Internet Document Filtering Using Fourier Domain Scoring

Laurence A.F. Park, Marimuthu Palaniswami, and Ramamohanarao Kotagiri

ARC Special Research Centre for Ultra-Broadband Information Networks
Department of Electrical & Elecronic Engineering
The University of Melbourne
Parkville, Victoria, Australia 3010
{lapark,swami,rao}@ee.mu.oz.au
http://www.ee.mu.oz.au/cubin

Abstract. Most search engines return a lot of unwanted information. A more thorough filtering process can be performed on this information to sort out the relevant documents. A new method called Frequency Domain Scoring (FDS), which is based on the Fourier Transform is proposed. FDS performs the filtering by examining the locality of the keywords throughout the documents. This is examined and compared to the well known techniques Latent Semantic Indexing (LSI) and Cosine measure. We found that FDS obtains better results of how relevant the document is to the query. The other two methods (cosine measure, LSI) do not perform as well mainly because they need a wider variety of documents to determine the topic.

1 Introduction

The ability of automatically classifying a document accurately has become an important issue in the past few years due to the exponential growth of the Internet and the availability of on-line information. Many methods such as topic identification have been tried by search engines creators and abused by web page writers who try their best to mislead the search engine so that their page appears at the top of the search results.

At present, the only way to find any useful information on the Internet is to use a search engine and manually sort through all of the results returned.

There has been a huge interest in relevant document retrieval, and several people have developed methods to allow the user to obtain the right information. For example, Spertus [10] uses different types of connectivity and spatial locality to detect relevant pages; Mladenič [6] examines the pages previously visited by the user and uses these examples to learn what to retrieve in the future; Carrière et al. [3] calculates the score of a page based on how many relevant pages point to it through links; Ngu et al. [7] recommend that rather than search engines maintaining information about the entire Internet, each site should produce the information needed to produce better search results; Howe et al. [5] queries the existing range of search engines to obtain the best results from the collection of pages returned.

The method proposed here will examine the documents searched and try to find those with the topic requested by giving a score based on their spectrum, so that the user obtains the documents he/she truly requires.

This paper will proceed as follows, Sect. 2 contains a description of the document filtering process, Sect. 3 gives the problem formulation and explains how the current methods of filtering are not using all of the document information, Sect. 4 introduces the FDS method and explains how to calculate the score, Sect. 5 contains a short discussion on the computational complexity of the FDS method, Sect. 6 shows results from two separate experiments (one based on the TREC database and the other from three actual Internet searches) and gives an analysis of the results from both experiments, and finally the conclusion is given in Sect. 7.

2 Document Filtering

The objective of document filtering is to extract all of the relevant documents related to a certain topic from a set of documents of unknown topics. Examples of document filtering methods include the cosine measure, latent semantic indexing (LSI) and the new superior method Frequency Domain Scoring (FDS) introduced in this paper.

The methods used in this paper perform the filtering on a document set considered relevant by a selection of Internet search engines. Therefore the document filtering will be performed on a local machine rather than a remote machine.

Even though the trials were performed on the results of a few Internet search engines, these methods could easily be incorporated in the search engine itself.

3 Problem Formulation

Let $\mathcal{A}(t)$ be the entire collection of documents on the Internet at time t, where $\mathcal{A}(t) = \{d_0, d_1, \ldots, d_{\mathcal{N}}\}$ and $0 < \mathcal{N} < \infty$ is the number of documents on the Internet at time t. Each document d_n is represented as the tuple $\{i_n, \mathcal{L}_n\}$ where i is the information contained in document n and $\mathcal{L}_n \subset \mathcal{A}(t)$ is the set of documents which can be accessed through d_n via hypertext links.

There exists a set $\mathcal{R}_T \subset \mathcal{A}(t)$, where $\mathcal{R}_T$ is the set of all relevant documents to topic $T \in \mathbb{T}$ (the topic space). We want a function $\mathcal{S} : \mathbb{T} \to \mathcal{A}(t)$ such that $\mathcal{S}(T) = \mathcal{R}_T$. The current non-existence of the function $\mathcal{S}$ is the reason why search engines (which try to approximate $\mathcal{S}$) do not always return correct results. Internet search engines will give us a subset of $\mathcal{R}_T$ and a set of irrelevant information (related to the search topic T). This can be represented as $R_T \cup E_T$ where $R_T \subseteq \mathcal{R}_T$ and $E_T \subset \mathcal{A}(t) \backslash \mathcal{R}_T$.

Rather sifting through the entire collection of documents on the Internet ($\mathcal{A}(t)$), we will use the results from several search engines ($R_T \cup E_T$) and try to

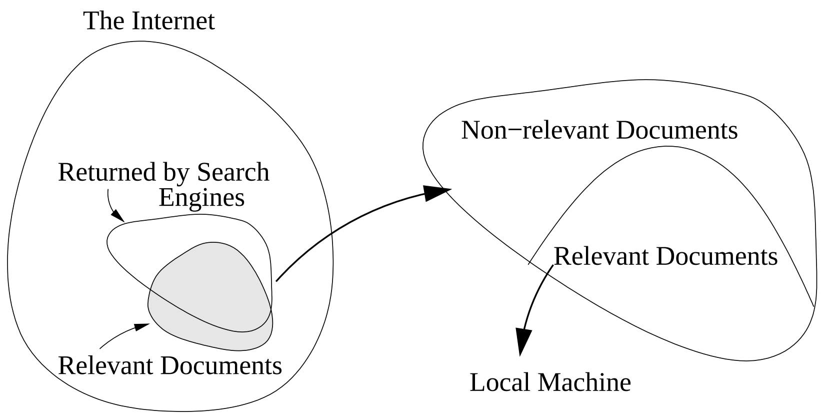

Fig. 1. This figure gives a visual example of the problem at hand. It can be seen that the set of documents returned by search engines contain irrelevant information. The problem is how do we extract the useful information from this set given to us

extract the set of documents R_T or remove the unwanted (irrelevant) E_T. This is shown in Fig. 1. This approach will lead to a good approximation of $\mathcal{S}$.

Current searching methods use the following approach. By defining a function $g : \mathcal{A}(t) \to \mathbb{R}^{\mathcal{M}}$, we are able to represent $d_n \in \mathcal{A}(t)$ as an $\mathcal{M}$ dimensional vector in real space. The mapping is performed by treating each word in the document as a single dimension, the number of times that word appears in the document will be its value. Therefore

$$g(d_n) = \delta_n = \left[c(d_n, w_1) \; c(d_n, w_2) \; \ldots \; c(d_n, w_{\mathcal{M}}) \right]^T \tag{1}$$

is a document vector, where $c(d_n, w_m) \in \mathbb{N}$ is the frequency count of word w_m, all w_m are unique and the dictionary contains $\mathcal{M}$ words. This document vector is then used to give the relevance for the document.

The above mapping of the document space into the M dimensional real space causes all of the important spatial information of the documents (the order of the words) to be lost. This also applies to the topic spatial information.

4 Frequency Domain Scoring

When a few key words are entered to search for, they are usually on the one topic. For example, if the words “aluminium cricket bat” is entered, we would expect to get documents on aluminium cricket bats. The classification methods listed so far would also return documents on cricket bats and aluminium.

To make use of the spatial information of the document, the vectors used here represent the positions of the search terms throughout the documents. Documents which have keywords appearing periodically and which contain the keywords together are given a higher relevance than the documents that have the keywords apart. To analyse the relative positions the vectors are mapped into the frequency domain.

Just as the discrete Fourier transform can map discrete time intervals in to the frequency domain, so to can it map discrete word spatial intervals into the frequency domain, as shown in Fig. 2

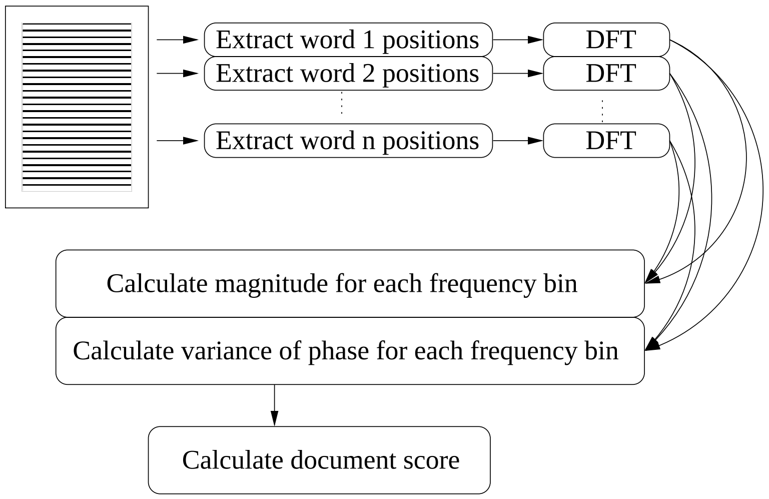

Fig. 2. This picture gives a visual example of how the Frequency Domain Scoring is performed. As shown, for each word in the search term, the document is split into equally sized bins. The value of each bin is the frequency of the word within that section of text. The DFT is performed and the magnitude and phase results are used to give the document score

By counting the number of appearances of a word in a document, we can treat the word position as the position in time. Performing the DFT allows us to observe the spectrum of the word. By splitting the spectrum into the magnitude and phase, we can see the power and delay of the word at certain frequencies.

By treating each word as a discrete time interval, we get a string of ones and zeros. To be more efficient, sequences of words can be clustered into bins (eg. the first fifty words in bin 1, the second fifty in bin 2, ...). This reduces the size of the input to the DFT and also gives larger counts than one in each bin.

Once the DFT is performed, the word spectrum shows the frequency components of which the word signal is made up. Each frequency component is a complex number of the form $H_f e^{i\phi_f}$ where $H_f \in \mathbb{R}$ represents the power of the frequency component f, and $\phi_f \in \mathbb{R}$ is the phase shift of f.

Terms made from several words are normally the topic of the document when the words appear close together and periodically. Therefore a document in which frequency f has a large magnitude (H_f) for all of the words from the set T, and the phases of each word from T are similar, then it is most likely that the T is a subset of the topic.

In mathematical terms, given a query T where

$$T = \{w_1, w_2, \ldots, w_M\} \tag{2}$$

we can define a counting function $\mathbf{cf} : \mathcal{A}(t) \times T \to \mathbb{R}^B$

$$\mathbf{cf}(d_n, w_m) = \left[c_1(d_n, w_m)/\beta \; c_2(d_n, w_m)/\beta \ldots c_B(d_n, w_m)/\beta \right] \tag{3}$$

where $c_b(d_n, w_m)$ is the count of word w_m in bin $1 \le b \le B$ of document d_n and β is the number of words per bin. The spectrum of $\mathbf{cf}$ can be found using the Fourier transform.

$$\begin{aligned}\mathcal{C}(d_n, w_m) &= \mathcal{F}\left[\mathbf{cf}(d_n, w_m)\right] &(4)\\ &= \left[H_1^{(n,m)} e^{i\phi_1^{(n,m)}} \; H_2^{(n,m)} e^{i\phi_2^{(n,m)}} \ldots H_B^{(n,m)} e^{i\phi_B^{(n,m)}} \right] &(5)\end{aligned}$$

where $\mathcal{F}$ is the Fourier Transform, $H_b^{(n,m)} \in \mathbb{R}$ and $\phi_b^{(n,m)} \in \mathbb{R}$ are the magnitude and phase of the bth frequency bin from the nth document and mth word respectively. If

$$\operatorname{var}\left(\left\{ \phi_b^{(n,1)}, \phi_b^{(n,2)}, \ldots, \phi_b^{(n,M)} \right\}\right) < \epsilon \tag{6}$$

and

$$H_b^{(n,m)} > E \qquad \forall\, m \tag{7}$$

where ϵ is a small value and E is a large value, then more likely that $d_n \in \mathcal{R}_T$. Therefore we want to give a higher relevance score to those documents with a higher magnitude and lower variance in phase of each frequency component.

The measure of variance does not take into account the circular data of the phase (modulo 2π). To overcome this problem, a measure of precision (rather than variance, not to be confused with the precision measure of document retrieval) was used. If

$$\bar{C}_b^{(n)} = \frac{1}{M}\sum_{m=1}^{M} \cos \phi_b^{(n,m)} \qquad \text{and} \qquad \bar{S}_b^{(n)} = \frac{1}{M}\sum_{m=1}^{M} \sin \phi_b^{(n,m)} \tag{8}$$

then the precision ($\bar{r}$) and mean ($\bar{\phi}$) are defined by

$$\bar{C}_b^{(n)} = \bar{r}_b^{(n)} \cos \bar{\phi}_b^{(n)} \qquad \text{and} \qquad \bar{S}_b^{(n)} = \bar{r}_b^{(n)} \sin \bar{\phi}_b^{(n)} \tag{9}$$

so

$$\bar{r}_b^{(n)} = \sqrt{\left(\bar{C}_b^{(n)}\right)^2 + \left(\bar{S}_b^{(n)}\right)^2} \tag{10}$$

The precision has a range of $[0, 1]$, where 1 is maximum precision. The notion of precision of the phases can be seen in the visual example given in Fig. 3.

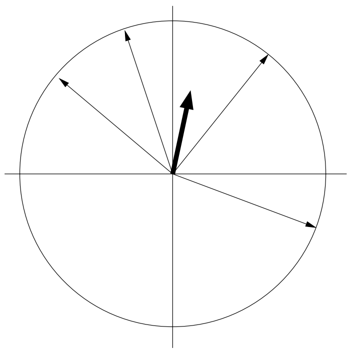

Fig. 3. This picture gives a visual example of how the precision function works. If each phase is considered as a unit vector (thin vectors) centered at zero, the precision will be the average of these vectors (thick vector)

This gives a score function of:

$$\mathrm{S}_{\mathrm{FDS}}\left(d_n, T\right) = \sum_{b=1}^{B} \bar{r}_b^{(n)} \log \left(\lambda + \sum_{m=1}^{M} \frac{H_b^{(n,m)}}{M} \right) \tag{11}$$

where

$$\lambda = \begin{cases} 0 & \text{if } H_b^{(n,m)} = 0 \ \forall \ m \\ Q & \text{if } H_b^{(n,m)} \neq 0 \ \forall \ m \end{cases} \tag{12}$$

where Q is a constant positive real number. Note that if the AC components are ignored, FDS will perform the same as the cosine measure. This is because $\bar{r}_1^{(n)} = 1$ for any n when $b = 1$. Therefore the cosine measure can be viewed as a special case of the FDS measure. By examining the spectrum of the words (and not just the count) we are able to obtain a better understanding of the content of the document.

The λ value was inserted to give a higher ranking to documents which contain all of the words in the query. This can be adjusted to suit the users preference.

5 Computational Complexity

Choosing an information retrieval method just because it gives accurate results is not sufficient. The methods have to be practical. This is why computational complexity comes into play when deciding which method is best. In most cases there is a trade off between speed and accuracy, where the level chosen should suit the user.

All methods suggested require scanning through the documents, word by word. This stage only depends on the length of the documents and has been omitted from the analysis since it is common to all methods. This process can also be pre-computed and stored for future classifications.

The FDS method performs the Fourier transforms on each word in the search query. Since the FDS method depends only on the document being processed, the spectral values have to be calculated only once and can then stored for later use.

To speed up the process, the Radix-2 FFT [8] was used. The only drawback to using this method is that the bin size must be a power of two ($B = 2^p$, $p \in \mathbb{N}$). This drops the computational complexity from $O(N^2)$ (direct DFT calculation) to $O\left(\frac{N}{2}\log_2 N\right)$ (Radix-2).

This implies that for M unique words in the search query, the time taken to calculate the score of one document will be in the order of $O\left(\frac{MN}{2}\log_2 N\right)$.

6 Results

We conducted two large experiments, one using the TREC database [11] and the other using a database of documents selected from the results of Internet search engines.

6.1 Preliminaries

To find how effective the FDS method was we compared the results given with trials using the cosine measure [12] and Latent Semantic Indexing (LSI) [1]. Pre-processing was performed on the data to make computation easier and give fairer results. This consisted of removing stop words, stemming, and using log-entropy normalisation (found in [4]). After performing several trials using different values of Q, it was chosen to be 1. The bin size was set to 16 to give a good tradeoff between accuracy and disk space. The document filtering methods FDS, cosine measure, and LSI are then performed to evaluate their relative merits.

The results (in table 2 and table 3) were evaluated by examining two accuracy measures of precision, which in the information retrieval sense, is the measure of the proportion of relevant documents to retrieved documents. The two measures are:

Average Precision This value is best explained by observing table 1.
***R*-Precision** is the precision after the first R documents, where R is the number of relevant documents for that query.

The time taken to perform the ranking was very similar for each of the methods.

6.2 Methods Compared

Cosine Measure. The score for each document was calculated by finding the normalised dot product of the document vector (shown in equation 1 and the query vector. This method will be referred to as COS throughout this document.

Table 1. The average precision is calculated by first calculating the sub-precisions of each relevant document (shown on each line of the table). Sub-precision is found by including only the documents ranked higher than the selected relevant document

Relevant document number	Rank	Precision
1	r_1	$1/r_1$
2	r_2	$2/r_2$
3	r_3	$3/r_3$
$\vdots$	$\vdots$	$\vdots$
n	r_n	n/r_n
	Average Precision:	$\frac{\sum_{i=1}^{n} i/r_i}{n}$

Latent Semantic Indexing. Latent Semantic Indexing produces document vectors of smaller dimension than the cosine measure, but closest to them in the least squares sense, via Singular Value Decomposition (SVD) [2]. This reduction of dimensionality reveals a latent structure of the documents that would not have been noticed otherwise. In this experiment, the dimension was reduced to 280.

Due to the large amounts of data and most of it being zeros, a sparse matrix data structure was used (found in [9]). Results were found by comparing each document vector to the query vector using the normalised dot product. The query vector was created by taking the average of the word vectors (produced by the SVD) that appeared in the search term. This method will be referred to as LSI throughout this document.

6.3 Experiment One: TREC Data

The TREC data [11] contains millions of documents from many different sources. This is useful to evaluate Internet search engines and searching tools for large databases. The method proposed in this paper is a refining process. It takes a subset of the whole data and extracts the truly relevant information from that. To emulate the process of the search engines, a number of random documents were chosen from the original data set, while making sure the documents classified relevant and irrelevant were included (shown in Fig. 4). The irrelevant documents are those that have been classified as relevant by other information retrieval methods but found to be wrong. By including these documents, the information retrieval methods applied here will be truly challenged. The results were evaluated using 'trec_eval', which was supplied by the TREC organisers.

The results from using the TREC data are shown in table 2. The data focused on was from the Associated Press document set, using queries 101 through to 200. Two experiments were run, the first processed 500 pseudo-random documents from the AP document set, the second processed 1000 documents.

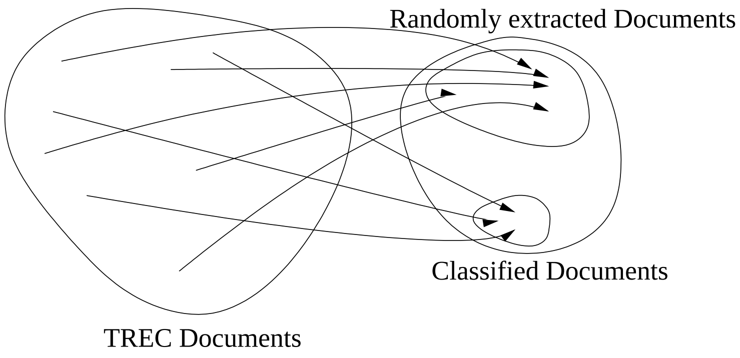

Fig. 4. To emulate the process of the search engines, a sample of relevant and irrelevant documents were taken from the TREC data set to feed into the document filtering process

It should be noted that it would have been very unlikely that any document contained every word in a selected query, due to the nature of the query. Rather than being a phrase or a few key words (which is what would be normally supplied to a search engine), the queries were structured into the form of a description of desired documents.

Table 2. Results given by trec_eval from a semi-random sample of documents using various filtering methods. FDS has the greatest precision for 500 documents and gives similar results to COS for 1000 documents

	500 documents		1000 documents	
Method	Average Precision	R-Precision	Average Precision	R-Precision
FDS	0.4439	0.3972	0.4560	0.3933
COS	0.4371	0.3830	0.4598	0.3930
LSI	0.3756	0.3508	0.3661	0.3508

The results show that FDS gives better results compared to the other methods. For all of the methods, the scores are low. A reason for this is the way the TREC queries are set up. A typical TREC query is of the form :

A {type of document} will identify {case 1} and {case 2} or ... but not ...

Therefore it requires parsing to obtain the keywords and anti-keywords. Some of the query results with lower precision did not contain examples. They only contained statements like "...contains information about a country...". The methods used contained no data on what country names are and so could not find the relevant documents.

6.4 Experiment Two: Internet Documents

The following results were obtained from a data set of documents returned from various search engines on the Internet after searching for three items with various degrees of difficulty. The query *"Aluminium Cricket Bat"* had only a few relevant documents, *"Bullet the blue sky mp3"* had some relevant documents, and *"Prank calls from Bart Simpson to Moe"* contained many relevant documents. The data was cleaned by extracting the text from the html structure, changing all the letters to lowercase, removing any stop-words (listed in a previously compiled list) and converting each word to its stem by using Porter's Stemming Algorithm.

The html documents were individually examined and assigned a label of *relevant* or *not relevant*. These were then compared with the score given by the listed methods (mentioned throughout the document). The results are presented in table 3 showing the method, the number of documents used, the number of relative documents, and the precision of the method for that search.

Table 3. This table shows how well the each method worked on different sets of documents retrieved from the Internet

Search for "Aluminium Cricket Bat"

Method	No. of Documents		Average Precision	R-Precision
	Total	Relevant		
FDS	120	2	0.5667	0.5000
COS	120	2	0.2857	0.5000
LSI	120	2	0.2845	0.5000

Search for "Bullet the blue sky mp3"

Method	No. of Documents		Average Precision	R-Precision
	Total	Relevant		
FDS	120	13	0.9822	0.9231
COS	120	13	0.7467	0.7692
LSI	120	13	0.6567	0.6923

Search for "Prank calls from Bart Simpson to Moe"

Method	No. of Documents		Average Precision	R-Precision
	Total	Relevant		
FDS	120	27	0.9279	0.9259
COS	120	27	0.7425	0.7778
LSI	120	27	0.7349	0.7778

By observing the results obtained, it can be seen that the FDS technique is a superior method and works far better than the LSI and COS document indexing schemes. This is due to the fact that FDS is able to extract more information from the documents. LSI and COS treat the documents as though they are a '*bag of words*', while FDS observes any structures of the searched terms in the document. COS can be considered a special case of the FDS method, therefore FDS is expected to obtain better results.

7 Conclusion

While Internet search engines produce good results, they don't always give us exactly what we want. The proposed FDS method over comes this problem by filtering the results given by the search engines. The LSI and COS methods need a wide range of document types to really focus on the important documents. In this case most of the documents will be of the same class and therefore these methods will not work as well as the new FDS method.

LSI and COS methods could be considered sub-methods of FDS. LSI and COS methods consider only the DC components while FDS makes use of the full spectrum. This shows that LSI and COS do not require as much storage space for the calculations since it only takes a fraction of the data needed by FDS. But at the current rate in which storage media is growing in capacity, this is hardly an issue. The size of the stored information is proportional to the number of frequency components, which can be adjusted by changing the words per bin.

FDS can be implemented on the client side (as discussed throughout this document) or it could be implemented on the server side. It can easily be included in systems like Internet search engines since it is scalable (when extra documents are introduced into the database, the other documents are not affected), the frequency data can easily be put into an indexing table (current indexing tables only include the DC component, therefore FDS would be a simple extension of this), and the most important reason is that it gives excellent results. Including the FDS method in any search engine would boost the quality (in terms of results) of the search engine, and return a more relevant document set.

Acknowledgements. We would like to thank the ARC Special Research Centre for Ultra-Broadband Information Networks for their support and funding of this research.

References

1. M. W. Berry, S. T. Dumais, and G. W. O'Brien. Using linear algebra for intelligent information retrieval. Technical report, Computer Science Department, The University of Tennessee, Knoxville, TN, December 1994.
2. Michael W. Berry. *Large scale sparse singular value computations.* Department of Computer Science, University of Tennessee, 107 Ayres Hall, Knoxville, 1993.
3. S. Jeromy Carrière and Rick Kazman. Webquery: searching and visualising the web through connectivity. *Computer Networks and ISDN Systems*, 29:1257–1267, 1997.
4. Susan T. Dumais. Improving the retrieval of information from external sources. *Behaviour Research Methods, Instruments & Computers*, 23(2):229–236, 1991.
5. Adele E. Howe and Daniel Dreilinger. Savvysearch. *AI Magazine*, pages 19–25, Summer 1997.
6. Dunja Mladeniĉ. Personal webwatcher: design and implementation. Technical report, Dept. for Intelligent Systems, J. Stefan Institute, Jamova 39, 11000 Ljubljana, Slovania, 1996.

7. Daniel Siaw Weng Ngu and Xindong Wu. Site helper: a localised agent that helps incremental exploration of the world wide web. *Computer Networks and ISDN Systems*, 29:1249–1255, 1997.
8. John G. Proakis and Dimitris G. Manolakis. *Digital signal processing : principles, algorithms, and applications.* Prentice-Hall, Inc, 3rd edition, 1996.
9. Yousef Saad. *Iterative methods for sparse linear systems.* PWS series in computer science. PWS Pub. Co., Boston, 1996.
10. Ellen Spertus. Parasite: mining structural information on the web. *Computer Networks and ISDN Systems*, 29:1205–1215, 1997.
11. National Institute Of Standards and Technology. Text retrieval conference (trec) `http://trec.nist.gov/`. World Wide Web, 2001.
12. Ian H. Witten, Alistair Moffat, and Timothy C. Bell. *Managing gigabytes : compressing and indexing documents and images.* Morgan Kaufmann Publishers, 1999.