       0.0000
    at 0.90       0.0000
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0000
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.0000
  At   15 docs:   0.0000
  At   20 docs:   0.0000
  At   30 docs:   0.0000
  At  100 docs:   0.0000
  At  200 docs:   0.0000
  At  500 docs:   0.0000
  At 1000 docs:   0.0000
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):        17
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         8
    Rel_ret:          8
Interpolated Recall - Precision Averages:
    at 0.00       0.3125
    at 0.10       0.3125
    at 0.20       0.3125
    at 0.30       0.3125
    at 0.40       0.3125
    at 0.50       0.3125
    at 0.60       0.3125
    at 0.70       0.2692
    at 0.80       0.2692
    at 0.90       0.1176
    at 1.00       0.1176
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.2287
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.2000
  At   15 docs:   0.2667
  At   20 docs:   0.2500
  At   30 docs:   0.2333
  At  100 docs:   0.0800
  At  200 docs:   0.0400
  At  500 docs:   0.0160
  At 1000 docs:   0.0080
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.2500
```

```
Queryid (Num):        18
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         5
    Rel_ret:          5
Interpolated Recall - Precision Averages:
    at 0.00       0.5000
    at 0.10       0.5000
    at 0.20       0.5000
    at 0.30       0.5000
    at 0.40       0.5000
    at 0.50       0.2000
    at 0.60       0.2000
    at 0.70       0.2000
    at 0.80       0.2000
    at 0.90       0.0450
    at 1.00       0.0450
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.2890
Precision:
  At    5 docs:   0.4000
  At   10 docs:   0.2000
  At   15 docs:   0.2000
  At   20 docs:   0.2000
  At   30 docs:   0.1333
  At  100 docs:   0.0400
  At  200 docs:   0.0250
  At  500 docs:   0.0100
  At 1000 docs:   0.0050
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.4000
```

```
Queryid (Num):        19
Total number of documents over all queries
    Retrieved:       96
    Relevant:         5
    Rel_ret:          5
Interpolated Recall - Precision Averages:
    at 0.00       1.0000
    at 0.10       1.0000
    at 0.20       1.0000
    at 0.30       0.3750
    at 0.40       0.3750
    at 0.50       0.3750
    at 0.60       0.3750
    at 0.70       0.2500
    at 0.80       0.2500
    at 0.90       0.2500
    at 1.00       0.2500
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.4242
Precision:
  At    5 docs:   0.2000
  At   10 docs:   0.3000
  At   15 docs:   0.2000
  At   20 docs:   0.2500
  At   30 docs:   0.1667
  At  100 docs:   0.0500
  At  200 docs:   0.0250
  At  500 docs:   0.0100
  At 1000 docs:   0.0050
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.2000
```

```
Queryid (Num):        20
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         2
    Rel_ret:          2
Interpolated Recall - Precision Averages:
    at 0.00       0.0833
    at 0.10       0.0833
    at 0.20       0.0833
    at 0.30       0.0833
    at 0.40       0.0833
    at 0.50       0.0833
    at 0.60       0.0408
    at 0.70       0.0408
    at 0.80       0.0408
    at 0.90       0.0408
    at 1.00       0.0408
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0621
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.0000
  At   15 docs:   0.0667
  At   20 docs:   0.0500
  At   30 docs:   0.0333
  At  100 docs:   0.0200
  At  200 docs:   0.0100
  At  500 docs:   0.0040
  At 1000 docs:   0.0020
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):        21
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         1
    Rel_ret:          1
Interpolated Recall - Precision Averages:
    at 0.00       1.0000
    at 0.10       1.0000
    at 0.20       1.0000
    at 0.30       1.0000
    at 0.40       1.0000
    at 0.50       1.0000
    at 0.60       1.0000
    at 0.70       1.0000
    at 0.80       1.0000
    at 0.90       1.0000
    at 1.00       1.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  1.0000
Precision:
  At    5 docs:   0.2000
  At   10 docs:   0.1000
  At   15 docs:   0.0667
  At   20 docs:   0.0500
  At   30 docs:   0.0333
  At  100 docs:   0.0100
  At  200 docs:   0.0050
  At  500 docs:   0.0020
  At 1000 docs:   0.0010
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        1.0000
```

```
Queryid (Num):        22
Total number of documents over all queries
    Retrieved:     1000
    Relevant:        11
    Rel_ret:          9
Interpolated Recall - Precision Averages:
    at 0.00       0.1579
    at 0.10       0.1579
    at 0.20       0.1579
    at 0.30       0.0976
    at 0.40       0.0735
    at 0.50       0.0706
    at 0.60       0.0380
    at 0.70       0.0279
    at 0.80       0.0278
    at 0.90       0.0000
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0676
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.1000
  At   15 docs:   0.0667
  At   20 docs:   0.1500
  At   30 docs:   0.1000
  At  100 docs:   0.0600
  At  200 docs:   0.0350
  At  500 docs:   0.0180
  At 1000 docs:   0.0090
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0909
```

```
Queryid (Num):        23
Total number of documents over all queries
    Retrieved:        0
    Relevant:         0
    Rel_ret:          0
Interpolated Recall - Precision Averages:
    at 0.00       0.0000
    at 0.10       0.0000
    at 0.20       0.0000
    at 0.30       0.0000
    at 0.40       0.0000
    at 0.50       0.0000
    at 0.60       0.0000
    at 0.70       0.0000
    at 0.80       0.0000
    at 0.90       0.0000
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0000
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.0000
  At   15 docs:   0.0000
  At   20 docs:   0.0000
  At   30 docs:   0.0000
  At  100 docs:   0.0000
  At  200 docs:   0.0000
  At  500 docs:   0.0000
  At 1000 docs:   0.0000
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):        24
Total number of documents over all queries
    Retrieved:     1000
    Relevant:        12
    Rel_ret:          4
Interpolated Recall - Precision Averages:
    at 0.00       0.0833
    at 0.10       0.0345
    at 0.20       0.0081
    at 0.30       0.0081
    at 0.40       0.0000
    at 0.50       0.0000
    at 0.60       0.0000
    at 0.70       0.0000
    at 0.80       0.0000
    at 0.90       0.0000
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0111
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.0000
  At   15 docs:   0.0667
  At   20 docs:   0.0500
  At   30 docs:   0.0333
  At  100 docs:   0.0200
  At  200 docs:   0.0100
  At  500 docs:   0.0080
  At 1000 docs:   0.0040
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0833
```

```
Queryid (Num):        25
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         2
    Rel_ret:          2
Interpolated Recall - Precision Averages:
    at 0.00       0.0952
    at 0.10       0.0952
    at 0.20       0.0952
    at 0.30       0.0952
    at 0.40       0.0952
    at 0.50       0.0952
    at 0.60       0.0952
    at 0.70       0.0952
    at 0.80       0.0952
    at 0.90       0.0952
    at 1.00       0.0952
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0833
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.0000
  At   15 docs:   0.0667
  At   20 docs:   0.0500
  At   30 docs:   0.0667
  At  100 docs:   0.0200
  At  200 docs:   0.0100
  At  500 docs:   0.0040
  At 1000 docs:   0.0020
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):        26
Total number of documents over all queries
    Retrieved:     1000
    Relevant:        19
    Rel_ret:         19
Interpolated Recall - Precision Averages:
    at 0.00       0.6667
    at 0.10       0.6667
    at 0.20       0.6316
    at 0.30       0.6316
    at 0.40       0.6316
    at 0.50       0.6316
    at 0.60       0.6316
    at 0.70       0.3810
    at 0.80       0.3810
    at 0.90       0.0293
    at 1.00       0.0293
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.4527
Precision:
  At    5 docs:   0.6000
  At   10 docs:   0.5000
  At   15 docs:   0.6000
  At   20 docs:   0.6000
  At   30 docs:   0.4000
  At  100 docs:   0.1600
  At  200 docs:   0.0850
  At  500 docs:   0.0340
  At 1000 docs:   0.0190
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.6316
```

```
Queryid (Num):          27
Total number of documents over all queries
    Retrieved:          0
    Relevant:           0
    Rel_ret:            0
Interpolated Recall - Precision Averages:
    at 0.00         0.0000
    at 0.10         0.0000
    at 0.20         0.0000
    at 0.30         0.0000
    at 0.40         0.0000
    at 0.50         0.0000
    at 0.60         0.0000
    at 0.70         0.0000
    at 0.80         0.0000
    at 0.90         0.0000
    at 1.00         0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                    0.0000
Precision:
  At    5 docs:     0.0000
  At   10 docs:     0.0000
  At   15 docs:     0.0000
  At   20 docs:     0.0000
  At   30 docs:     0.0000
  At  100 docs:     0.0000
  At  200 docs:     0.0000
  At  500 docs:     0.0000
  At 1000 docs:     0.0000
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:          0.0000
```

```
Queryid (Num):        28
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         2
    Rel_ret:          2
Interpolated Recall - Precision Averages:
    at 0.00       0.5000
    at 0.10       0.5000
    at 0.20       0.5000
    at 0.30       0.5000
    at 0.40       0.5000
    at 0.50       0.5000
    at 0.60       0.0027
    at 0.70       0.0027
    at 0.80       0.0027
    at 0.90       0.0027
    at 1.00       0.0027
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.2514
Precision:
  At    5 docs:   0.2000
  At   10 docs:   0.1000
  At   15 docs:   0.0667
  At   20 docs:   0.0500
  At   30 docs:   0.0333
  At  100 docs:   0.0100
  At  200 docs:   0.0050
  At  500 docs:   0.0020
  At 1000 docs:   0.0020
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.5000
```

```
Queryid (Num):        29
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         5
    Rel_ret:          4
Interpolated Recall - Precision Averages:
    at 0.00       1.0000
    at 0.10       1.0000
    at 0.20       1.0000
    at 0.30       0.7500
    at 0.40       0.7500
    at 0.50       0.7500
    at 0.60       0.7500
    at 0.70       0.1905
    at 0.80       0.1905
    at 0.90       0.0000
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.5214
Precision:
  At    5 docs:   0.6000
  At   10 docs:   0.3000
  At   15 docs:   0.2000
  At   20 docs:   0.1500
  At   30 docs:   0.1333
  At  100 docs:   0.0400
  At  200 docs:   0.0200
  At  500 docs:   0.0080
  At 1000 docs:   0.0040
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.6000
```

```
Queryid (Num):          30
Total number of documents over all queries
    Retrieved:          0
    Relevant:           0
    Rel_ret:            0
Interpolated Recall - Precision Averages:
    at 0.00         0.0000
    at 0.10         0.0000
    at 0.20         0.0000
    at 0.30         0.0000
    at 0.40         0.0000
    at 0.50         0.0000
    at 0.60         0.0000
    at 0.70         0.0000
    at 0.80         0.0000
    at 0.90         0.0000
    at 1.00         0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                    0.0000
Precision:
  At    5 docs:     0.0000
  At   10 docs:     0.0000
  At   15 docs:     0.0000
  At   20 docs:     0.0000
  At   30 docs:     0.0000
  At  100 docs:     0.0000
  At  200 docs:     0.0000
  At  500 docs:     0.0000
  At 1000 docs:     0.0000
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:          0.0000
```

```
Queryid (Num):        31
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         3
    Rel_ret:          3
Interpolated Recall - Precision Averages:
    at 0.00       0.3000
    at 0.10       0.3000
    at 0.20       0.3000
    at 0.30       0.3000
    at 0.40       0.3000
    at 0.50       0.3000
    at 0.60       0.3000
    at 0.70       0.3000
    at 0.80       0.3000
    at 0.90       0.3000
    at 1.00       0.3000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.2217
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.3000
  At   15 docs:   0.2000
  At   20 docs:   0.1500
  At   30 docs:   0.1000
  At  100 docs:   0.0300
  At  200 docs:   0.0150
  At  500 docs:   0.0060
  At 1000 docs:   0.0030
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):        32
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         2
    Rel_ret:          2
Interpolated Recall - Precision Averages:
    at 0.00       1.0000
    at 0.10       1.0000
    at 0.20       1.0000
    at 0.30       1.0000
    at 0.40       1.0000
    at 0.50       1.0000
    at 0.60       1.0000
    at 0.70       1.0000
    at 0.80       1.0000
    at 0.90       1.0000
    at 1.00       1.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  1.0000
Precision:
  At    5 docs:   0.4000
  At   10 docs:   0.2000
  At   15 docs:   0.1333
  At   20 docs:   0.1000
  At   30 docs:   0.0667
  At  100 docs:   0.0200
  At  200 docs:   0.0100
  At  500 docs:   0.0040
  At 1000 docs:   0.0020
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        1.0000
```

Queryid (Num): 33
Total number of documents over all queries
Retrieved: 1000
Relevant: 3
Rel_ret: 3
Interpolated Recall - Precision Averages:
at 0.00 1.0000
at 0.10 1.0000
at 0.20 1.0000
at 0.30 1.0000
at 0.40 1.0000
at 0.50 1.0000
at 0.60 1.0000
at 0.70 0.0070
at 0.80 0.0070
at 0.90 0.0070
at 1.00 0.0070
Average precision (non-interpolated) for all rel docs(averaged over queries)
0.6690
Precision:
At 5 docs: 0.4000
At 10 docs: 0.2000
At 15 docs: 0.1333
At 20 docs: 0.1000
At 30 docs: 0.0667
At 100 docs: 0.0200
At 200 docs: 0.0100
At 500 docs: 0.0060
At 1000 docs: 0.0030
R-Precision (precision after R (= num_rel for a query) docs retrieved):
Exact: 0.6667

```
Queryid (Num):        34
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         2
    Rel_ret:          2
Interpolated Recall - Precision Averages:
    at 0.00       0.5000
    at 0.10       0.5000
    at 0.20       0.5000
    at 0.30       0.5000
    at 0.40       0.5000
    at 0.50       0.5000
    at 0.60       0.5000
    at 0.70       0.5000
    at 0.80       0.5000
    at 0.90       0.5000
    at 1.00       0.5000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.4167
Precision:
  At    5 docs:   0.4000
  At   10 docs:   0.2000
  At   15 docs:   0.1333
  At   20 docs:   0.1000
  At   30 docs:   0.0667
  At  100 docs:   0.0200
  At  200 docs:   0.0100
  At  500 docs:   0.0040
  At 1000 docs:   0.0020
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):        35
Total number of documents over all queries
    Retrieved:        0
    Relevant:         0
    Rel_ret:          0
Interpolated Recall - Precision Averages:
    at 0.00       0.0000
    at 0.10       0.0000
    at 0.20       0.0000
    at 0.30       0.0000
    at 0.40       0.0000
    at 0.50       0.0000
    at 0.60       0.0000
    at 0.70       0.0000
    at 0.80       0.0000
    at 0.90       0.0000
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0000
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.0000
  At   15 docs:   0.0000
  At   20 docs:   0.0000
  At   30 docs:   0.0000
  At  100 docs:   0.0000
  At  200 docs:   0.0000
  At  500 docs:   0.0000
  At 1000 docs:   0.0000
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):        36
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         3
    Rel_ret:          3
Interpolated Recall - Precision Averages:
    at 0.00       1.0000
    at 0.10       1.0000
    at 0.20       1.0000
    at 0.30       1.0000
    at 0.40       1.0000
    at 0.50       1.0000
    at 0.60       1.0000
    at 0.70       0.7500
    at 0.80       0.7500
    at 0.90       0.7500
    at 1.00       0.7500
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.9167
Precision:
  At    5 docs:   0.6000
  At   10 docs:   0.3000
  At   15 docs:   0.2000
  At   20 docs:   0.1500
  At   30 docs:   0.1000
  At  100 docs:   0.0300
  At  200 docs:   0.0150
  At  500 docs:   0.0060
  At 1000 docs:   0.0030
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.6667
```

```
Queryid (Num):        37
Total number of documents over all queries
    Retrieved:        0
    Relevant:         0
    Rel_ret:          0
Interpolated Recall - Precision Averages:
    at 0.00       0.0000
    at 0.10       0.0000
    at 0.20       0.0000
    at 0.30       0.0000
    at 0.40       0.0000
    at 0.50       0.0000
    at 0.60       0.0000
    at 0.70       0.0000
    at 0.80       0.0000
    at 0.90       0.0000
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0000
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.0000
  At   15 docs:   0.0000
  At   20 docs:   0.0000
  At   30 docs:   0.0000
  At  100 docs:   0.0000
  At  200 docs:   0.0000
  At  500 docs:   0.0000
  At 1000 docs:   0.0000
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):        38
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         8
    Rel_ret:          6
Interpolated Recall - Precision Averages:
    at 0.00       0.5000
    at 0.10       0.5000
    at 0.20       0.0400
    at 0.30       0.0400
    at 0.40       0.0400
    at 0.50       0.0400
    at 0.60       0.0258
    at 0.70       0.0258
    at 0.80       0.0000
    at 0.90       0.0000
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0824
Precision:
  At    5 docs:   0.2000
  At   10 docs:   0.1000
  At   15 docs:   0.0667
  At   20 docs:   0.0500
  At   30 docs:   0.0333
  At  100 docs:   0.0400
  At  200 docs:   0.0200
  At  500 docs:   0.0120
  At 1000 docs:   0.0060
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.1250
```

```
Queryid (Num):          39
Total number of documents over all queries
    Retrieved:          0
    Relevant:           0
    Rel_ret:            0
Interpolated Recall - Precision Averages:
    at 0.00         0.0000
    at 0.10         0.0000
    at 0.20         0.0000
    at 0.30         0.0000
    at 0.40         0.0000
    at 0.50         0.0000
    at 0.60         0.0000
    at 0.70         0.0000
    at 0.80         0.0000
    at 0.90         0.0000
    at 1.00         0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                    0.0000
Precision:
  At    5 docs:     0.0000
  At   10 docs:     0.0000
  At   15 docs:     0.0000
  At   20 docs:     0.0000
  At   30 docs:     0.0000
  At  100 docs:     0.0000
  At  200 docs:     0.0000
  At  500 docs:     0.0000
  At 1000 docs:     0.0000
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:          0.0000
```

```
Queryid (Num):        40
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         5
    Rel_ret:          4
Interpolated Recall - Precision Averages:
    at 0.00       0.5000
    at 0.10       0.5000
    at 0.20       0.5000
    at 0.30       0.2222
    at 0.40       0.2222
    at 0.50       0.0064
    at 0.60       0.0064
    at 0.70       0.0051
    at 0.80       0.0051
    at 0.90       0.0000
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.1468
Precision:
  At    5 docs:   0.2000
  At   10 docs:   0.2000
  At   15 docs:   0.1333
  At   20 docs:   0.1000
  At   30 docs:   0.0667
  At  100 docs:   0.0200
  At  200 docs:   0.0100
  At  500 docs:   0.0060
  At 1000 docs:   0.0040
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.2000
```

```
Queryid (Num):        41
Total number of documents over all queries
    Retrieved:        0
    Relevant:         0
    Rel_ret:          0
Interpolated Recall - Precision Averages:
    at 0.00       0.0000
    at 0.10       0.0000
    at 0.20       0.0000
    at 0.30       0.0000
    at 0.40       0.0000
    at 0.50       0.0000
    at 0.60       0.0000
    at 0.70       0.0000
    at 0.80       0.0000
    at 0.90       0.0000
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0000
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.0000
  At   15 docs:   0.0000
  At   20 docs:   0.0000
  At   30 docs:   0.0000
  At  100 docs:   0.0000
  At  200 docs:   0.0000
  At  500 docs:   0.0000
  At 1000 docs:   0.0000
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):        42
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         3
    Rel_ret:          3
Interpolated Recall - Precision Averages:
    at 0.00       0.0294
    at 0.10       0.0294
    at 0.20       0.0294
    at 0.30       0.0294
    at 0.40       0.0146
    at 0.50       0.0146
    at 0.60       0.0146
    at 0.70       0.0035
    at 0.80       0.0035
    at 0.90       0.0035
    at 1.00       0.0035
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0158
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.0000
  At   15 docs:   0.0000
  At   20 docs:   0.0000
  At   30 docs:   0.0000
  At  100 docs:   0.0100
  At  200 docs:   0.0100
  At  500 docs:   0.0040
  At 1000 docs:   0.0030
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):        43
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         8
    Rel_ret:          8
Interpolated Recall - Precision Averages:
    at 0.00       1.0000
    at 0.10       1.0000
    at 0.20       1.0000
    at 0.30       1.0000
    at 0.40       1.0000
    at 0.50       1.0000
    at 0.60       0.5556
    at 0.70       0.4286
    at 0.80       0.3684
    at 0.90       0.0274
    at 1.00       0.0274
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.6725
Precision:
  At    5 docs:   0.8000
  At   10 docs:   0.5000
  At   15 docs:   0.4000
  At   20 docs:   0.3500
  At   30 docs:   0.2333
  At  100 docs:   0.0700
  At  200 docs:   0.0350
  At  500 docs:   0.0160
  At 1000 docs:   0.0080
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.5000
```

```
Queryid (Num):        44
Total number of documents over all queries
    Retrieved:     1000
    Relevant:        26
    Rel_ret:         26
Interpolated Recall - Precision Averages:
    at 0.00       1.0000
    at 0.10       1.0000
    at 0.20       0.5385
    at 0.30       0.5000
    at 0.40       0.5000
    at 0.50       0.1831
    at 0.60       0.1417
    at 0.70       0.1242
    at 0.80       0.1068
    at 0.90       0.0836
    at 1.00       0.0615
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.3603
Precision:
  At    5 docs:   0.8000
  At   10 docs:   0.5000
  At   15 docs:   0.4667
  At   20 docs:   0.4500
  At   30 docs:   0.3667
  At  100 docs:   0.1400
  At  200 docs:   0.1000
  At  500 docs:   0.0520
  At 1000 docs:   0.0260
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.4231
```

```
Queryid (Num):        45
Total number of documents over all queries
    Retrieved:      284
    Relevant:        26
    Rel_ret:         10
Interpolated Recall - Precision Averages:
    at 0.00       1.0000
    at 0.10       0.3750
    at 0.20       0.0366
    at 0.30       0.0366
    at 0.40       0.0000
    at 0.50       0.0000
    at 0.60       0.0000
    at 0.70       0.0000
    at 0.80       0.0000
    at 0.90       0.0000
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.1008
Precision:
  At    5 docs:   0.4000
  At   10 docs:   0.3000
  At   15 docs:   0.2000
  At   20 docs:   0.1500
  At   30 docs:   0.1667
  At  100 docs:   0.0500
  At  200 docs:   0.0300
  At  500 docs:   0.0200
  At 1000 docs:   0.0100
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.1923
```

```
Queryid (Num):        46
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         2
    Rel_ret:          2
Interpolated Recall - Precision Averages:
    at 0.00       0.1667
    at 0.10       0.1667
    at 0.20       0.1667
    at 0.30       0.1667
    at 0.40       0.1667
    at 0.50       0.1667
    at 0.60       0.0833
    at 0.70       0.0833
    at 0.80       0.0833
    at 0.90       0.0833
    at 1.00       0.0833
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.1250
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.1000
  At   15 docs:   0.0667
  At   20 docs:   0.0500
  At   30 docs:   0.0667
  At  100 docs:   0.0200
  At  200 docs:   0.0100
  At  500 docs:   0.0040
  At 1000 docs:   0.0020
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):        47
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         3
    Rel_ret:          3
Interpolated Recall - Precision Averages:
    at 0.00       0.1111
    at 0.10       0.1111
    at 0.20       0.1111
    at 0.30       0.1111
    at 0.40       0.1034
    at 0.50       0.1034
    at 0.60       0.1034
    at 0.70       0.1034
    at 0.80       0.1034
    at 0.90       0.1034
    at 1.00       0.1034
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.1049
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.1000
  At   15 docs:   0.0667
  At   20 docs:   0.1000
  At   30 docs:   0.1000
  At  100 docs:   0.0300
  At  200 docs:   0.0150
  At  500 docs:   0.0060
  At 1000 docs:   0.0030
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):        48
Total number of documents over all queries
    Retrieved:        0
    Relevant:         0
    Rel_ret:          0
Interpolated Recall - Precision Averages:
    at 0.00       0.0000
    at 0.10       0.0000
    at 0.20       0.0000
    at 0.30       0.0000
    at 0.40       0.0000
    at 0.50       0.0000
    at 0.60       0.0000
    at 0.70       0.0000
    at 0.80       0.0000
    at 0.90       0.0000
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0000
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.0000
  At   15 docs:   0.0000
  At   20 docs:   0.0000
  At   30 docs:   0.0000
  At  100 docs:   0.0000
  At  200 docs:   0.0000
  At  500 docs:   0.0000
  At 1000 docs:   0.0000
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):        49
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         2
    Rel_ret:          2
Interpolated Recall - Precision Averages:
    at 0.00       0.5000
    at 0.10       0.5000
    at 0.20       0.5000
    at 0.30       0.5000
    at 0.40       0.5000
    at 0.50       0.5000
    at 0.60       0.0157
    at 0.70       0.0157
    at 0.80       0.0157
    at 0.90       0.0157
    at 1.00       0.0157
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.2579
Precision:
  At    5 docs:   0.2000
  At   10 docs:   0.1000
  At   15 docs:   0.0667
  At   20 docs:   0.0500
  At   30 docs:   0.0333
  At  100 docs:   0.0100
  At  200 docs:   0.0100
  At  500 docs:   0.0040
  At 1000 docs:   0.0020
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.5000
```

```
Queryid (Num):        50
Total number of documents over all queries
    Retrieved:        0
    Relevant:         0
    Rel_ret:          0
Interpolated Recall - Precision Averages:
    at 0.00       0.0000
    at 0.10       0.0000
    at 0.20       0.0000
    at 0.30       0.0000
    at 0.40       0.0000
    at 0.50       0.0000
    at 0.60       0.0000
    at 0.70       0.0000
    at 0.80       0.0000
    at 0.90       0.0000
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0000
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.0000
  At   15 docs:   0.0000
  At   20 docs:   0.0000
  At   30 docs:   0.0000
  At  100 docs:   0.0000
  At  200 docs:   0.0000
  At  500 docs:   0.0000
  At 1000 docs:   0.0000
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

Best Guess Parameter Ranking Experiment (link3)

Combining Linkage and Content evidence using 'best-guess' parameters

```
Queryid (Num):        1
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         7
    Rel_ret:          4
Interpolated Recall - Precision Averages:
    at 0.00       0.5000
    at 0.10       0.5000
    at 0.20       0.5000
    at 0.30       0.1765
    at 0.40       0.1765
    at 0.50       0.0059
    at 0.60       0.0000
    at 0.70       0.0000
    at 0.80       0.0000
    at 0.90       0.0000
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.1689
Precision:
  At    5 docs:   0.4000
  At   10 docs:   0.2000
  At   15 docs:   0.1333
  At   20 docs:   0.1500
  At   30 docs:   0.1000
  At  100 docs:   0.0300
  At  200 docs:   0.0150
  At  500 docs:   0.0060
  At 1000 docs:   0.0040
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.2857
```

```
Queryid (Num):        2
Total number of documents over all queries
    Retrieved:     1000
    Relevant:        16
    Rel_ret:         16
Interpolated Recall - Precision Averages:
    at 0.00       0.1481
    at 0.10       0.1481
    at 0.20       0.1481
    at 0.30       0.1034
    at 0.40       0.0326
    at 0.50       0.0272
    at 0.60       0.0254
    at 0.70       0.0168
    at 0.80       0.0168
    at 0.90       0.0168
    at 1.00       0.0168
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0538
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.0000
  At   15 docs:   0.0667
  At   20 docs:   0.1000
  At   30 docs:   0.1333
  At  100 docs:   0.0600
  At  200 docs:   0.0300
  At  500 docs:   0.0200
  At 1000 docs:   0.0160
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0625
```

```
Queryid (Num):        3
Total number of documents over all queries
    Retrieved:     1000
    Relevant:        16
    Rel_ret:         15
Interpolated Recall - Precision Averages:
    at 0.00       1.0000
    at 0.10       1.0000
    at 0.20       1.0000
    at 0.30       0.8333
    at 0.40       0.6429
    at 0.50       0.6429
    at 0.60       0.5263
    at 0.70       0.5217
    at 0.80       0.3095
    at 0.90       0.1786
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.5985
Precision:
  At    5 docs:   0.8000
  At   10 docs:   0.6000
  At   15 docs:   0.6000
  At   20 docs:   0.5000
  At   30 docs:   0.4000
  At  100 docs:   0.1500
  At  200 docs:   0.0750
  At  500 docs:   0.0300
  At 1000 docs:   0.0150
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.5625
```

```
Queryid (Num):        4
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         8
    Rel_ret:          8
Interpolated Recall - Precision Averages:
    at 0.00       1.0000
    at 0.10       1.0000
    at 0.20       0.5000
    at 0.30       0.5000
    at 0.40       0.5000
    at 0.50       0.5000
    at 0.60       0.2273
    at 0.70       0.1429
    at 0.80       0.1321
    at 0.90       0.1212
    at 1.00       0.1212
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.3779
Precision:
  At    5 docs:   0.4000
  At   10 docs:   0.4000
  At   15 docs:   0.2667
  At   20 docs:   0.2000
  At   30 docs:   0.1667
  At  100 docs:   0.0800
  At  200 docs:   0.0400
  At  500 docs:   0.0160
  At 1000 docs:   0.0080
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.5000
```

```
Queryid (Num):        5
Total number of documents over all queries
    Retrieved:        0
    Relevant:         0
    Rel_ret:          0
Interpolated Recall - Precision Averages:
    at 0.00       0.0000
    at 0.10       0.0000
    at 0.20       0.0000
    at 0.30       0.0000
    at 0.40       0.0000
    at 0.50       0.0000
    at 0.60       0.0000
    at 0.70       0.0000
    at 0.80       0.0000
    at 0.90       0.0000
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0000
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.0000
  At   15 docs:   0.0000
  At   20 docs:   0.0000
  At   30 docs:   0.0000
  At  100 docs:   0.0000
  At  200 docs:   0.0000
  At  500 docs:   0.0000
  At 1000 docs:   0.0000
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):        6
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         2
    Rel_ret:          2
Interpolated Recall - Precision Averages:
    at 0.00       0.0588
    at 0.10       0.0588
    at 0.20       0.0588
    at 0.30       0.0588
    at 0.40       0.0588
    at 0.50       0.0588
    at 0.60       0.0094
    at 0.70       0.0094
    at 0.80       0.0094
    at 0.90       0.0094
    at 1.00       0.0094
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0341
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.0000
  At   15 docs:   0.0000
  At   20 docs:   0.0500
  At   30 docs:   0.0333
  At  100 docs:   0.0100
  At  200 docs:   0.0050
  At  500 docs:   0.0040
  At 1000 docs:   0.0020
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):        7
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         6
    Rel_ret:          6
Interpolated Recall - Precision Averages:
    at 0.00       0.1000
    at 0.10       0.1000
    at 0.20       0.0645
    at 0.30       0.0645
    at 0.40       0.0645
    at 0.50       0.0645
    at 0.60       0.0645
    at 0.70       0.0442
    at 0.80       0.0442
    at 0.90       0.0337
    at 1.00       0.0337
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0545
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.1000
  At   15 docs:   0.0667
  At   20 docs:   0.0500
  At   30 docs:   0.0333
  At  100 docs:   0.0400
  At  200 docs:   0.0300
  At  500 docs:   0.0120
  At 1000 docs:   0.0060
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):          8
Total number of documents over all queries
    Retrieved:          0
    Relevant:           0
    Rel_ret:            0
Interpolated Recall - Precision Averages:
    at 0.00         0.0000
    at 0.10         0.0000
    at 0.20         0.0000
    at 0.30         0.0000
    at 0.40         0.0000
    at 0.50         0.0000
    at 0.60         0.0000
    at 0.70         0.0000
    at 0.80         0.0000
    at 0.90         0.0000
    at 1.00         0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                    0.0000
Precision:
  At    5 docs:     0.0000
  At   10 docs:     0.0000
  At   15 docs:     0.0000
  At   20 docs:     0.0000
  At   30 docs:     0.0000
  At  100 docs:     0.0000
  At  200 docs:     0.0000
  At  500 docs:     0.0000
  At 1000 docs:     0.0000
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:          0.0000
```

```
Queryid (Num):        9
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         1
    Rel_ret:          0
Interpolated Recall - Precision Averages:
    at 0.00       0.0000
    at 0.10       0.0000
    at 0.20       0.0000
    at 0.30       0.0000
    at 0.40       0.0000
    at 0.50       0.0000
    at 0.60       0.0000
    at 0.70       0.0000
    at 0.80       0.0000
    at 0.90       0.0000
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0000
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.0000
  At   15 docs:   0.0000
  At   20 docs:   0.0000
  At   30 docs:   0.0000
  At  100 docs:   0.0000
  At  200 docs:   0.0000
  At  500 docs:   0.0000
  At 1000 docs:   0.0000
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):        10
Total number of documents over all queries
    Retrieved:        0
    Relevant:         0
    Rel_ret:          0
Interpolated Recall - Precision Averages:
    at 0.00       0.0000
    at 0.10       0.0000
    at 0.20       0.0000
    at 0.30       0.0000
    at 0.40       0.0000
    at 0.50       0.0000
    at 0.60       0.0000
    at 0.70       0.0000
    at 0.80       0.0000
    at 0.90       0.0000
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0000
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.0000
  At   15 docs:   0.0000
  At   20 docs:   0.0000
  At   30 docs:   0.0000
  At  100 docs:   0.0000
  At  200 docs:   0.0000
  At  500 docs:   0.0000
  At 1000 docs:   0.0000
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):        11
Total number of documents over all queries
    Retrieved:      816
    Relevant:         1
    Rel_ret:          1
Interpolated Recall - Precision Averages:
    at 0.00       0.0222
    at 0.10       0.0222
    at 0.20       0.0222
    at 0.30       0.0222
    at 0.40       0.0222
    at 0.50       0.0222
    at 0.60       0.0222
    at 0.70       0.0222
    at 0.80       0.0222
    at 0.90       0.0222
    at 1.00       0.0222
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0222
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.0000
  At   15 docs:   0.0000
  At   20 docs:   0.0000
  At   30 docs:   0.0000
  At  100 docs:   0.0100
  At  200 docs:   0.0050
  At  500 docs:   0.0020
  At 1000 docs:   0.0010
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):         12
Total number of documents over all queries
    Retrieved:     1000
    Relevant:        11
    Rel_ret:         11
Interpolated Recall - Precision Averages:
    at 0.00       0.2500
    at 0.10       0.2500
    at 0.20       0.2500
    at 0.30       0.2500
    at 0.40       0.1613
    at 0.50       0.1429
    at 0.60       0.0762
    at 0.70       0.0762
    at 0.80       0.0474
    at 0.90       0.0379
    at 1.00       0.0377
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.1378
Precision:
  At    5 docs:   0.2000
  At   10 docs:   0.2000
  At   15 docs:   0.2000
  At   20 docs:   0.2000
  At   30 docs:   0.1333
  At  100 docs:   0.0700
  At  200 docs:   0.0450
  At  500 docs:   0.0220
  At 1000 docs:   0.0110
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.1818
```

```
Queryid (Num):        13
Total number of documents over all queries
    Retrieved:     1000
    Relevant:        16
    Rel_ret:         15
Interpolated Recall - Precision Averages:
    at 0.00       1.0000
    at 0.10       0.6667
    at 0.20       0.3684
    at 0.30       0.3684
    at 0.40       0.3684
    at 0.50       0.3448
    at 0.60       0.3448
    at 0.70       0.1008
    at 0.80       0.0890
    at 0.90       0.0401
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.3001
Precision:
  At    5 docs:   0.4000
  At   10 docs:   0.2000
  At   15 docs:   0.3333
  At   20 docs:   0.3500
  At   30 docs:   0.3333
  At  100 docs:   0.1100
  At  200 docs:   0.0650
  At  500 docs:   0.0300
  At 1000 docs:   0.0150
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.3125
```

```
Queryid (Num):        14
Total number of documents over all queries
    Retrieved:        0
    Relevant:         0
    Rel_ret:          0
Interpolated Recall - Precision Averages:
    at 0.00       0.0000
    at 0.10       0.0000
    at 0.20       0.0000
    at 0.30       0.0000
    at 0.40       0.0000
    at 0.50       0.0000
    at 0.60       0.0000
    at 0.70       0.0000
    at 0.80       0.0000
    at 0.90       0.0000
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0000
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.0000
  At   15 docs:   0.0000
  At   20 docs:   0.0000
  At   30 docs:   0.0000
  At  100 docs:   0.0000
  At  200 docs:   0.0000
  At  500 docs:   0.0000
  At 1000 docs:   0.0000
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):        15
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         2
    Rel_ret:          2
Interpolated Recall - Precision Averages:
    at 0.00       0.0571
    at 0.10       0.0571
    at 0.20       0.0571
    at 0.30       0.0571
    at 0.40       0.0571
    at 0.50       0.0571
    at 0.60       0.0571
    at 0.70       0.0571
    at 0.80       0.0571
    at 0.90       0.0571
    at 1.00       0.0571
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0494
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.0000
  At   15 docs:   0.0000
  At   20 docs:   0.0000
  At   30 docs:   0.0333
  At  100 docs:   0.0200
  At  200 docs:   0.0100
  At  500 docs:   0.0040
  At 1000 docs:   0.0020
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):        16
Total number of documents over all queries
    Retrieved:        0
    Relevant:         0
    Rel_ret:          0
Interpolated Recall - Precision Averages:
    at 0.00       0.0000
    at 0.10       0.0000
    at 0.20       0.0000
    at 0.30       0.0000
    at 0.40       0.0000
    at 0.50       0.0000
    at 0.60       0.0000
    at 0.70       0.0000
    at 0.80       0.0000
    at 0.90       0.0000
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0000
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.0000
  At   15 docs:   0.0000
  At   20 docs:   0.0000
  At   30 docs:   0.0000
  At  100 docs:   0.0000
  At  200 docs:   0.0000
  At  500 docs:   0.0000
  At 1000 docs:   0.0000
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):        17
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         8
    Rel_ret:          8
Interpolated Recall - Precision Averages:
    at 0.00       0.3333
    at 0.10       0.3333
    at 0.20       0.3333
    at 0.30       0.3333
    at 0.40       0.3125
    at 0.50       0.3125
    at 0.60       0.3125
    at 0.70       0.1346
    at 0.80       0.1346
    at 0.90       0.1081
    at 1.00       0.1081
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.2115
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.3000
  At   15 docs:   0.2667
  At   20 docs:   0.2500
  At   30 docs:   0.1667
  At  100 docs:   0.0800
  At  200 docs:   0.0400
  At  500 docs:   0.0160
  At 1000 docs:   0.0080
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.2500
```

```
Queryid (Num):        18
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         5
    Rel_ret:          5
Interpolated Recall - Precision Averages:
    at 0.00       0.5000
    at 0.10       0.5000
    at 0.20       0.5000
    at 0.30       0.5000
    at 0.40       0.5000
    at 0.50       0.2000
    at 0.60       0.2000
    at 0.70       0.2000
    at 0.80       0.2000
    at 0.90       0.0442
    at 1.00       0.0442
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.2888
Precision:
  At    5 docs:   0.4000
  At   10 docs:   0.2000
  At   15 docs:   0.2000
  At   20 docs:   0.2000
  At   30 docs:   0.1333
  At  100 docs:   0.0400
  At  200 docs:   0.0250
  At  500 docs:   0.0100
  At 1000 docs:   0.0050
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.4000
```

```
Queryid (Num):          19
Total number of documents over all queries
    Retrieved:         96
    Relevant:           5
    Rel_ret:            5
Interpolated Recall - Precision Averages:
    at 0.00         1.0000
    at 0.10         1.0000
    at 0.20         1.0000
    at 0.30         0.3333
    at 0.40         0.3333
    at 0.50         0.3333
    at 0.60         0.3333
    at 0.70         0.2273
    at 0.80         0.2273
    at 0.90         0.2273
    at 1.00         0.2273
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                    0.4002
Precision:
  At    5 docs:     0.2000
  At   10 docs:     0.3000
  At   15 docs:     0.2000
  At   20 docs:     0.1500
  At   30 docs:     0.1667
  At  100 docs:     0.0500
  At  200 docs:     0.0250
  At  500 docs:     0.0100
  At 1000 docs:     0.0050
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:          0.2000
```

```
Queryid (Num):        20
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         2
    Rel_ret:          2
Interpolated Recall - Precision Averages:
    at 0.00       0.0833
    at 0.10       0.0833
    at 0.20       0.0833
    at 0.30       0.0833
    at 0.40       0.0833
    at 0.50       0.0833
    at 0.60       0.0408
    at 0.70       0.0408
    at 0.80       0.0408
    at 0.90       0.0408
    at 1.00       0.0408
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0621
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.0000
  At   15 docs:   0.0667
  At   20 docs:   0.0500
  At   30 docs:   0.0333
  At  100 docs:   0.0200
  At  200 docs:   0.0100
  At  500 docs:   0.0040
  At 1000 docs:   0.0020
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):        21
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         1
    Rel_ret:          1
Interpolated Recall - Precision Averages:
    at 0.00       1.0000
    at 0.10       1.0000
    at 0.20       1.0000
    at 0.30       1.0000
    at 0.40       1.0000
    at 0.50       1.0000
    at 0.60       1.0000
    at 0.70       1.0000
    at 0.80       1.0000
    at 0.90       1.0000
    at 1.00       1.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  1.0000
Precision:
  At    5 docs:   0.2000
  At   10 docs:   0.1000
  At   15 docs:   0.0667
  At   20 docs:   0.0500
  At   30 docs:   0.0333
  At  100 docs:   0.0100
  At  200 docs:   0.0050
  At  500 docs:   0.0020
  At 1000 docs:   0.0010
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        1.0000
```

```
Queryid (Num):        22
Total number of documents over all queries
    Retrieved:     1000
    Relevant:        11
    Rel_ret:          9
Interpolated Recall - Precision Averages:
    at 0.00       0.1579
    at 0.10       0.1579
    at 0.20       0.1579
    at 0.30       0.0952
    at 0.40       0.0725
    at 0.50       0.0706
    at 0.60       0.0370
    at 0.70       0.0289
    at 0.80       0.0289
    at 0.90       0.0000
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0665
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.1000
  At   15 docs:   0.0667
  At   20 docs:   0.1500
  At   30 docs:   0.1000
  At  100 docs:   0.0600
  At  200 docs:   0.0350
  At  500 docs:   0.0180
  At 1000 docs:   0.0090
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0909
```

```
Queryid (Num):        23
Total number of documents over all queries
    Retrieved:        0
    Relevant:         0
    Rel_ret:          0
Interpolated Recall - Precision Averages:
    at 0.00       0.0000
    at 0.10       0.0000
    at 0.20       0.0000
    at 0.30       0.0000
    at 0.40       0.0000
    at 0.50       0.0000
    at 0.60       0.0000
    at 0.70       0.0000
    at 0.80       0.0000
    at 0.90       0.0000
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0000
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.0000
  At   15 docs:   0.0000
  At   20 docs:   0.0000
  At   30 docs:   0.0000
  At  100 docs:   0.0000
  At  200 docs:   0.0000
  At  500 docs:   0.0000
  At 1000 docs:   0.0000
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):        24
Total number of documents over all queries
    Retrieved:     1000
    Relevant:        12
    Rel_ret:          4
Interpolated Recall - Precision Averages:
    at 0.00       0.0833
    at 0.10       0.0333
    at 0.20       0.0080
    at 0.30       0.0080
    at 0.40       0.0000
    at 0.50       0.0000
    at 0.60       0.0000
    at 0.70       0.0000
    at 0.80       0.0000
    at 0.90       0.0000
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0110
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.0000
  At   15 docs:   0.0667
  At   20 docs:   0.0500
  At   30 docs:   0.0333
  At  100 docs:   0.0200
  At  200 docs:   0.0100
  At  500 docs:   0.0080
  At 1000 docs:   0.0040
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0833
```

```
Queryid (Num):        25
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         2
    Rel_ret:          2
Interpolated Recall - Precision Averages:
    at 0.00       0.0870
    at 0.10       0.0870
    at 0.20       0.0870
    at 0.30       0.0870
    at 0.40       0.0870
    at 0.50       0.0870
    at 0.60       0.0870
    at 0.70       0.0870
    at 0.80       0.0870
    at 0.90       0.0870
    at 1.00       0.0870
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0792
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.0000
  At   15 docs:   0.0667
  At   20 docs:   0.0500
  At   30 docs:   0.0667
  At  100 docs:   0.0200
  At  200 docs:   0.0100
  At  500 docs:   0.0040
  At 1000 docs:   0.0020
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):        26
Total number of documents over all queries
    Retrieved:     1000
    Relevant:        19
    Rel_ret:         19
Interpolated Recall - Precision Averages:
    at 0.00       0.6667
    at 0.10       0.6667
    at 0.20       0.6316
    at 0.30       0.6316
    at 0.40       0.6316
    at 0.50       0.6316
    at 0.60       0.6316
    at 0.70       0.3721
    at 0.80       0.3721
    at 0.90       0.0293
    at 1.00       0.0293
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.4517
Precision:
  At    5 docs:   0.6000
  At   10 docs:   0.5000
  At   15 docs:   0.6000
  At   20 docs:   0.6000
  At   30 docs:   0.4000
  At  100 docs:   0.1600
  At  200 docs:   0.0850
  At  500 docs:   0.0340
  At 1000 docs:   0.0190
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.6316
```

```
Queryid (Num):        27
Total number of documents over all queries
    Retrieved:        0
    Relevant:         0
    Rel_ret:          0
Interpolated Recall - Precision Averages:
    at 0.00       0.0000
    at 0.10       0.0000
    at 0.20       0.0000
    at 0.30       0.0000
    at 0.40       0.0000
    at 0.50       0.0000
    at 0.60       0.0000
    at 0.70       0.0000
    at 0.80       0.0000
    at 0.90       0.0000
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0000
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.0000
  At   15 docs:   0.0000
  At   20 docs:   0.0000
  At   30 docs:   0.0000
  At  100 docs:   0.0000
  At  200 docs:   0.0000
  At  500 docs:   0.0000
  At 1000 docs:   0.0000
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):        28
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         2
    Rel_ret:          2
Interpolated Recall - Precision Averages:
    at 0.00       0.5000
    at 0.10       0.5000
    at 0.20       0.5000
    at 0.30       0.5000
    at 0.40       0.5000
    at 0.50       0.5000
    at 0.60       0.0028
    at 0.70       0.0028
    at 0.80       0.0028
    at 0.90       0.0028
    at 1.00       0.0028
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.2514
Precision:
  At    5 docs:   0.2000
  At   10 docs:   0.1000
  At   15 docs:   0.0667
  At   20 docs:   0.0500
  At   30 docs:   0.0333
  At  100 docs:   0.0100
  At  200 docs:   0.0050
  At  500 docs:   0.0020
  At 1000 docs:   0.0020
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.5000
```

```
Queryid (Num):        29
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         5
    Rel_ret:          4
Interpolated Recall - Precision Averages:
    at 0.00       1.0000
    at 0.10       1.0000
    at 0.20       1.0000
    at 0.30       0.7500
    at 0.40       0.7500
    at 0.50       0.7500
    at 0.60       0.7500
    at 0.70       0.1905
    at 0.80       0.1905
    at 0.90       0.0000
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.5214
Precision:
  At    5 docs:   0.6000
  At   10 docs:   0.3000
  At   15 docs:   0.2000
  At   20 docs:   0.1500
  At   30 docs:   0.1333
  At  100 docs:   0.0400
  At  200 docs:   0.0200
  At  500 docs:   0.0080
  At 1000 docs:   0.0040
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.6000
```

```
Queryid (Num):        30
Total number of documents over all queries
    Retrieved:        0
    Relevant:         0
    Rel_ret:          0
Interpolated Recall - Precision Averages:
    at 0.00       0.0000
    at 0.10       0.0000
    at 0.20       0.0000
    at 0.30       0.0000
    at 0.40       0.0000
    at 0.50       0.0000
    at 0.60       0.0000
    at 0.70       0.0000
    at 0.80       0.0000
    at 0.90       0.0000
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0000
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.0000
  At   15 docs:   0.0000
  At   20 docs:   0.0000
  At   30 docs:   0.0000
  At  100 docs:   0.0000
  At  200 docs:   0.0000
  At  500 docs:   0.0000
  At 1000 docs:   0.0000
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

Queryid (Num): 31
Total number of documents over all queries
Retrieved: 1000
Relevant: 3
Rel_ret: 3
Interpolated Recall - Precision Averages:
at 0.00 0.3000
at 0.10 0.3000
at 0.20 0.3000
at 0.30 0.3000
at 0.40 0.3000
at 0.50 0.3000
at 0.60 0.3000
at 0.70 0.3000
at 0.80 0.3000
at 0.90 0.3000
at 1.00 0.3000
Average precision (non-interpolated) for all rel docs(averaged over queries)
0.2217
Precision:
At 5 docs: 0.0000
At 10 docs: 0.3000
At 15 docs: 0.2000
At 20 docs: 0.1500
At 30 docs: 0.1000
At 100 docs: 0.0300
At 200 docs: 0.0150
At 500 docs: 0.0060
At 1000 docs: 0.0030
R-Precision (precision after R (= num_rel for a query) docs retrieved):
Exact: 0.0000

```
Queryid (Num):        32
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         2
    Rel_ret:          2
Interpolated Recall - Precision Averages:
    at 0.00       1.0000
    at 0.10       1.0000
    at 0.20       1.0000
    at 0.30       1.0000
    at 0.40       1.0000
    at 0.50       1.0000
    at 0.60       0.6667
    at 0.70       0.6667
    at 0.80       0.6667
    at 0.90       0.6667
    at 1.00       0.6667
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.8333
Precision:
  At    5 docs:   0.4000
  At   10 docs:   0.2000
  At   15 docs:   0.1333
  At   20 docs:   0.1000
  At   30 docs:   0.0667
  At  100 docs:   0.0200
  At  200 docs:   0.0100
  At  500 docs:   0.0040
  At 1000 docs:   0.0020
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.5000
```

```
Queryid (Num):          33
Total number of documents over all queries
    Retrieved:      1000
    Relevant:          3
    Rel_ret:           3
Interpolated Recall - Precision Averages:
    at 0.00        1.0000
    at 0.10        1.0000
    at 0.20        1.0000
    at 0.30        1.0000
    at 0.40        1.0000
    at 0.50        1.0000
    at 0.60        1.0000
    at 0.70        0.0070
    at 0.80        0.0070
    at 0.90        0.0070
    at 1.00        0.0070
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                   0.6690
Precision:
  At    5 docs:    0.4000
  At   10 docs:    0.2000
  At   15 docs:    0.1333
  At   20 docs:    0.1000
  At   30 docs:    0.0667
  At  100 docs:    0.0200
  At  200 docs:    0.0100
  At  500 docs:    0.0060
  At 1000 docs:    0.0030
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:         0.6667
```

```
Queryid (Num):        34
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         2
    Rel_ret:          2
Interpolated Recall - Precision Averages:
    at 0.00       0.5000
    at 0.10       0.5000
    at 0.20       0.5000
    at 0.30       0.5000
    at 0.40       0.5000
    at 0.50       0.5000
    at 0.60       0.5000
    at 0.70       0.5000
    at 0.80       0.5000
    at 0.90       0.5000
    at 1.00       0.5000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.4167
Precision:
  At    5 docs:   0.4000
  At   10 docs:   0.2000
  At   15 docs:   0.1333
  At   20 docs:   0.1000
  At   30 docs:   0.0667
  At  100 docs:   0.0200
  At  200 docs:   0.0100
  At  500 docs:   0.0040
  At 1000 docs:   0.0020
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):          35
Total number of documents over all queries
    Retrieved:          0
    Relevant:           0
    Rel_ret:            0
Interpolated Recall - Precision Averages:
    at 0.00         0.0000
    at 0.10         0.0000
    at 0.20         0.0000
    at 0.30         0.0000
    at 0.40         0.0000
    at 0.50         0.0000
    at 0.60         0.0000
    at 0.70         0.0000
    at 0.80         0.0000
    at 0.90         0.0000
    at 1.00         0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                    0.0000
Precision:
  At    5 docs:     0.0000
  At   10 docs:     0.0000
  At   15 docs:     0.0000
  At   20 docs:     0.0000
  At   30 docs:     0.0000
  At  100 docs:     0.0000
  At  200 docs:     0.0000
  At  500 docs:     0.0000
  At 1000 docs:     0.0000
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:          0.0000
```

```
Queryid (Num):        36
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         3
    Rel_ret:          3
Interpolated Recall - Precision Averages:
    at 0.00       1.0000
    at 0.10       1.0000
    at 0.20       1.0000
    at 0.30       1.0000
    at 0.40       1.0000
    at 0.50       1.0000
    at 0.60       1.0000
    at 0.70       0.7500
    at 0.80       0.7500
    at 0.90       0.7500
    at 1.00       0.7500
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.9167
Precision:
  At    5 docs:   0.6000
  At   10 docs:   0.3000
  At   15 docs:   0.2000
  At   20 docs:   0.1500
  At   30 docs:   0.1000
  At  100 docs:   0.0300
  At  200 docs:   0.0150
  At  500 docs:   0.0060
  At 1000 docs:   0.0030
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.6667
```

```
Queryid (Num):        37
Total number of documents over all queries
    Retrieved:        0
    Relevant:         0
    Rel_ret:          0
Interpolated Recall - Precision Averages:
    at 0.00       0.0000
    at 0.10       0.0000
    at 0.20       0.0000
    at 0.30       0.0000
    at 0.40       0.0000
    at 0.50       0.0000
    at 0.60       0.0000
    at 0.70       0.0000
    at 0.80       0.0000
    at 0.90       0.0000
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0000
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.0000
  At   15 docs:   0.0000
  At   20 docs:   0.0000
  At   30 docs:   0.0000
  At  100 docs:   0.0000
  At  200 docs:   0.0000
  At  500 docs:   0.0000
  At 1000 docs:   0.0000
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):        38
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         8
    Rel_ret:          6
Interpolated Recall - Precision Averages:
    at 0.00       0.5000
    at 0.10       0.5000
    at 0.20       0.0400
    at 0.30       0.0400
    at 0.40       0.0400
    at 0.50       0.0400
    at 0.60       0.0256
    at 0.70       0.0256
    at 0.80       0.0000
    at 0.90       0.0000
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0824
Precision:
  At    5 docs:   0.2000
  At   10 docs:   0.1000
  At   15 docs:   0.0667
  At   20 docs:   0.0500
  At   30 docs:   0.0333
  At  100 docs:   0.0400
  At  200 docs:   0.0200
  At  500 docs:   0.0120
  At 1000 docs:   0.0060
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.1250
```

```
Queryid (Num):        39
Total number of documents over all queries
    Retrieved:        0
    Relevant:         0
    Rel_ret:          0
Interpolated Recall - Precision Averages:
    at 0.00       0.0000
    at 0.10       0.0000
    at 0.20       0.0000
    at 0.30       0.0000
    at 0.40       0.0000
    at 0.50       0.0000
    at 0.60       0.0000
    at 0.70       0.0000
    at 0.80       0.0000
    at 0.90       0.0000
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0000
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.0000
  At   15 docs:   0.0000
  At   20 docs:   0.0000
  At   30 docs:   0.0000
  At  100 docs:   0.0000
  At  200 docs:   0.0000
  At  500 docs:   0.0000
  At 1000 docs:   0.0000
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):         40
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         5
    Rel_ret:          4
Interpolated Recall - Precision Averages:
    at 0.00       0.5000
    at 0.10       0.5000
    at 0.20       0.5000
    at 0.30       0.2222
    at 0.40       0.2222
    at 0.50       0.0064
    at 0.60       0.0064
    at 0.70       0.0052
    at 0.80       0.0052
    at 0.90       0.0000
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.1468
Precision:
  At    5 docs:   0.2000
  At   10 docs:   0.2000
  At   15 docs:   0.1333
  At   20 docs:   0.1000
  At   30 docs:   0.0667
  At  100 docs:   0.0200
  At  200 docs:   0.0100
  At  500 docs:   0.0060
  At 1000 docs:   0.0040
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.2000
```

```
Queryid (Num):          41
Total number of documents over all queries
    Retrieved:          0
    Relevant:           0
    Rel_ret:            0
Interpolated Recall - Precision Averages:
    at 0.00         0.0000
    at 0.10         0.0000
    at 0.20         0.0000
    at 0.30         0.0000
    at 0.40         0.0000
    at 0.50         0.0000
    at 0.60         0.0000
    at 0.70         0.0000
    at 0.80         0.0000
    at 0.90         0.0000
    at 1.00         0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                    0.0000
Precision:
  At    5 docs:     0.0000
  At   10 docs:     0.0000
  At   15 docs:     0.0000
  At   20 docs:     0.0000
  At   30 docs:     0.0000
  At  100 docs:     0.0000
  At  200 docs:     0.0000
  At  500 docs:     0.0000
  At 1000 docs:     0.0000
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:          0.0000
```

```
Queryid (Num):        42
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         3
    Rel_ret:          3
Interpolated Recall - Precision Averages:
    at 0.00       0.0270
    at 0.10       0.0270
    at 0.20       0.0270
    at 0.30       0.0270
    at 0.40       0.0137
    at 0.50       0.0137
    at 0.60       0.0137
    at 0.70       0.0035
    at 0.80       0.0035
    at 0.90       0.0035
    at 1.00       0.0035
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0147
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.0000
  At   15 docs:   0.0000
  At   20 docs:   0.0000
  At   30 docs:   0.0000
  At  100 docs:   0.0100
  At  200 docs:   0.0100
  At  500 docs:   0.0040
  At 1000 docs:   0.0030
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):        43
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         8
    Rel_ret:          8
Interpolated Recall - Precision Averages:
    at 0.00       1.0000
    at 0.10       1.0000
    at 0.20       1.0000
    at 0.30       1.0000
    at 0.40       1.0000
    at 0.50       1.0000
    at 0.60       0.5556
    at 0.70       0.4286
    at 0.80       0.3684
    at 0.90       0.0273
    at 1.00       0.0273
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.6725
Precision:
  At    5 docs:   0.8000
  At   10 docs:   0.5000
  At   15 docs:   0.4000
  At   20 docs:   0.3500
  At   30 docs:   0.2333
  At  100 docs:   0.0700
  At  200 docs:   0.0350
  At  500 docs:   0.0160
  At 1000 docs:   0.0080
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.5000
```

```
Queryid (Num):        44
Total number of documents over all queries
    Retrieved:     1000
    Relevant:        26
    Rel_ret:         26
Interpolated Recall - Precision Averages:
    at 0.00       1.0000
    at 0.10       1.0000
    at 0.20       0.5385
    at 0.30       0.5000
    at 0.40       0.5000
    at 0.50       0.1831
    at 0.60       0.1417
    at 0.70       0.1242
    at 0.80       0.1068
    at 0.90       0.0833
    at 1.00       0.0613
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.3603
Precision:
  At    5 docs:   0.8000
  At   10 docs:   0.5000
  At   15 docs:   0.4667
  At   20 docs:   0.4500
  At   30 docs:   0.3667
  At  100 docs:   0.1400
  At  200 docs:   0.1000
  At  500 docs:   0.0520
  At 1000 docs:   0.0260
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.4231
```

```
Queryid (Num):        45
Total number of documents over all queries
    Retrieved:      284
    Relevant:        26
    Rel_ret:         10
Interpolated Recall - Precision Averages:
    at 0.00       1.0000
    at 0.10       0.2727
    at 0.20       0.0364
    at 0.30       0.0364
    at 0.40       0.0000
    at 0.50       0.0000
    at 0.60       0.0000
    at 0.70       0.0000
    at 0.80       0.0000
    at 0.90       0.0000
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0982
Precision:
  At    5 docs:   0.4000
  At   10 docs:   0.2000
  At   15 docs:   0.2667
  At   20 docs:   0.2000
  At   30 docs:   0.1667
  At  100 docs:   0.0500
  At  200 docs:   0.0300
  At  500 docs:   0.0200
  At 1000 docs:   0.0100
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.1538
```

```
Queryid (Num):        46
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         2
    Rel_ret:          2
Interpolated Recall - Precision Averages:
    at 0.00       0.1667
    at 0.10       0.1667
    at 0.20       0.1667
    at 0.30       0.1667
    at 0.40       0.1667
    at 0.50       0.1667
    at 0.60       0.0833
    at 0.70       0.0833
    at 0.80       0.0833
    at 0.90       0.0833
    at 1.00       0.0833
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.1250
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.1000
  At   15 docs:   0.0667
  At   20 docs:   0.0500
  At   30 docs:   0.0667
  At  100 docs:   0.0200
  At  200 docs:   0.0100
  At  500 docs:   0.0040
  At 1000 docs:   0.0020
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):          47
Total number of documents over all queries
    Retrieved:      1000
    Relevant:          3
    Rel_ret:           3
Interpolated Recall - Precision Averages:
    at 0.00        0.1111
    at 0.10        0.1111
    at 0.20        0.1111
    at 0.30        0.1111
    at 0.40        0.1034
    at 0.50        0.1034
    at 0.60        0.1034
    at 0.70        0.1034
    at 0.80        0.1034
    at 0.90        0.1034
    at 1.00        0.1034
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                   0.1049
Precision:
  At    5 docs:    0.0000
  At   10 docs:    0.1000
  At   15 docs:    0.0667
  At   20 docs:    0.1000
  At   30 docs:    0.1000
  At  100 docs:    0.0300
  At  200 docs:    0.0150
  At  500 docs:    0.0060
  At 1000 docs:    0.0030
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:         0.0000
```

```
Queryid (Num):          48
Total number of documents over all queries
    Retrieved:          0
    Relevant:           0
    Rel_ret:            0
Interpolated Recall - Precision Averages:
    at 0.00         0.0000
    at 0.10         0.0000
    at 0.20         0.0000
    at 0.30         0.0000
    at 0.40         0.0000
    at 0.50         0.0000
    at 0.60         0.0000
    at 0.70         0.0000
    at 0.80         0.0000
    at 0.90         0.0000
    at 1.00         0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                    0.0000
Precision:
  At    5 docs:     0.0000
  At   10 docs:     0.0000
  At   15 docs:     0.0000
  At   20 docs:     0.0000
  At   30 docs:     0.0000
  At  100 docs:     0.0000
  At  200 docs:     0.0000
  At  500 docs:     0.0000
  At 1000 docs:     0.0000
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:          0.0000
```

```
Queryid (Num):        49
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         2
    Rel_ret:          2
Interpolated Recall - Precision Averages:
    at 0.00       0.5000
    at 0.10       0.5000
    at 0.20       0.5000
    at 0.30       0.5000
    at 0.40       0.5000
    at 0.50       0.5000
    at 0.60       0.0163
    at 0.70       0.0163
    at 0.80       0.0163
    at 0.90       0.0163
    at 1.00       0.0163
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.2581
Precision:
  At    5 docs:   0.2000
  At   10 docs:   0.1000
  At   15 docs:   0.0667
  At   20 docs:   0.0500
  At   30 docs:   0.0333
  At  100 docs:   0.0100
  At  200 docs:   0.0100
  At  500 docs:   0.0040
  At 1000 docs:   0.0020
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.5000
```

```
Queryid (Num):          50
Total number of documents over all queries
    Retrieved:          0
    Relevant:           0
    Rel_ret:            0
Interpolated Recall - Precision Averages:
    at 0.00         0.0000
    at 0.10         0.0000
    at 0.20         0.0000
    at 0.30         0.0000
    at 0.40         0.0000
    at 0.50         0.0000
    at 0.60         0.0000
    at 0.70         0.0000
    at 0.80         0.0000
    at 0.90         0.0000
    at 1.00         0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                    0.0000
Precision:
  At    5 docs:     0.0000
  At   10 docs:     0.0000
  At   15 docs:     0.0000
  At   20 docs:     0.0000
  At   30 docs:     0.0000
  At  100 docs:     0.0000
  At  200 docs:     0.0000
  At  500 docs:     0.0000
  At 1000 docs:     0.0000
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:          0.0000
```

Indegree Ranking Experiment using Normalised indegree scores incorporating the Scarcity-Abundance technique (link4)

Combining Linkage and Content evidence using the scarcity-abundance technique for regulating linkage influence

```
Queryid (Num):        1
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         7
    Rel_ret:          4
Interpolated Recall - Precision Averages:
    at 0.00       0.5000
    at 0.10       0.5000
    at 0.20       0.5000
    at 0.30       0.1765
    at 0.40       0.1765
    at 0.50       0.0059
    at 0.60       0.0000
    at 0.70       0.0000
    at 0.80       0.0000
    at 0.90       0.0000
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.1689
Precision:
  At    5 docs:   0.4000
  At   10 docs:   0.2000
  At   15 docs:   0.1333
  At   20 docs:   0.1500
  At   30 docs:   0.1000
  At  100 docs:   0.0300
  At  200 docs:   0.0150
  At  500 docs:   0.0060
  At 1000 docs:   0.0040
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.2857
```

```
Queryid (Num):        2
Total number of documents over all queries
    Retrieved:     1000
    Relevant:        16
    Rel_ret:         16
Interpolated Recall - Precision Averages:
    at 0.00       0.1667
    at 0.10       0.1667
    at 0.20       0.1667
    at 0.30       0.1220
    at 0.40       0.0333
    at 0.50       0.0274
    at 0.60       0.0257
    at 0.70       0.0168
    at 0.80       0.0168
    at 0.90       0.0168
    at 1.00       0.0168
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0604
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.0000
  At   15 docs:   0.1333
  At   20 docs:   0.1000
  At   30 docs:   0.1333
  At  100 docs:   0.0600
  At  200 docs:   0.0300
  At  500 docs:   0.0200
  At 1000 docs:   0.0160
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.1250
```

```
Queryid (Num):        3
Total number of documents over all queries
    Retrieved:     1000
    Relevant:        16
    Rel_ret:         15
Interpolated Recall - Precision Averages:
    at 0.00       1.0000
    at 0.10       1.0000
    at 0.20       1.0000
    at 0.30       0.7143
    at 0.40       0.6000
    at 0.50       0.6000
    at 0.60       0.5217
    at 0.70       0.5217
    at 0.80       0.3171
    at 0.90       0.1765
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.5829
Precision:
  At    5 docs:   0.8000
  At   10 docs:   0.6000
  At   15 docs:   0.6000
  At   20 docs:   0.5000
  At   30 docs:   0.4000
  At  100 docs:   0.1500
  At  200 docs:   0.0750
  At  500 docs:   0.0300
  At 1000 docs:   0.0150
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.5625
```

```
Queryid (Num):        4
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         8
    Rel_ret:          8
Interpolated Recall - Precision Averages:
    at 0.00       1.0000
    at 0.10       1.0000
    at 0.20       0.5000
    at 0.30       0.5000
    at 0.40       0.4444
    at 0.50       0.4444
    at 0.60       0.2273
    at 0.70       0.1429
    at 0.80       0.1296
    at 0.90       0.1212
    at 1.00       0.1212
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.3707
Precision:
  At    5 docs:   0.4000
  At   10 docs:   0.4000
  At   15 docs:   0.2667
  At   20 docs:   0.2000
  At   30 docs:   0.1667
  At  100 docs:   0.0800
  At  200 docs:   0.0400
  At  500 docs:   0.0160
  At 1000 docs:   0.0080
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.3750
```

```
Queryid (Num):        5
Total number of documents over all queries
    Retrieved:        0
    Relevant:         0
    Rel_ret:          0
Interpolated Recall - Precision Averages:
    at 0.00       0.0000
    at 0.10       0.0000
    at 0.20       0.0000
    at 0.30       0.0000
    at 0.40       0.0000
    at 0.50       0.0000
    at 0.60       0.0000
    at 0.70       0.0000
    at 0.80       0.0000
    at 0.90       0.0000
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0000
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.0000
  At   15 docs:   0.0000
  At   20 docs:   0.0000
  At   30 docs:   0.0000
  At  100 docs:   0.0000
  At  200 docs:   0.0000
  At  500 docs:   0.0000
  At 1000 docs:   0.0000
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):        6
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         2
    Rel_ret:          2
Interpolated Recall - Precision Averages:
    at 0.00       0.0588
    at 0.10       0.0588
    at 0.20       0.0588
    at 0.30       0.0588
    at 0.40       0.0588
    at 0.50       0.0588
    at 0.60       0.0094
    at 0.70       0.0094
    at 0.80       0.0094
    at 0.90       0.0094
    at 1.00       0.0094
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0341
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.0000
  At   15 docs:   0.0000
  At   20 docs:   0.0500
  At   30 docs:   0.0333
  At  100 docs:   0.0100
  At  200 docs:   0.0050
  At  500 docs:   0.0040
  At 1000 docs:   0.0020
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):        7
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         6
    Rel_ret:          6
Interpolated Recall - Precision Averages:
    at 0.00       0.1000
    at 0.10       0.1000
    at 0.20       0.0645
    at 0.30       0.0645
    at 0.40       0.0645
    at 0.50       0.0645
    at 0.60       0.0645
    at 0.70       0.0442
    at 0.80       0.0442
    at 0.90       0.0335
    at 1.00       0.0335
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0545
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.1000
  At   15 docs:   0.0667
  At   20 docs:   0.0500
  At   30 docs:   0.0333
  At  100 docs:   0.0400
  At  200 docs:   0.0300
  At  500 docs:   0.0120
  At 1000 docs:   0.0060
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):          8
Total number of documents over all queries
    Retrieved:          0
    Relevant:           0
    Rel_ret:            0
Interpolated Recall - Precision Averages:
    at 0.00         0.0000
    at 0.10         0.0000
    at 0.20         0.0000
    at 0.30         0.0000
    at 0.40         0.0000
    at 0.50         0.0000
    at 0.60         0.0000
    at 0.70         0.0000
    at 0.80         0.0000
    at 0.90         0.0000
    at 1.00         0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                    0.0000
Precision:
  At    5 docs:     0.0000
  At   10 docs:     0.0000
  At   15 docs:     0.0000
  At   20 docs:     0.0000
  At   30 docs:     0.0000
  At  100 docs:     0.0000
  At  200 docs:     0.0000
  At  500 docs:     0.0000
  At 1000 docs:     0.0000
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:          0.0000
```

```
Queryid (Num):        9
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         1
    Rel_ret:          0
Interpolated Recall - Precision Averages:
    at 0.00       0.0000
    at 0.10       0.0000
    at 0.20       0.0000
    at 0.30       0.0000
    at 0.40       0.0000
    at 0.50       0.0000
    at 0.60       0.0000
    at 0.70       0.0000
    at 0.80       0.0000
    at 0.90       0.0000
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0000
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.0000
  At   15 docs:   0.0000
  At   20 docs:   0.0000
  At   30 docs:   0.0000
  At  100 docs:   0.0000
  At  200 docs:   0.0000
  At  500 docs:   0.0000
  At 1000 docs:   0.0000
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):        10
Total number of documents over all queries
    Retrieved:        0
    Relevant:         0
    Rel_ret:          0
Interpolated Recall - Precision Averages:
    at 0.00       0.0000
    at 0.10       0.0000
    at 0.20       0.0000
    at 0.30       0.0000
    at 0.40       0.0000
    at 0.50       0.0000
    at 0.60       0.0000
    at 0.70       0.0000
    at 0.80       0.0000
    at 0.90       0.0000
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0000
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.0000
  At   15 docs:   0.0000
  At   20 docs:   0.0000
  At   30 docs:   0.0000
  At  100 docs:   0.0000
  At  200 docs:   0.0000
  At  500 docs:   0.0000
  At 1000 docs:   0.0000
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):          11
Total number of documents over all queries
    Retrieved:        816
    Relevant:           1
    Rel_ret:            1
Interpolated Recall - Precision Averages:
    at 0.00         0.0227
    at 0.10         0.0227
    at 0.20         0.0227
    at 0.30         0.0227
    at 0.40         0.0227
    at 0.50         0.0227
    at 0.60         0.0227
    at 0.70         0.0227
    at 0.80         0.0227
    at 0.90         0.0227
    at 1.00         0.0227
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                    0.0227
Precision:
  At    5 docs:     0.0000
  At   10 docs:     0.0000
  At   15 docs:     0.0000
  At   20 docs:     0.0000
  At   30 docs:     0.0000
  At  100 docs:     0.0100
  At  200 docs:     0.0050
  At  500 docs:     0.0020
  At 1000 docs:     0.0010
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:          0.0000
```

```
Queryid (Num):        12
Total number of documents over all queries
    Retrieved:     1000
    Relevant:        11
    Rel_ret:         11
Interpolated Recall - Precision Averages:
    at 0.00       0.2500
    at 0.10       0.2500
    at 0.20       0.2500
    at 0.30       0.2500
    at 0.40       0.1613
    at 0.50       0.1429
    at 0.60       0.0762
    at 0.70       0.0762
    at 0.80       0.0474
    at 0.90       0.0377
    at 1.00       0.0377
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.1378
Precision:
  At    5 docs:   0.2000
  At   10 docs:   0.2000
  At   15 docs:   0.2000
  At   20 docs:   0.2000
  At   30 docs:   0.1333
  At  100 docs:   0.0700
  At  200 docs:   0.0450
  At  500 docs:   0.0220
  At 1000 docs:   0.0110
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.1818
```

```
Queryid (Num):        13
Total number of documents over all queries
    Retrieved:     1000
    Relevant:        16
    Rel_ret:         15
Interpolated Recall - Precision Averages:
    at 0.00       1.0000
    at 0.10       0.6667
    at 0.20       0.3684
    at 0.30       0.3684
    at 0.40       0.3684
    at 0.50       0.3448
    at 0.60       0.3448
    at 0.70       0.0960
    at 0.80       0.0903
    at 0.90       0.0396
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.2993
Precision:
  At    5 docs:   0.4000
  At   10 docs:   0.2000
  At   15 docs:   0.3333
  At   20 docs:   0.3500
  At   30 docs:   0.3333
  At  100 docs:   0.1100
  At  200 docs:   0.0700
  At  500 docs:   0.0300
  At 1000 docs:   0.0150
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.3125
```

```
Queryid (Num):        14
Total number of documents over all queries
    Retrieved:        0
    Relevant:         0
    Rel_ret:          0
Interpolated Recall - Precision Averages:
    at 0.00       0.0000
    at 0.10       0.0000
    at 0.20       0.0000
    at 0.30       0.0000
    at 0.40       0.0000
    at 0.50       0.0000
    at 0.60       0.0000
    at 0.70       0.0000
    at 0.80       0.0000
    at 0.90       0.0000
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0000
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.0000
  At   15 docs:   0.0000
  At   20 docs:   0.0000
  At   30 docs:   0.0000
  At  100 docs:   0.0000
  At  200 docs:   0.0000
  At  500 docs:   0.0000
  At 1000 docs:   0.0000
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):        15
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         2
    Rel_ret:          2
Interpolated Recall - Precision Averages:
    at 0.00       0.0556
    at 0.10       0.0556
    at 0.20       0.0556
    at 0.30       0.0556
    at 0.40       0.0556
    at 0.50       0.0556
    at 0.60       0.0556
    at 0.70       0.0556
    at 0.80       0.0556
    at 0.90       0.0556
    at 1.00       0.0556
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0486
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.0000
  At   15 docs:   0.0000
  At   20 docs:   0.0000
  At   30 docs:   0.0333
  At  100 docs:   0.0200
  At  200 docs:   0.0100
  At  500 docs:   0.0040
  At 1000 docs:   0.0020
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):        16
Total number of documents over all queries
    Retrieved:        0
    Relevant:         0
    Rel_ret:          0
Interpolated Recall - Precision Averages:
    at 0.00       0.0000
    at 0.10       0.0000
    at 0.20       0.0000
    at 0.30       0.0000
    at 0.40       0.0000
    at 0.50       0.0000
    at 0.60       0.0000
    at 0.70       0.0000
    at 0.80       0.0000
    at 0.90       0.0000
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0000
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.0000
  At   15 docs:   0.0000
  At   20 docs:   0.0000
  At   30 docs:   0.0000
  At  100 docs:   0.0000
  At  200 docs:   0.0000
  At  500 docs:   0.0000
  At 1000 docs:   0.0000
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):        17
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         8
    Rel_ret:          8
Interpolated Recall - Precision Averages:
    at 0.00       0.3333
    at 0.10       0.3333
    at 0.20       0.3333
    at 0.30       0.3333
    at 0.40       0.3125
    at 0.50       0.3125
    at 0.60       0.3125
    at 0.70       0.1296
    at 0.80       0.1296
    at 0.90       0.1053
    at 1.00       0.1053
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.2091
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.3000
  At   15 docs:   0.2667
  At   20 docs:   0.2500
  At   30 docs:   0.1667
  At  100 docs:   0.0800
  At  200 docs:   0.0400
  At  500 docs:   0.0160
  At 1000 docs:   0.0080
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.2500
```

```
Queryid (Num):        18
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         5
    Rel_ret:          5
Interpolated Recall - Precision Averages:
    at 0.00       0.5000
    at 0.10       0.5000
    at 0.20       0.5000
    at 0.30       0.5000
    at 0.40       0.5000
    at 0.50       0.2000
    at 0.60       0.2000
    at 0.70       0.1905
    at 0.80       0.1905
    at 0.90       0.0442
    at 1.00       0.0442
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.2869
Precision:
  At    5 docs:   0.4000
  At   10 docs:   0.2000
  At   15 docs:   0.2000
  At   20 docs:   0.1500
  At   30 docs:   0.1333
  At  100 docs:   0.0400
  At  200 docs:   0.0250
  At  500 docs:   0.0100
  At 1000 docs:   0.0050
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.4000
```

```
Queryid (Num):        19
Total number of documents over all queries
    Retrieved:       96
    Relevant:         5
    Rel_ret:          5
Interpolated Recall - Precision Averages:
    at 0.00       1.0000
    at 0.10       1.0000
    at 0.20       1.0000
    at 0.30       0.3750
    at 0.40       0.3750
    at 0.50       0.3750
    at 0.60       0.3750
    at 0.70       0.2500
    at 0.80       0.2500
    at 0.90       0.2500
    at 1.00       0.2500
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.4242
Precision:
  At    5 docs:   0.2000
  At   10 docs:   0.3000
  At   15 docs:   0.2000
  At   20 docs:   0.2500
  At   30 docs:   0.1667
  At  100 docs:   0.0500
  At  200 docs:   0.0250
  At  500 docs:   0.0100
  At 1000 docs:   0.0050
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.2000
```

```
Queryid (Num):          20
Total number of documents over all queries
    Retrieved:      1000
    Relevant:          2
    Rel_ret:           2
Interpolated Recall - Precision Averages:
    at 0.00        0.0833
    at 0.10        0.0833
    at 0.20        0.0833
    at 0.30        0.0833
    at 0.40        0.0833
    at 0.50        0.0833
    at 0.60        0.0392
    at 0.70        0.0392
    at 0.80        0.0392
    at 0.90        0.0392
    at 1.00        0.0392
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                   0.0613
Precision:
  At    5 docs:    0.0000
  At   10 docs:    0.0000
  At   15 docs:    0.0667
  At   20 docs:    0.0500
  At   30 docs:    0.0333
  At  100 docs:    0.0200
  At  200 docs:    0.0100
  At  500 docs:    0.0040
  At 1000 docs:    0.0020
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:         0.0000
```

```
Queryid (Num):        21
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         1
    Rel_ret:          1
Interpolated Recall - Precision Averages:
    at 0.00       1.0000
    at 0.10       1.0000
    at 0.20       1.0000
    at 0.30       1.0000
    at 0.40       1.0000
    at 0.50       1.0000
    at 0.60       1.0000
    at 0.70       1.0000
    at 0.80       1.0000
    at 0.90       1.0000
    at 1.00       1.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  1.0000
Precision:
  At    5 docs:   0.2000
  At   10 docs:   0.1000
  At   15 docs:   0.0667
  At   20 docs:   0.0500
  At   30 docs:   0.0333
  At  100 docs:   0.0100
  At  200 docs:   0.0050
  At  500 docs:   0.0020
  At 1000 docs:   0.0010
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        1.0000
```

```
Queryid (Num):        22
Total number of documents over all queries
    Retrieved:     1000
    Relevant:        11
    Rel_ret:          9
Interpolated Recall - Precision Averages:
    at 0.00       0.1579
    at 0.10       0.1579
    at 0.20       0.1579
    at 0.30       0.0952
    at 0.40       0.0725
    at 0.50       0.0706
    at 0.60       0.0366
    at 0.70       0.0289
    at 0.80       0.0289
    at 0.90       0.0000
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0664
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.1000
  At   15 docs:   0.0667
  At   20 docs:   0.1500
  At   30 docs:   0.1000
  At  100 docs:   0.0600
  At  200 docs:   0.0350
  At  500 docs:   0.0180
  At 1000 docs:   0.0090
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0909
```

```
Queryid (Num):        23
Total number of documents over all queries
    Retrieved:        0
    Relevant:         0
    Rel_ret:          0
Interpolated Recall - Precision Averages:
    at 0.00       0.0000
    at 0.10       0.0000
    at 0.20       0.0000
    at 0.30       0.0000
    at 0.40       0.0000
    at 0.50       0.0000
    at 0.60       0.0000
    at 0.70       0.0000
    at 0.80       0.0000
    at 0.90       0.0000
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0000
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.0000
  At   15 docs:   0.0000
  At   20 docs:   0.0000
  At   30 docs:   0.0000
  At  100 docs:   0.0000
  At  200 docs:   0.0000
  At  500 docs:   0.0000
  At 1000 docs:   0.0000
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):        24
Total number of documents over all queries
    Retrieved:     1000
    Relevant:        12
    Rel_ret:          4
Interpolated Recall - Precision Averages:
    at 0.00       0.0833
    at 0.10       0.0333
    at 0.20       0.0080
    at 0.30       0.0080
    at 0.40       0.0000
    at 0.50       0.0000
    at 0.60       0.0000
    at 0.70       0.0000
    at 0.80       0.0000
    at 0.90       0.0000
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0110
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.0000
  At   15 docs:   0.0667
  At   20 docs:   0.0500
  At   30 docs:   0.0333
  At  100 docs:   0.0200
  At  200 docs:   0.0100
  At  500 docs:   0.0080
  At 1000 docs:   0.0040
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0833
```

```
Queryid (Num):        25
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         2
    Rel_ret:          2
Interpolated Recall - Precision Averages:
    at 0.00       0.0870
    at 0.10       0.0870
    at 0.20       0.0870
    at 0.30       0.0870
    at 0.40       0.0870
    at 0.50       0.0870
    at 0.60       0.0870
    at 0.70       0.0870
    at 0.80       0.0870
    at 0.90       0.0870
    at 1.00       0.0870
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0768
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.0000
  At   15 docs:   0.0667
  At   20 docs:   0.0500
  At   30 docs:   0.0667
  At  100 docs:   0.0200
  At  200 docs:   0.0100
  At  500 docs:   0.0040
  At 1000 docs:   0.0020
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):        26
Total number of documents over all queries
    Retrieved:     1000
    Relevant:        19
    Rel_ret:         19
Interpolated Recall - Precision Averages:
    at 0.00       0.6667
    at 0.10       0.6667
    at 0.20       0.6316
    at 0.30       0.6316
    at 0.40       0.6316
    at 0.50       0.6316
    at 0.60       0.6316
    at 0.70       0.3721
    at 0.80       0.3721
    at 0.90       0.0293
    at 1.00       0.0293
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.4508
Precision:
  At    5 docs:   0.6000
  At   10 docs:   0.5000
  At   15 docs:   0.6000
  At   20 docs:   0.6000
  At   30 docs:   0.4000
  At  100 docs:   0.1600
  At  200 docs:   0.0850
  At  500 docs:   0.0340
  At 1000 docs:   0.0190
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.6316
```

```
Queryid (Num):        27
Total number of documents over all queries
    Retrieved:        0
    Relevant:         0
    Rel_ret:          0
Interpolated Recall - Precision Averages:
    at 0.00       0.0000
    at 0.10       0.0000
    at 0.20       0.0000
    at 0.30       0.0000
    at 0.40       0.0000
    at 0.50       0.0000
    at 0.60       0.0000
    at 0.70       0.0000
    at 0.80       0.0000
    at 0.90       0.0000
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0000
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.0000
  At   15 docs:   0.0000
  At   20 docs:   0.0000
  At   30 docs:   0.0000
  At  100 docs:   0.0000
  At  200 docs:   0.0000
  At  500 docs:   0.0000
  At 1000 docs:   0.0000
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):        28
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         2
    Rel_ret:          2
Interpolated Recall - Precision Averages:
    at 0.00       0.5000
    at 0.10       0.5000
    at 0.20       0.5000
    at 0.30       0.5000
    at 0.40       0.5000
    at 0.50       0.5000
    at 0.60       0.0028
    at 0.70       0.0028
    at 0.80       0.0028
    at 0.90       0.0028
    at 1.00       0.0028
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.2514
Precision:
  At    5 docs:   0.2000
  At   10 docs:   0.1000
  At   15 docs:   0.0667
  At   20 docs:   0.0500
  At   30 docs:   0.0333
  At  100 docs:   0.0100
  At  200 docs:   0.0050
  At  500 docs:   0.0020
  At 1000 docs:   0.0020
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.5000
```

```
Queryid (Num):        29
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         5
    Rel_ret:          4
Interpolated Recall - Precision Averages:
    at 0.00       1.0000
    at 0.10       1.0000
    at 0.20       1.0000
    at 0.30       0.7500
    at 0.40       0.7500
    at 0.50       0.7500
    at 0.60       0.7500
    at 0.70       0.1905
    at 0.80       0.1905
    at 0.90       0.0000
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.5214
Precision:
  At    5 docs:   0.6000
  At   10 docs:   0.3000
  At   15 docs:   0.2000
  At   20 docs:   0.1500
  At   30 docs:   0.1333
  At  100 docs:   0.0400
  At  200 docs:   0.0200
  At  500 docs:   0.0080
  At 1000 docs:   0.0040
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.6000
```

```
Queryid (Num):          30
Total number of documents over all queries
    Retrieved:          0
    Relevant:           0
    Rel_ret:            0
Interpolated Recall - Precision Averages:
    at 0.00         0.0000
    at 0.10         0.0000
    at 0.20         0.0000
    at 0.30         0.0000
    at 0.40         0.0000
    at 0.50         0.0000
    at 0.60         0.0000
    at 0.70         0.0000
    at 0.80         0.0000
    at 0.90         0.0000
    at 1.00         0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                    0.0000
Precision:
  At    5 docs:     0.0000
  At   10 docs:     0.0000
  At   15 docs:     0.0000
  At   20 docs:     0.0000
  At   30 docs:     0.0000
  At  100 docs:     0.0000
  At  200 docs:     0.0000
  At  500 docs:     0.0000
  At 1000 docs:     0.0000
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:          0.0000
```

```
Queryid (Num):        31
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         3
    Rel_ret:          3
Interpolated Recall - Precision Averages:
    at 0.00       0.3000
    at 0.10       0.3000
    at 0.20       0.3000
    at 0.30       0.3000
    at 0.40       0.3000
    at 0.50       0.3000
    at 0.60       0.3000
    at 0.70       0.3000
    at 0.80       0.3000
    at 0.90       0.3000
    at 1.00       0.3000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.2217
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.3000
  At   15 docs:   0.2000
  At   20 docs:   0.1500
  At   30 docs:   0.1000
  At  100 docs:   0.0300
  At  200 docs:   0.0150
  At  500 docs:   0.0060
  At 1000 docs:   0.0030
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):        32
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         2
    Rel_ret:          2
Interpolated Recall - Precision Averages:
    at 0.00       1.0000
    at 0.10       1.0000
    at 0.20       1.0000
    at 0.30       1.0000
    at 0.40       1.0000
    at 0.50       1.0000
    at 0.60       0.6667
    at 0.70       0.6667
    at 0.80       0.6667
    at 0.90       0.6667
    at 1.00       0.6667
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.8333
Precision:
  At    5 docs:   0.4000
  At   10 docs:   0.2000
  At   15 docs:   0.1333
  At   20 docs:   0.1000
  At   30 docs:   0.0667
  At  100 docs:   0.0200
  At  200 docs:   0.0100
  At  500 docs:   0.0040
  At 1000 docs:   0.0020
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.5000
```

```
Queryid (Num):        33
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         3
    Rel_ret:          3
Interpolated Recall - Precision Averages:
    at 0.00       1.0000
    at 0.10       1.0000
    at 0.20       1.0000
    at 0.30       1.0000
    at 0.40       1.0000
    at 0.50       1.0000
    at 0.60       1.0000
    at 0.70       0.0070
    at 0.80       0.0070
    at 0.90       0.0070
    at 1.00       0.0070
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.6690
Precision:
  At    5 docs:   0.4000
  At   10 docs:   0.2000
  At   15 docs:   0.1333
  At   20 docs:   0.1000
  At   30 docs:   0.0667
  At  100 docs:   0.0200
  At  200 docs:   0.0100
  At  500 docs:   0.0060
  At 1000 docs:   0.0030
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.6667
```

```
Queryid (Num):        34
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         2
    Rel_ret:          2
Interpolated Recall - Precision Averages:
    at 0.00       0.5000
    at 0.10       0.5000
    at 0.20       0.5000
    at 0.30       0.5000
    at 0.40       0.5000
    at 0.50       0.5000
    at 0.60       0.5000
    at 0.70       0.5000
    at 0.80       0.5000
    at 0.90       0.5000
    at 1.00       0.5000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.4167
Precision:
  At    5 docs:   0.4000
  At   10 docs:   0.2000
  At   15 docs:   0.1333
  At   20 docs:   0.1000
  At   30 docs:   0.0667
  At  100 docs:   0.0200
  At  200 docs:   0.0100
  At  500 docs:   0.0040
  At 1000 docs:   0.0020
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):          35
Total number of documents over all queries
    Retrieved:          0
    Relevant:           0
    Rel_ret:            0
Interpolated Recall - Precision Averages:
    at 0.00         0.0000
    at 0.10         0.0000
    at 0.20         0.0000
    at 0.30         0.0000
    at 0.40         0.0000
    at 0.50         0.0000
    at 0.60         0.0000
    at 0.70         0.0000
    at 0.80         0.0000
    at 0.90         0.0000
    at 1.00         0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                    0.0000
Precision:
  At    5 docs:     0.0000
  At   10 docs:     0.0000
  At   15 docs:     0.0000
  At   20 docs:     0.0000
  At   30 docs:     0.0000
  At  100 docs:     0.0000
  At  200 docs:     0.0000
  At  500 docs:     0.0000
  At 1000 docs:     0.0000
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:          0.0000
```

```
Queryid (Num):        36
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         3
    Rel_ret:          3
Interpolated Recall - Precision Averages:
    at 0.00       1.0000
    at 0.10       1.0000
    at 0.20       1.0000
    at 0.30       1.0000
    at 0.40       1.0000
    at 0.50       1.0000
    at 0.60       1.0000
    at 0.70       0.7500
    at 0.80       0.7500
    at 0.90       0.7500
    at 1.00       0.7500
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.9167
Precision:
  At    5 docs:   0.6000
  At   10 docs:   0.3000
  At   15 docs:   0.2000
  At   20 docs:   0.1500
  At   30 docs:   0.1000
  At  100 docs:   0.0300
  At  200 docs:   0.0150
  At  500 docs:   0.0060
  At 1000 docs:   0.0030
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.6667
```

```
Queryid (Num):          37
Total number of documents over all queries
    Retrieved:          0
    Relevant:           0
    Rel_ret:            0
Interpolated Recall - Precision Averages:
    at 0.00         0.0000
    at 0.10         0.0000
    at 0.20         0.0000
    at 0.30         0.0000
    at 0.40         0.0000
    at 0.50         0.0000
    at 0.60         0.0000
    at 0.70         0.0000
    at 0.80         0.0000
    at 0.90         0.0000
    at 1.00         0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                    0.0000
Precision:
  At    5 docs:     0.0000
  At   10 docs:     0.0000
  At   15 docs:     0.0000
  At   20 docs:     0.0000
  At   30 docs:     0.0000
  At  100 docs:     0.0000
  At  200 docs:     0.0000
  At  500 docs:     0.0000
  At 1000 docs:     0.0000
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:          0.0000
```

```
Queryid (Num):          38
Total number of documents over all queries
    Retrieved:      1000
    Relevant:          8
    Rel_ret:           6
Interpolated Recall - Precision Averages:
    at 0.00        0.5000
    at 0.10        0.5000
    at 0.20        0.0396
    at 0.30        0.0396
    at 0.40        0.0396
    at 0.50        0.0396
    at 0.60        0.0256
    at 0.70        0.0256
    at 0.80        0.0000
    at 0.90        0.0000
    at 1.00        0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                   0.0823
Precision:
  At    5 docs:    0.2000
  At   10 docs:    0.1000
  At   15 docs:    0.0667
  At   20 docs:    0.0500
  At   30 docs:    0.0333
  At  100 docs:    0.0300
  At  200 docs:    0.0200
  At  500 docs:    0.0120
  At 1000 docs:    0.0060
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:         0.1250
```

```
Queryid (Num):        39
Total number of documents over all queries
    Retrieved:        0
    Relevant:         0
    Rel_ret:          0
Interpolated Recall - Precision Averages:
    at 0.00       0.0000
    at 0.10       0.0000
    at 0.20       0.0000
    at 0.30       0.0000
    at 0.40       0.0000
    at 0.50       0.0000
    at 0.60       0.0000
    at 0.70       0.0000
    at 0.80       0.0000
    at 0.90       0.0000
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0000
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.0000
  At   15 docs:   0.0000
  At   20 docs:   0.0000
  At   30 docs:   0.0000
  At  100 docs:   0.0000
  At  200 docs:   0.0000
  At  500 docs:   0.0000
  At 1000 docs:   0.0000
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):        40
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         5
    Rel_ret:          4
Interpolated Recall - Precision Averages:
    at 0.00       0.5000
    at 0.10       0.5000
    at 0.20       0.5000
    at 0.30       0.2222
    at 0.40       0.2222
    at 0.50       0.0063
    at 0.60       0.0063
    at 0.70       0.0052
    at 0.80       0.0052
    at 0.90       0.0000
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.1467
Precision:
  At    5 docs:   0.2000
  At   10 docs:   0.2000
  At   15 docs:   0.1333
  At   20 docs:   0.1000
  At   30 docs:   0.0667
  At  100 docs:   0.0200
  At  200 docs:   0.0100
  At  500 docs:   0.0060
  At 1000 docs:   0.0040
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.2000
```

```
Queryid (Num):          41
Total number of documents over all queries
    Retrieved:          0
    Relevant:           0
    Rel_ret:            0
Interpolated Recall - Precision Averages:
    at 0.00         0.0000
    at 0.10         0.0000
    at 0.20         0.0000
    at 0.30         0.0000
    at 0.40         0.0000
    at 0.50         0.0000
    at 0.60         0.0000
    at 0.70         0.0000
    at 0.80         0.0000
    at 0.90         0.0000
    at 1.00         0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                    0.0000
Precision:
  At    5 docs:     0.0000
  At   10 docs:     0.0000
  At   15 docs:     0.0000
  At   20 docs:     0.0000
  At   30 docs:     0.0000
  At  100 docs:     0.0000
  At  200 docs:     0.0000
  At  500 docs:     0.0000
  At 1000 docs:     0.0000
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:          0.0000
```

```
Queryid (Num):        42
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         3
    Rel_ret:          3
Interpolated Recall - Precision Averages:
    at 0.00       0.0270
    at 0.10       0.0270
    at 0.20       0.0270
    at 0.30       0.0270
    at 0.40       0.0137
    at 0.50       0.0137
    at 0.60       0.0137
    at 0.70       0.0034
    at 0.80       0.0034
    at 0.90       0.0034
    at 1.00       0.0034
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0147
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.0000
  At   15 docs:   0.0000
  At   20 docs:   0.0000
  At   30 docs:   0.0000
  At  100 docs:   0.0100
  At  200 docs:   0.0100
  At  500 docs:   0.0040
  At 1000 docs:   0.0030
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):        43
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         8
    Rel_ret:          8
Interpolated Recall - Precision Averages:
    at 0.00       1.0000
    at 0.10       1.0000
    at 0.20       1.0000
    at 0.30       1.0000
    at 0.40       1.0000
    at 0.50       1.0000
    at 0.60       0.5556
    at 0.70       0.4286
    at 0.80       0.3684
    at 0.90       0.0273
    at 1.00       0.0273
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.6725
Precision:
  At    5 docs:   0.8000
  At   10 docs:   0.5000
  At   15 docs:   0.4000
  At   20 docs:   0.3500
  At   30 docs:   0.2333
  At  100 docs:   0.0700
  At  200 docs:   0.0350
  At  500 docs:   0.0160
  At 1000 docs:   0.0080
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.5000
```

```
Queryid (Num):        44
Total number of documents over all queries
    Retrieved:     1000
    Relevant:        26
    Rel_ret:         26
Interpolated Recall - Precision Averages:
    at 0.00       1.0000
    at 0.10       1.0000
    at 0.20       0.5385
    at 0.30       0.5000
    at 0.40       0.5000
    at 0.50       0.1831
    at 0.60       0.1417
    at 0.70       0.1242
    at 0.80       0.1063
    at 0.90       0.0833
    at 1.00       0.0613
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.3603
Precision:
  At    5 docs:   0.8000
  At   10 docs:   0.5000
  At   15 docs:   0.4667
  At   20 docs:   0.4500
  At   30 docs:   0.3667
  At  100 docs:   0.1400
  At  200 docs:   0.1000
  At  500 docs:   0.0520
  At 1000 docs:   0.0260
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.4231
```

Queryid (Num): 45
Total number of documents over all queries
Retrieved: 284
Relevant: 26
Rel_ret: 10
Interpolated Recall - Precision Averages:
at 0.00 1.0000
at 0.10 0.3750
at 0.20 0.0366
at 0.30 0.0366
at 0.40 0.0000
at 0.50 0.0000
at 0.60 0.0000
at 0.70 0.0000
at 0.80 0.0000
at 0.90 0.0000
at 1.00 0.0000
Average precision (non-interpolated) for all rel docs(averaged over queries)
0.1012
Precision:
At 5 docs: 0.4000
At 10 docs: 0.3000
At 15 docs: 0.2000
At 20 docs: 0.2000
At 30 docs: 0.1667
At 100 docs: 0.0500
At 200 docs: 0.0300
At 500 docs: 0.0200
At 1000 docs: 0.0100
R-Precision (precision after R (= num_rel for a query) docs retrieved):
Exact: 0.1923

```
Queryid (Num):        46
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         2
    Rel_ret:          2
Interpolated Recall - Precision Averages:
    at 0.00       0.2000
    at 0.10       0.2000
    at 0.20       0.2000
    at 0.30       0.2000
    at 0.40       0.2000
    at 0.50       0.2000
    at 0.60       0.0833
    at 0.70       0.0833
    at 0.80       0.0833
    at 0.90       0.0833
    at 1.00       0.0833
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.1417
Precision:
  At    5 docs:   0.2000
  At   10 docs:   0.1000
  At   15 docs:   0.0667
  At   20 docs:   0.0500
  At   30 docs:   0.0667
  At  100 docs:   0.0200
  At  200 docs:   0.0100
  At  500 docs:   0.0040
  At 1000 docs:   0.0020
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):          47
Total number of documents over all queries
    Retrieved:      1000
    Relevant:          3
    Rel_ret:           3
Interpolated Recall - Precision Averages:
    at 0.00        0.1111
    at 0.10        0.1111
    at 0.20        0.1111
    at 0.30        0.1111
    at 0.40        0.1000
    at 0.50        0.1000
    at 0.60        0.1000
    at 0.70        0.1000
    at 0.80        0.1000
    at 0.90        0.1000
    at 1.00        0.1000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                   0.1037
Precision:
  At    5 docs:    0.0000
  At   10 docs:    0.1000
  At   15 docs:    0.0667
  At   20 docs:    0.1000
  At   30 docs:    0.1000
  At  100 docs:    0.0300
  At  200 docs:    0.0150
  At  500 docs:    0.0060
  At 1000 docs:    0.0030
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:         0.0000
```

```
Queryid (Num):          48
Total number of documents over all queries
    Retrieved:          0
    Relevant:           0
    Rel_ret:            0
Interpolated Recall - Precision Averages:
    at 0.00         0.0000
    at 0.10         0.0000
    at 0.20         0.0000
    at 0.30         0.0000
    at 0.40         0.0000
    at 0.50         0.0000
    at 0.60         0.0000
    at 0.70         0.0000
    at 0.80         0.0000
    at 0.90         0.0000
    at 1.00         0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                    0.0000
Precision:
  At    5 docs:     0.0000
  At   10 docs:     0.0000
  At   15 docs:     0.0000
  At   20 docs:     0.0000
  At   30 docs:     0.0000
  At  100 docs:     0.0000
  At  200 docs:     0.0000
  At  500 docs:     0.0000
  At 1000 docs:     0.0000
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:          0.0000
```

```
Queryid (Num):        49
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         2
    Rel_ret:          2
Interpolated Recall - Precision Averages:
    at 0.00       0.5000
    at 0.10       0.5000
    at 0.20       0.5000
    at 0.30       0.5000
    at 0.40       0.5000
    at 0.50       0.5000
    at 0.60       0.0161
    at 0.70       0.0161
    at 0.80       0.0161
    at 0.90       0.0161
    at 1.00       0.0161
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.2581
Precision:
  At    5 docs:   0.2000
  At   10 docs:   0.1000
  At   15 docs:   0.0667
  At   20 docs:   0.0500
  At   30 docs:   0.0333
  At  100 docs:   0.0100
  At  200 docs:   0.0100
  At  500 docs:   0.0040
  At 1000 docs:   0.0020
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.5000
```

```
Queryid (Num):        50
Total number of documents over all queries
    Retrieved:        0
    Relevant:         0
    Rel_ret:          0
Interpolated Recall - Precision Averages:
    at 0.00       0.0000
    at 0.10       0.0000
    at 0.20       0.0000
    at 0.30       0.0000
    at 0.40       0.0000
    at 0.50       0.0000
    at 0.60       0.0000
    at 0.70       0.0000
    at 0.80       0.0000
    at 0.90       0.0000
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0000
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.0000
  At   15 docs:   0.0000
  At   20 docs:   0.0000
  At   30 docs:   0.0000
  At  100 docs:   0.0000
  At  200 docs:   0.0000
  At  500 docs:   0.0000
  At 1000 docs:   0.0000
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

SiteRank Experiment incorporating the Scarcity-Abundance technique (SiteRank_Query)

Combining Linkage and Content evidence using the scarcity-abundance technique for regulating linkage influence.

```
Queryid (Num):        1
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         7
    Rel_ret:          4
Interpolated Recall - Precision Averages:
    at 0.00       0.5000
    at 0.10       0.5000
    at 0.20       0.5000
    at 0.30       0.1667
    at 0.40       0.1667
    at 0.50       0.0048
    at 0.60       0.0000
    at 0.70       0.0000
    at 0.80       0.0000
    at 0.90       0.0000
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.1673
Precision:
  At    5 docs:   0.4000
  At   10 docs:   0.2000
  At   15 docs:   0.1333
  At   20 docs:   0.1500
  At   30 docs:   0.1000
  At  100 docs:   0.0300
  At  200 docs:   0.0150
  At  500 docs:   0.0060
  At 1000 docs:   0.0040
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.2857
```

```
Queryid (Num):        2
Total number of documents over all queries
    Retrieved:     1000
    Relevant:        16
    Rel_ret:         16
Interpolated Recall - Precision Averages:
    at 0.00       0.1429
    at 0.10       0.1429
    at 0.20       0.1379
    at 0.30       0.1091
    at 0.40       0.0402
    at 0.50       0.0332
    at 0.60       0.0251
    at 0.70       0.0169
    at 0.80       0.0169
    at 0.90       0.0169
    at 1.00       0.0166
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0574
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.0000
  At   15 docs:   0.1333
  At   20 docs:   0.1000
  At   30 docs:   0.1333
  At  100 docs:   0.0600
  At  200 docs:   0.0350
  At  500 docs:   0.0200
  At 1000 docs:   0.0160
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.1250
```

```
Queryid (Num):        3
Total number of documents over all queries
    Retrieved:     1000
    Relevant:        16
    Rel_ret:         15
Interpolated Recall - Precision Averages:
    at 0.00       1.0000
    at 0.10       0.8333
    at 0.20       0.8333
    at 0.30       0.8333
    at 0.40       0.6923
    at 0.50       0.6923
    at 0.60       0.6667
    at 0.70       0.4800
    at 0.80       0.3611
    at 0.90       0.1500
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.5838
Precision:
  At    5 docs:   0.8000
  At   10 docs:   0.6000
  At   15 docs:   0.6667
  At   20 docs:   0.5500
  At   30 docs:   0.4000
  At  100 docs:   0.1500
  At  200 docs:   0.0750
  At  500 docs:   0.0300
  At 1000 docs:   0.0150
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.6250
```

```
Queryid (Num):        4
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         8
    Rel_ret:          8
Interpolated Recall - Precision Averages:
    at 0.00       0.6000
    at 0.10       0.6000
    at 0.20       0.6000
    at 0.30       0.6000
    at 0.40       0.4444
    at 0.50       0.4444
    at 0.60       0.2000
    at 0.70       0.1538
    at 0.80       0.1333
    at 0.90       0.1333
    at 1.00       0.1333
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.3330
Precision:
  At    5 docs:   0.6000
  At   10 docs:   0.4000
  At   15 docs:   0.2667
  At   20 docs:   0.2000
  At   30 docs:   0.1667
  At  100 docs:   0.0800
  At  200 docs:   0.0400
  At  500 docs:   0.0160
  At 1000 docs:   0.0080
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.3750
```

```
Queryid (Num):          5
Total number of documents over all queries
    Retrieved:          0
    Relevant:           0
    Rel_ret:            0
Interpolated Recall - Precision Averages:
    at 0.00         0.0000
    at 0.10         0.0000
    at 0.20         0.0000
    at 0.30         0.0000
    at 0.40         0.0000
    at 0.50         0.0000
    at 0.60         0.0000
    at 0.70         0.0000
    at 0.80         0.0000
    at 0.90         0.0000
    at 1.00         0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                    0.0000
Precision:
  At    5 docs:     0.0000
  At   10 docs:     0.0000
  At   15 docs:     0.0000
  At   20 docs:     0.0000
  At   30 docs:     0.0000
  At  100 docs:     0.0000
  At  200 docs:     0.0000
  At  500 docs:     0.0000
  At 1000 docs:     0.0000
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:          0.0000
```

```
Queryid (Num):        6
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         2
    Rel_ret:          2
Interpolated Recall - Precision Averages:
    at 0.00       0.0476
    at 0.10       0.0476
    at 0.20       0.0476
    at 0.30       0.0476
    at 0.40       0.0476
    at 0.50       0.0476
    at 0.60       0.0073
    at 0.70       0.0073
    at 0.80       0.0073
    at 0.90       0.0073
    at 1.00       0.0073
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0274
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.0000
  At   15 docs:   0.0000
  At   20 docs:   0.0000
  At   30 docs:   0.0333
  At  100 docs:   0.0100
  At  200 docs:   0.0050
  At  500 docs:   0.0040
  At 1000 docs:   0.0020
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):        7
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         6
    Rel_ret:          6
Interpolated Recall - Precision Averages:
    at 0.00       0.1429
    at 0.10       0.1429
    at 0.20       0.0625
    at 0.30       0.0625
    at 0.40       0.0625
    at 0.50       0.0625
    at 0.60       0.0625
    at 0.70       0.0459
    at 0.80       0.0459
    at 0.90       0.0282
    at 1.00       0.0282
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0611
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.1000
  At   15 docs:   0.0667
  At   20 docs:   0.0500
  At   30 docs:   0.0333
  At  100 docs:   0.0400
  At  200 docs:   0.0250
  At  500 docs:   0.0120
  At 1000 docs:   0.0060
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):        8
Total number of documents over all queries
    Retrieved:        0
    Relevant:         0
    Rel_ret:          0
Interpolated Recall - Precision Averages:
    at 0.00       0.0000
    at 0.10       0.0000
    at 0.20       0.0000
    at 0.30       0.0000
    at 0.40       0.0000
    at 0.50       0.0000
    at 0.60       0.0000
    at 0.70       0.0000
    at 0.80       0.0000
    at 0.90       0.0000
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0000
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.0000
  At   15 docs:   0.0000
  At   20 docs:   0.0000
  At   30 docs:   0.0000
  At  100 docs:   0.0000
  At  200 docs:   0.0000
  At  500 docs:   0.0000
  At 1000 docs:   0.0000
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):        9
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         1
    Rel_ret:          0
Interpolated Recall - Precision Averages:
    at 0.00       0.0000
    at 0.10       0.0000
    at 0.20       0.0000
    at 0.30       0.0000
    at 0.40       0.0000
    at 0.50       0.0000
    at 0.60       0.0000
    at 0.70       0.0000
    at 0.80       0.0000
    at 0.90       0.0000
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0000
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.0000
  At   15 docs:   0.0000
  At   20 docs:   0.0000
  At   30 docs:   0.0000
  At  100 docs:   0.0000
  At  200 docs:   0.0000
  At  500 docs:   0.0000
  At 1000 docs:   0.0000
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):        10
Total number of documents over all queries
    Retrieved:        0
    Relevant:         0
    Rel_ret:          0
Interpolated Recall - Precision Averages:
    at 0.00       0.0000
    at 0.10       0.0000
    at 0.20       0.0000
    at 0.30       0.0000
    at 0.40       0.0000
    at 0.50       0.0000
    at 0.60       0.0000
    at 0.70       0.0000
    at 0.80       0.0000
    at 0.90       0.0000
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0000
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.0000
  At   15 docs:   0.0000
  At   20 docs:   0.0000
  At   30 docs:   0.0000
  At  100 docs:   0.0000
  At  200 docs:   0.0000
  At  500 docs:   0.0000
  At 1000 docs:   0.0000
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):        11
Total number of documents over all queries
    Retrieved:      816
    Relevant:         1
    Rel_ret:          1
Interpolated Recall - Precision Averages:
    at 0.00       0.0256
    at 0.10       0.0256
    at 0.20       0.0256
    at 0.30       0.0256
    at 0.40       0.0256
    at 0.50       0.0256
    at 0.60       0.0256
    at 0.70       0.0256
    at 0.80       0.0256
    at 0.90       0.0256
    at 1.00       0.0256
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0256
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.0000
  At   15 docs:   0.0000
  At   20 docs:   0.0000
  At   30 docs:   0.0000
  At  100 docs:   0.0100
  At  200 docs:   0.0050
  At  500 docs:   0.0020
  At 1000 docs:   0.0010
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):        12
Total number of documents over all queries
    Retrieved:     1000
    Relevant:        11
    Rel_ret:         11
Interpolated Recall - Precision Averages:
    at 0.00       0.3333
    at 0.10       0.2857
    at 0.20       0.2500
    at 0.30       0.2500
    at 0.40       0.1500
    at 0.50       0.1500
    at 0.60       0.0737
    at 0.70       0.0684
    at 0.80       0.0425
    at 0.90       0.0330
    at 1.00       0.0318
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.1514
Precision:
  At    5 docs:   0.2000
  At   10 docs:   0.2000
  At   15 docs:   0.2000
  At   20 docs:   0.2000
  At   30 docs:   0.1333
  At  100 docs:   0.0700
  At  200 docs:   0.0400
  At  500 docs:   0.0220
  At 1000 docs:   0.0110
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.1818
```

```
Queryid (Num):        13
Total number of documents over all queries
    Retrieved:     1000
    Relevant:        16
    Rel_ret:         15
Interpolated Recall - Precision Averages:
    at 0.00       1.0000
    at 0.10       0.4000
    at 0.20       0.3500
    at 0.30       0.3500
    at 0.40       0.3500
    at 0.50       0.3077
    at 0.60       0.2857
    at 0.70       0.1250
    at 0.80       0.0884
    at 0.90       0.0254
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.2671
Precision:
  At    5 docs:   0.4000
  At   10 docs:   0.2000
  At   15 docs:   0.2667
  At   20 docs:   0.3500
  At   30 docs:   0.3000
  At  100 docs:   0.1200
  At  200 docs:   0.0650
  At  500 docs:   0.0280
  At 1000 docs:   0.0150
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.2500
```

```
Queryid (Num):          14
Total number of documents over all queries
    Retrieved:          0
    Relevant:           0
    Rel_ret:            0
Interpolated Recall - Precision Averages:
    at 0.00         0.0000
    at 0.10         0.0000
    at 0.20         0.0000
    at 0.30         0.0000
    at 0.40         0.0000
    at 0.50         0.0000
    at 0.60         0.0000
    at 0.70         0.0000
    at 0.80         0.0000
    at 0.90         0.0000
    at 1.00         0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                    0.0000
Precision:
  At    5 docs:     0.0000
  At   10 docs:     0.0000
  At   15 docs:     0.0000
  At   20 docs:     0.0000
  At   30 docs:     0.0000
  At  100 docs:     0.0000
  At  200 docs:     0.0000
  At  500 docs:     0.0000
  At 1000 docs:     0.0000
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:          0.0000
```

```
Queryid (Num):          15
Total number of documents over all queries
    Retrieved:      1000
    Relevant:          2
    Rel_ret:           2
Interpolated Recall - Precision Averages:
    at 0.00        0.0455
    at 0.10        0.0455
    at 0.20        0.0455
    at 0.30        0.0455
    at 0.40        0.0455
    at 0.50        0.0455
    at 0.60        0.0455
    at 0.70        0.0455
    at 0.80        0.0455
    at 0.90        0.0455
    at 1.00        0.0455
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                   0.0400
Precision:
  At    5 docs:    0.0000
  At   10 docs:    0.0000
  At   15 docs:    0.0000
  At   20 docs:    0.0000
  At   30 docs:    0.0333
  At  100 docs:    0.0200
  At  200 docs:    0.0100
  At  500 docs:    0.0040
  At 1000 docs:    0.0020
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:         0.0000
```

```
Queryid (Num):          16
Total number of documents over all queries
    Retrieved:          0
    Relevant:           0
    Rel_ret:            0
Interpolated Recall - Precision Averages:
    at 0.00         0.0000
    at 0.10         0.0000
    at 0.20         0.0000
    at 0.30         0.0000
    at 0.40         0.0000
    at 0.50         0.0000
    at 0.60         0.0000
    at 0.70         0.0000
    at 0.80         0.0000
    at 0.90         0.0000
    at 1.00         0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                    0.0000
Precision:
  At    5 docs:     0.0000
  At   10 docs:     0.0000
  At   15 docs:     0.0000
  At   20 docs:     0.0000
  At   30 docs:     0.0000
  At  100 docs:     0.0000
  At  200 docs:     0.0000
  At  500 docs:     0.0000
  At 1000 docs:     0.0000
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:          0.0000
```

```
Queryid (Num):        17
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         8
    Rel_ret:          8
Interpolated Recall - Precision Averages:
    at 0.00       0.2941
    at 0.10       0.2941
    at 0.20       0.2941
    at 0.30       0.2941
    at 0.40       0.2941
    at 0.50       0.2941
    at 0.60       0.2941
    at 0.70       0.1132
    at 0.80       0.0805
    at 0.90       0.0576
    at 1.00       0.0576
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.1774
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.2000
  At   15 docs:   0.2000
  At   20 docs:   0.2500
  At   30 docs:   0.1667
  At  100 docs:   0.0700
  At  200 docs:   0.0400
  At  500 docs:   0.0160
  At 1000 docs:   0.0080
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.2500
```

```
Queryid (Num):        18
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         5
    Rel_ret:          5
Interpolated Recall - Precision Averages:
    at 0.00       1.0000
    at 0.10       1.0000
    at 0.20       1.0000
    at 0.30       0.5000
    at 0.40       0.5000
    at 0.50       0.2727
    at 0.60       0.2727
    at 0.70       0.1667
    at 0.80       0.1667
    at 0.90       0.0746
    at 1.00       0.0746
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.4028
Precision:
  At    5 docs:   0.4000
  At   10 docs:   0.2000
  At   15 docs:   0.2000
  At   20 docs:   0.1500
  At   30 docs:   0.1333
  At  100 docs:   0.0500
  At  200 docs:   0.0250
  At  500 docs:   0.0100
  At 1000 docs:   0.0050
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.4000
```

```
Queryid (Num):          19
Total number of documents over all queries
    Retrieved:         96
    Relevant:           5
    Rel_ret:            5
Interpolated Recall - Precision Averages:
    at 0.00         1.0000
    at 0.10         1.0000
    at 0.20         1.0000
    at 0.30         0.3750
    at 0.40         0.3750
    at 0.50         0.3750
    at 0.60         0.3750
    at 0.70         0.2500
    at 0.80         0.2500
    at 0.90         0.2500
    at 1.00         0.2500
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                    0.4242
Precision:
  At    5 docs:     0.2000
  At   10 docs:     0.3000
  At   15 docs:     0.2000
  At   20 docs:     0.2500
  At   30 docs:     0.1667
  At  100 docs:     0.0500
  At  200 docs:     0.0250
  At  500 docs:     0.0100
  At 1000 docs:     0.0050
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:          0.2000
```

```
Queryid (Num):        20
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         2
    Rel_ret:          2
Interpolated Recall - Precision Averages:
    at 0.00       0.0909
    at 0.10       0.0909
    at 0.20       0.0909
    at 0.30       0.0909
    at 0.40       0.0909
    at 0.50       0.0909
    at 0.60       0.0377
    at 0.70       0.0377
    at 0.80       0.0377
    at 0.90       0.0377
    at 1.00       0.0377
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0643
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.0000
  At   15 docs:   0.0667
  At   20 docs:   0.0500
  At   30 docs:   0.0333
  At  100 docs:   0.0200
  At  200 docs:   0.0100
  At  500 docs:   0.0040
  At 1000 docs:   0.0020
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):        21
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         1
    Rel_ret:          1
Interpolated Recall - Precision Averages:
    at 0.00       1.0000
    at 0.10       1.0000
    at 0.20       1.0000
    at 0.30       1.0000
    at 0.40       1.0000
    at 0.50       1.0000
    at 0.60       1.0000
    at 0.70       1.0000
    at 0.80       1.0000
    at 0.90       1.0000
    at 1.00       1.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  1.0000
Precision:
  At    5 docs:   0.2000
  At   10 docs:   0.1000
  At   15 docs:   0.0667
  At   20 docs:   0.0500
  At   30 docs:   0.0333
  At  100 docs:   0.0100
  At  200 docs:   0.0050
  At  500 docs:   0.0020
  At 1000 docs:   0.0010
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        1.0000
```

```
Queryid (Num):        22
Total number of documents over all queries
    Retrieved:     1000
    Relevant:        11
    Rel_ret:          9
Interpolated Recall - Precision Averages:
    at 0.00       0.1429
    at 0.10       0.1429
    at 0.20       0.1200
    at 0.30       0.0870
    at 0.40       0.0685
    at 0.50       0.0667
    at 0.60       0.0470
    at 0.70       0.0364
    at 0.80       0.0346
    at 0.90       0.0000
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0624
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.0000
  At   15 docs:   0.1333
  At   20 docs:   0.1000
  At   30 docs:   0.1000
  At  100 docs:   0.0600
  At  200 docs:   0.0350
  At  500 docs:   0.0180
  At 1000 docs:   0.0090
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):        23
Total number of documents over all queries
    Retrieved:        0
    Relevant:         0
    Rel_ret:          0
Interpolated Recall - Precision Averages:
    at 0.00       0.0000
    at 0.10       0.0000
    at 0.20       0.0000
    at 0.30       0.0000
    at 0.40       0.0000
    at 0.50       0.0000
    at 0.60       0.0000
    at 0.70       0.0000
    at 0.80       0.0000
    at 0.90       0.0000
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0000
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.0000
  At   15 docs:   0.0000
  At   20 docs:   0.0000
  At   30 docs:   0.0000
  At  100 docs:   0.0000
  At  200 docs:   0.0000
  At  500 docs:   0.0000
  At 1000 docs:   0.0000
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):        24
Total number of documents over all queries
    Retrieved:     1000
    Relevant:        12
    Rel_ret:          4
Interpolated Recall - Precision Averages:
    at 0.00       0.0833
    at 0.10       0.0278
    at 0.20       0.0101
    at 0.30       0.0101
    at 0.40       0.0000
    at 0.50       0.0000
    at 0.60       0.0000
    at 0.70       0.0000
    at 0.80       0.0000
    at 0.90       0.0000
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0108
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.0000
  At   15 docs:   0.0667
  At   20 docs:   0.0500
  At   30 docs:   0.0333
  At  100 docs:   0.0200
  At  200 docs:   0.0100
  At  500 docs:   0.0080
  At 1000 docs:   0.0040
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0833
```

```
Queryid (Num):          25
Total number of documents over all queries
    Retrieved:      1000
    Relevant:          2
    Rel_ret:           2
Interpolated Recall - Precision Averages:
    at 0.00        0.0909
    at 0.10        0.0909
    at 0.20        0.0909
    at 0.30        0.0909
    at 0.40        0.0909
    at 0.50        0.0909
    at 0.60        0.0909
    at 0.70        0.0909
    at 0.80        0.0909
    at 0.90        0.0909
    at 1.00        0.0909
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                   0.0812
Precision:
  At     5 docs:   0.0000
  At    10 docs:   0.0000
  At    15 docs:   0.0667
  At    20 docs:   0.0500
  At    30 docs:   0.0667
  At   100 docs:   0.0200
  At   200 docs:   0.0100
  At   500 docs:   0.0040
  At  1000 docs:   0.0020
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:         0.0000
```

```
Queryid (Num):        26
Total number of documents over all queries
    Retrieved:     1000
    Relevant:        19
    Rel_ret:         19
Interpolated Recall - Precision Averages:
    at 0.00       0.6667
    at 0.10       0.6667
    at 0.20       0.6250
    at 0.30       0.5714
    at 0.40       0.5714
    at 0.50       0.5714
    at 0.60       0.5714
    at 0.70       0.3659
    at 0.80       0.3636
    at 0.90       0.0239
    at 1.00       0.0239
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.4203
Precision:
  At    5 docs:   0.6000
  At   10 docs:   0.5000
  At   15 docs:   0.4000
  At   20 docs:   0.5500
  At   30 docs:   0.4000
  At  100 docs:   0.1600
  At  200 docs:   0.0850
  At  500 docs:   0.0340
  At 1000 docs:   0.0190
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.5263
```

```
Queryid (Num):        27
Total number of documents over all queries
    Retrieved:        0
    Relevant:         0
    Rel_ret:          0
Interpolated Recall - Precision Averages:
    at 0.00       0.0000
    at 0.10       0.0000
    at 0.20       0.0000
    at 0.30       0.0000
    at 0.40       0.0000
    at 0.50       0.0000
    at 0.60       0.0000
    at 0.70       0.0000
    at 0.80       0.0000
    at 0.90       0.0000
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0000
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.0000
  At   15 docs:   0.0000
  At   20 docs:   0.0000
  At   30 docs:   0.0000
  At  100 docs:   0.0000
  At  200 docs:   0.0000
  At  500 docs:   0.0000
  At 1000 docs:   0.0000
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):        28
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         2
    Rel_ret:          2
Interpolated Recall - Precision Averages:
    at 0.00       0.5000
    at 0.10       0.5000
    at 0.20       0.5000
    at 0.30       0.5000
    at 0.40       0.5000
    at 0.50       0.5000
    at 0.60       0.0027
    at 0.70       0.0027
    at 0.80       0.0027
    at 0.90       0.0027
    at 1.00       0.0027
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.2514
Precision:
  At    5 docs:   0.2000
  At   10 docs:   0.1000
  At   15 docs:   0.0667
  At   20 docs:   0.0500
  At   30 docs:   0.0333
  At  100 docs:   0.0100
  At  200 docs:   0.0050
  At  500 docs:   0.0020
  At 1000 docs:   0.0020
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.5000
```

```
Queryid (Num):        29
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         5
    Rel_ret:          4
Interpolated Recall - Precision Averages:
    at 0.00       1.0000
    at 0.10       1.0000
    at 0.20       1.0000
    at 0.30       0.7500
    at 0.40       0.7500
    at 0.50       0.7500
    at 0.60       0.7500
    at 0.70       0.1905
    at 0.80       0.1905
    at 0.90       0.0000
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.5214
Precision:
  At    5 docs:   0.6000
  At   10 docs:   0.3000
  At   15 docs:   0.2000
  At   20 docs:   0.1500
  At   30 docs:   0.1333
  At  100 docs:   0.0400
  At  200 docs:   0.0200
  At  500 docs:   0.0080
  At 1000 docs:   0.0040
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.6000
```

```
Queryid (Num):          30
Total number of documents over all queries
    Retrieved:          0
    Relevant:           0
    Rel_ret:            0
Interpolated Recall - Precision Averages:
    at 0.00         0.0000
    at 0.10         0.0000
    at 0.20         0.0000
    at 0.30         0.0000
    at 0.40         0.0000
    at 0.50         0.0000
    at 0.60         0.0000
    at 0.70         0.0000
    at 0.80         0.0000
    at 0.90         0.0000
    at 1.00         0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                    0.0000
Precision:
  At    5 docs:     0.0000
  At   10 docs:     0.0000
  At   15 docs:     0.0000
  At   20 docs:     0.0000
  At   30 docs:     0.0000
  At  100 docs:     0.0000
  At  200 docs:     0.0000
  At  500 docs:     0.0000
  At 1000 docs:     0.0000
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:          0.0000
```

```
Queryid (Num):          31
Total number of documents over all queries
    Retrieved:      1000
    Relevant:          3
    Rel_ret:           3
Interpolated Recall - Precision Averages:
    at 0.00        0.2500
    at 0.10        0.2500
    at 0.20        0.2500
    at 0.30        0.2500
    at 0.40        0.2500
    at 0.50        0.2500
    at 0.60        0.2500
    at 0.70        0.2500
    at 0.80        0.2500
    at 0.90        0.2500
    at 1.00        0.2500
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                   0.2222
Precision:
  At    5 docs:    0.0000
  At   10 docs:    0.2000
  At   15 docs:    0.2000
  At   20 docs:    0.1500
  At   30 docs:    0.1000
  At  100 docs:    0.0300
  At  200 docs:    0.0150
  At  500 docs:    0.0060
  At 1000 docs:    0.0030
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:         0.0000
```

```
Queryid (Num):        32
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         2
    Rel_ret:          2
Interpolated Recall - Precision Averages:
    at 0.00       1.0000
    at 0.10       1.0000
    at 0.20       1.0000
    at 0.30       1.0000
    at 0.40       1.0000
    at 0.50       1.0000
    at 0.60       1.0000
    at 0.70       1.0000
    at 0.80       1.0000
    at 0.90       1.0000
    at 1.00       1.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  1.0000
Precision:
  At    5 docs:   0.4000
  At   10 docs:   0.2000
  At   15 docs:   0.1333
  At   20 docs:   0.1000
  At   30 docs:   0.0667
  At  100 docs:   0.0200
  At  200 docs:   0.0100
  At  500 docs:   0.0040
  At 1000 docs:   0.0020
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        1.0000
```

```
Queryid (Num):          33
Total number of documents over all queries
    Retrieved:      1000
    Relevant:          3
    Rel_ret:           3
Interpolated Recall - Precision Averages:
    at 0.00        1.0000
    at 0.10        1.0000
    at 0.20        1.0000
    at 0.30        1.0000
    at 0.40        1.0000
    at 0.50        1.0000
    at 0.60        1.0000
    at 0.70        0.0064
    at 0.80        0.0064
    at 0.90        0.0064
    at 1.00        0.0064
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                   0.6688
Precision:
  At     5 docs:   0.4000
  At    10 docs:   0.2000
  At    15 docs:   0.1333
  At    20 docs:   0.1000
  At    30 docs:   0.0667
  At   100 docs:   0.0200
  At   200 docs:   0.0100
  At   500 docs:   0.0060
  At  1000 docs:   0.0030
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:         0.6667
```

```
Queryid (Num):        34
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         2
    Rel_ret:          2
Interpolated Recall - Precision Averages:
    at 0.00       0.5000
    at 0.10       0.5000
    at 0.20       0.5000
    at 0.30       0.5000
    at 0.40       0.5000
    at 0.50       0.5000
    at 0.60       0.5000
    at 0.70       0.5000
    at 0.80       0.5000
    at 0.90       0.5000
    at 1.00       0.5000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.4167
Precision:
  At    5 docs:   0.4000
  At   10 docs:   0.2000
  At   15 docs:   0.1333
  At   20 docs:   0.1000
  At   30 docs:   0.0667
  At  100 docs:   0.0200
  At  200 docs:   0.0100
  At  500 docs:   0.0040
  At 1000 docs:   0.0020
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):          35
Total number of documents over all queries
    Retrieved:          0
    Relevant:           0
    Rel_ret:            0
Interpolated Recall - Precision Averages:
    at 0.00         0.0000
    at 0.10         0.0000
    at 0.20         0.0000
    at 0.30         0.0000
    at 0.40         0.0000
    at 0.50         0.0000
    at 0.60         0.0000
    at 0.70         0.0000
    at 0.80         0.0000
    at 0.90         0.0000
    at 1.00         0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                    0.0000
Precision:
  At    5 docs:     0.0000
  At   10 docs:     0.0000
  At   15 docs:     0.0000
  At   20 docs:     0.0000
  At   30 docs:     0.0000
  At  100 docs:     0.0000
  At  200 docs:     0.0000
  At  500 docs:     0.0000
  At 1000 docs:     0.0000
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:          0.0000
```

```
Queryid (Num):          36
Total number of documents over all queries
    Retrieved:      1000
    Relevant:          3
    Rel_ret:           3
Interpolated Recall - Precision Averages:
    at 0.00        1.0000
    at 0.10        1.0000
    at 0.20        1.0000
    at 0.30        1.0000
    at 0.40        1.0000
    at 0.50        1.0000
    at 0.60        1.0000
    at 0.70        0.7500
    at 0.80        0.7500
    at 0.90        0.7500
    at 1.00        0.7500
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                   0.9167
Precision:
  At     5 docs:   0.6000
  At    10 docs:   0.3000
  At    15 docs:   0.2000
  At    20 docs:   0.1500
  At    30 docs:   0.1000
  At   100 docs:   0.0300
  At   200 docs:   0.0150
  At   500 docs:   0.0060
  At  1000 docs:   0.0030
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:         0.6667
```

REFERENCE

```
Queryid (Num):          37
Total number of documents over all queries
    Retrieved:          0
    Relevant:           0
    Rel_ret:            0
Interpolated Recall - Precision Averages:
    at 0.00         0.0000
    at 0.10         0.0000
    at 0.20         0.0000
    at 0.30         0.0000
    at 0.40         0.0000
    at 0.50         0.0000
    at 0.60         0.0000
    at 0.70         0.0000
    at 0.80         0.0000
    at 0.90         0.0000
    at 1.00         0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                    0.0000
Precision:
  At    5 docs:     0.0000
  At   10 docs:     0.0000
  At   15 docs:     0.0000
  At   20 docs:     0.0000
  At   30 docs:     0.0000
  At  100 docs:     0.0000
  At  200 docs:     0.0000
  At  500 docs:     0.0000
  At 1000 docs:     0.0000
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:          0.0000
```

```
Queryid (Num):        38
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         8
    Rel_ret:          7
Interpolated Recall - Precision Averages:
    at 0.00       0.2500
    at 0.10       0.2500
    at 0.20       0.0408
    at 0.30       0.0408
    at 0.40       0.0408
    at 0.50       0.0408
    at 0.60       0.0318
    at 0.70       0.0283
    at 0.80       0.0072
    at 0.90       0.0000
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0528
Precision:
  At    5 docs:   0.2000
  At   10 docs:   0.1000
  At   15 docs:   0.0667
  At   20 docs:   0.0500
  At   30 docs:   0.0333
  At  100 docs:   0.0400
  At  200 docs:   0.0250
  At  500 docs:   0.0120
  At 1000 docs:   0.0070
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.1250
```

```
Queryid (Num):        39
Total number of documents over all queries
    Retrieved:        0
    Relevant:         0
    Rel_ret:          0
Interpolated Recall - Precision Averages:
    at 0.00       0.0000
    at 0.10       0.0000
    at 0.20       0.0000
    at 0.30       0.0000
    at 0.40       0.0000
    at 0.50       0.0000
    at 0.60       0.0000
    at 0.70       0.0000
    at 0.80       0.0000
    at 0.90       0.0000
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0000
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.0000
  At   15 docs:   0.0000
  At   20 docs:   0.0000
  At   30 docs:   0.0000
  At  100 docs:   0.0000
  At  200 docs:   0.0000
  At  500 docs:   0.0000
  At 1000 docs:   0.0000
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):        40
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         5
    Rel_ret:          4
Interpolated Recall - Precision Averages:
    at 0.00       0.5000
    at 0.10       0.5000
    at 0.20       0.5000
    at 0.30       0.2000
    at 0.40       0.2000
    at 0.50       0.0055
    at 0.60       0.0055
    at 0.70       0.0043
    at 0.80       0.0043
    at 0.90       0.0000
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.1420
Precision:
  At    5 docs:   0.2000
  At   10 docs:   0.2000
  At   15 docs:   0.1333
  At   20 docs:   0.1000
  At   30 docs:   0.0667
  At  100 docs:   0.0200
  At  200 docs:   0.0100
  At  500 docs:   0.0040
  At 1000 docs:   0.0040
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.2000
```

```
Queryid (Num):          41
Total number of documents over all queries
    Retrieved:          0
    Relevant:           0
    Rel_ret:            0
Interpolated Recall - Precision Averages:
    at 0.00         0.0000
    at 0.10         0.0000
    at 0.20         0.0000
    at 0.30         0.0000
    at 0.40         0.0000
    at 0.50         0.0000
    at 0.60         0.0000
    at 0.70         0.0000
    at 0.80         0.0000
    at 0.90         0.0000
    at 1.00         0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                    0.0000
Precision:
  At    5 docs:     0.0000
  At   10 docs:     0.0000
  At   15 docs:     0.0000
  At   20 docs:     0.0000
  At   30 docs:     0.0000
  At  100 docs:     0.0000
  At  200 docs:     0.0000
  At  500 docs:     0.0000
  At 1000 docs:     0.0000
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:          0.0000
```

```
Queryid (Num):        42
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         3
    Rel_ret:          3
Interpolated Recall - Precision Averages:
    at 0.00       0.0208
    at 0.10       0.0208
    at 0.20       0.0208
    at 0.30       0.0208
    at 0.40       0.0150
    at 0.50       0.0150
    at 0.60       0.0150
    at 0.70       0.0032
    at 0.80       0.0032
    at 0.90       0.0032
    at 1.00       0.0032
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0130
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.0000
  At   15 docs:   0.0000
  At   20 docs:   0.0000
  At   30 docs:   0.0000
  At  100 docs:   0.0100
  At  200 docs:   0.0100
  At  500 docs:   0.0040
  At 1000 docs:   0.0030
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):        43
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         8
    Rel_ret:          8
Interpolated Recall - Precision Averages:
    at 0.00       1.0000
    at 0.10       1.0000
    at 0.20       1.0000
    at 0.30       1.0000
    at 0.40       1.0000
    at 0.50       1.0000
    at 0.60       0.6250
    at 0.70       0.6000
    at 0.80       0.2800
    at 0.90       0.0240
    at 1.00       0.0240
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.6911
Precision:
  At    5 docs:   0.8000
  At   10 docs:   0.6000
  At   15 docs:   0.4000
  At   20 docs:   0.3000
  At   30 docs:   0.2333
  At  100 docs:   0.0700
  At  200 docs:   0.0350
  At  500 docs:   0.0160
  At 1000 docs:   0.0080
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.6250
```

```
Queryid (Num):        44
Total number of documents over all queries
    Retrieved:     1000
    Relevant:        26
    Rel_ret:         26
Interpolated Recall - Precision Averages:
    at 0.00       1.0000
    at 0.10       1.0000
    at 0.20       0.5000
    at 0.30       0.4706
    at 0.40       0.3793
    at 0.50       0.1781
    at 0.60       0.1356
    at 0.70       0.1193
    at 0.80       0.1193
    at 0.90       0.0822
    at 1.00       0.0559
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.3420
Precision:
  At    5 docs:   0.8000
  At   10 docs:   0.5000
  At   15 docs:   0.4667
  At   20 docs:   0.4000
  At   30 docs:   0.3667
  At  100 docs:   0.1500
  At  200 docs:   0.1050
  At  500 docs:   0.0520
  At 1000 docs:   0.0260
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.3077
```

```
Queryid (Num):        45
Total number of documents over all queries
    Retrieved:      284
    Relevant:        26
    Rel_ret:         10
Interpolated Recall - Precision Averages:
    at 0.00       1.0000
    at 0.10       0.3750
    at 0.20       0.0366
    at 0.30       0.0366
    at 0.40       0.0000
    at 0.50       0.0000
    at 0.60       0.0000
    at 0.70       0.0000
    at 0.80       0.0000
    at 0.90       0.0000
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0989
Precision:
  At    5 docs:   0.4000
  At   10 docs:   0.3000
  At   15 docs:   0.2000
  At   20 docs:   0.1500
  At   30 docs:   0.1667
  At  100 docs:   0.0500
  At  200 docs:   0.0300
  At  500 docs:   0.0200
  At 1000 docs:   0.0100
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.1923
```

```
Queryid (Num):        46
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         2
    Rel_ret:          2
Interpolated Recall - Precision Averages:
    at 0.00       0.5000
    at 0.10       0.5000
    at 0.20       0.5000
    at 0.30       0.5000
    at 0.40       0.5000
    at 0.50       0.5000
    at 0.60       0.0800
    at 0.70       0.0800
    at 0.80       0.0800
    at 0.90       0.0800
    at 1.00       0.0800
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.2900
Precision:
  At    5 docs:   0.2000
  At   10 docs:   0.1000
  At   15 docs:   0.0667
  At   20 docs:   0.0500
  At   30 docs:   0.0667
  At  100 docs:   0.0200
  At  200 docs:   0.0100
  At  500 docs:   0.0040
  At 1000 docs:   0.0020
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.5000
```

```
Queryid (Num):        47
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         3
    Rel_ret:          3
Interpolated Recall - Precision Averages:
    at 0.00       0.1200
    at 0.10       0.1200
    at 0.20       0.1200
    at 0.30       0.1200
    at 0.40       0.1200
    at 0.50       0.1200
    at 0.60       0.1200
    at 0.70       0.1200
    at 0.80       0.1200
    at 0.90       0.1200
    at 1.00       0.1200
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.1121
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.1000
  At   15 docs:   0.0667
  At   20 docs:   0.1000
  At   30 docs:   0.1000
  At  100 docs:   0.0300
  At  200 docs:   0.0150
  At  500 docs:   0.0060
  At 1000 docs:   0.0030
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):          48
Total number of documents over all queries
    Retrieved:          0
    Relevant:           0
    Rel_ret:            0
Interpolated Recall - Precision Averages:
    at 0.00         0.0000
    at 0.10         0.0000
    at 0.20         0.0000
    at 0.30         0.0000
    at 0.40         0.0000
    at 0.50         0.0000
    at 0.60         0.0000
    at 0.70         0.0000
    at 0.80         0.0000
    at 0.90         0.0000
    at 1.00         0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                    0.0000
Precision:
  At    5 docs:     0.0000
  At   10 docs:     0.0000
  At   15 docs:     0.0000
  At   20 docs:     0.0000
  At   30 docs:     0.0000
  At  100 docs:     0.0000
  At  200 docs:     0.0000
  At  500 docs:     0.0000
  At 1000 docs:     0.0000
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:          0.0000
```

```
Queryid (Num):        49
Total number of documents over all queries
    Retrieved:     1000
    Relevant:         2
    Rel_ret:          2
Interpolated Recall - Precision Averages:
    at 0.00       0.3333
    at 0.10       0.3333
    at 0.20       0.3333
    at 0.30       0.3333
    at 0.40       0.3333
    at 0.50       0.3333
    at 0.60       0.0117
    at 0.70       0.0117
    at 0.80       0.0117
    at 0.90       0.0117
    at 1.00       0.0117
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.1725
Precision:
  At    5 docs:   0.2000
  At   10 docs:   0.1000
  At   15 docs:   0.0667
  At   20 docs:   0.0500
  At   30 docs:   0.0333
  At  100 docs:   0.0100
  At  200 docs:   0.0100
  At  500 docs:   0.0040
  At 1000 docs:   0.0020
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```

```
Queryid (Num):        50
Total number of documents over all queries
    Retrieved:        0
    Relevant:         0
    Rel_ret:          0
Interpolated Recall - Precision Averages:
    at 0.00       0.0000
    at 0.10       0.0000
    at 0.20       0.0000
    at 0.30       0.0000
    at 0.40       0.0000
    at 0.50       0.0000
    at 0.60       0.0000
    at 0.70       0.0000
    at 0.80       0.0000
    at 0.90       0.0000
    at 1.00       0.0000
Average precision (non-interpolated) for all rel
docs(averaged over queries)
                  0.0000
Precision:
  At    5 docs:   0.0000
  At   10 docs:   0.0000
  At   15 docs:   0.0000
  At   20 docs:   0.0000
  At   30 docs:   0.0000
  At  100 docs:   0.0000
  At  200 docs:   0.0000
  At  500 docs:   0.0000
  At 1000 docs:   0.0000
R-Precision (precision after R (= num_rel for a
query) docs retrieved):
    Exact:        0.0000
```