SUBMITTED FOR PUBLICATION TO: IEEE TRANS. KNOWLEDGE AND DATA ENGINEERING, 2002, MAY 22, 2002

Fourier Domain Scoring : A novel document ranking method

Laurence A. F. Park Kotagiri Ramamohanarao and Marimuthu Palaniswami

Laurence A. F. Park is with the ARC Special Research Centre for Ultra-Broadband Information Networks, Department of Electrical and Electronic Engineering, The University of Melbourne, Parkville, Victoria, Australia 3010. Phone: (+61 3 8344 3822) E-mail: lapark@ee.mu.oz.au

Kotagiri Ramamohanarao (corresponding author) is with the ARC Special Research Centre for Ultra-Broadband Information Networks, Department of Computer Science and Software Engineering, The University of Melbourne, Parkville, Victoria, Australia 3010. Phone: (+61 3 8344 9101) E-mail: rao@cs.mu.oz.au

Marimuthu Palaniswami is with the Department of Electrical and Electronic Engineering, The University of Melbourne, Parkville, Victoria, Australia 3010. Phone: (+61 3 8344 6710) E-mail: swami@ee.mu.oz.au

Fourier Domain Scoring : A novel document ranking method

Laurence A. F. Park Kotagiri Ramamohanarao and Marimuthu Palaniswami

Abstract

Current document retrieval methods use a vector space similarity measure to give scores of relevance to documents when related to a specific query. The central problem with these methods is that they neglect any spatial information in the documents of question. We present a method called Fourier Domain Scoring (FDS) which takes advantage of this spatial information, via the Fourier transform, to give a more accurate ordering of relevance to a document set. We show that FDS gives a 60% improvement in precision over the vector space similarity measures for the common case of small queries, and that FDS matches the precision of the vector space similarity measures in the case of large queries.

Index Terms

Fourier Domain Scoring, Information retrieval, Search engine, vector space similarity measure, document ranking

I. Introduction

THE BIRTH of the World Wide Web just over a decade ago created many new areas of research and has brought back to life some that had been stagnant. The Web (which can be considered a huge database of documents) has become so popular that its content has grown to over a billion documents. With so much information, the World Wide Web has the need for some organization. Today this organization comes in the form of a Web search engine. The job of the search engine is to remove the chaos and clutter of the Web by allowing users to supply key words so that the search engine can find information relevant to these key words on the Web. Due to the importance of search engines, the interest in the field of information retrieval has risen.

There have been many methods for seeking out information on the Web. Many new techniques involve utilizing the HTML tags found on Web pages to obtain a higher understanding of the page content [1], [2], [3], [4], [5]. There have also been many page creators who have abused searching methods to obtain higher rankings in search engine results. A few search engines have branched out to analyzing pages in the form of text, PDF, Postscript, and a few Microsoft file formats as well as the traditional HTML.

To identify the topic of a document, the text within must be analyzed. This is why the text analysis method used in any search engine is critical. By obtaining a clear understanding of a page, we are able to find more relevant documents and make it harder for people to cheat the system.

Some variant of the vector space model [6] has been the dominant method used to analyse the text in information retrieval systems for many years. The concept behind the vector space model is to convert each document into a vector, so they can be easily compared to other document vectors (to find similarity amongst documents) and they can also be compared to query vectors (to find relevance to the query). The

Laurence A. F. Park is with the ARC Special Research Centre for Ultra-Broadband Information Networks, Department of Electrical and Electronic Engineering, The University of Melbourne, Parkville, Victoria, Australia 3010. Phone: (+61 3 8344 3822) E-mail: lapark@ee.mu.oz.au

Kotagiri Ramamohanarao (corresponding author) is with the ARC Special Research Centre for Ultra-Broadband Information Networks, Department of Computer Science and Software Engineering, The University of Melbourne, Parkville, Victoria, Australia 3010. Phone: (+61 3 8344 9101) E-mail: rao@cs.mu.oz.au

Marimuthu Palaniswami is with the Department of Electrical and Electronic Engineering, The University of Melbourne, Parkville, Victoria, Australia 3010. Phone: (+61 3 8344 6710) E-mail: swami@ee.mu.oz.au

document vectors exist in a space where each dimension is one unique term from the document set. The size of the document vector along each dimension is a weighted count of that word in the document.

Many information retrieval systems employ the use of a vector space model to classify text. Some examples of current work which use the vector space model are SMART by Buckley *et al.* [7], [8], the Okapi Basic Search System by Robertson *et al.* [9], IRIS by Yang *et al.* [10] and INQUERY by Allan *et al.* [11].

The problem with these techniques is that any spatial information contained in the documents is lost. Once the documents are converted into document vectors, the number of times each word appears is represented in the vector, but the positions of the words (the flow of the document) is ignored.

We present a method entitled Fourier Domain Scoring (FDS) which can retain the document spatial information and use it to effectively rank documents. The difference between FDS and other vector space similarity measures is that rather than storing only the count of a frequency term per document, FDS stores a term signal. The term signal shows how the term is spread throughout the document. If the spectrum of the query term signals are compared, we are able to observe which documents have a high occurrence of the query terms and which documents have the query terms appearing together. This information is obtained by comparing the magnitude and phase of the spectrum across different term signals respectively.

FDS is a new text classification which can be used in information retrieval systems by replacing the existing vector space similarity measures. Just as in information retrieval systems which use vector space similarity measures, other enhancements can be added on top of FDS to fully utilize the document environment and improve the results of the search (as in examining hypertext links on the Web, using a thesaurus,...).

Cases of systems which use the vector space similarity measures which could easily be replaced by FDS can be found everywhere. Some examples are as follows.

Google [12] records the position of every word in the document to give a higher ranking to document which contain query terms closer together. To calculate the distances between every term would require a lot of calculations at query time, or a large space to store all of the previously calculated distances. FDS could easily be implemented in the Google search engine to store the precalculated phases of each word. Requiring less space and less time for calculations at query time.

Ebert *et al.* [13] describe a method of visualizing clusters in documents. This system takes sequences of characters from the text to create vectors for the user to explore. It would be very interesting to replace these n-gram vectors with the Fourier transform of each character sequence or word sequence. This might provide a more compact representation and give better results.

The Latent Semantic Indexing method [14] uses singular value decomposition (SVD) to reduce the dimension of the index used by conventional search engines. By reducing the dimension, we are trying to remove noise generated by our use of language and trying to find the true semantic structure of the document. This method uses document vectors containing the count of each word as the elements, therefore any spatial information is lost. If the document vectors were replaced by a document matrix, generated by FDS, then SVD could easily be performed on each of the sets of spatial bins. This would produce a structural latent semantic index.

This paper is organised as follows. Section II will give a brief introduction to the Fourier Domain Scoring method and outline the basic steps needed to perform such a task on large document sets. Section III will list the major stages which will affect the score given to a document and further explain different methods which can be performed in each of the stages. Section IV will show some of the properties of the Fourier transform and prove some of the statements made in section III. Section V will look into the new data extracted from the document and how it copes with large queries. Section VI explains in detail the experiments performed using the TIPSER data set with short queries and compares the results with

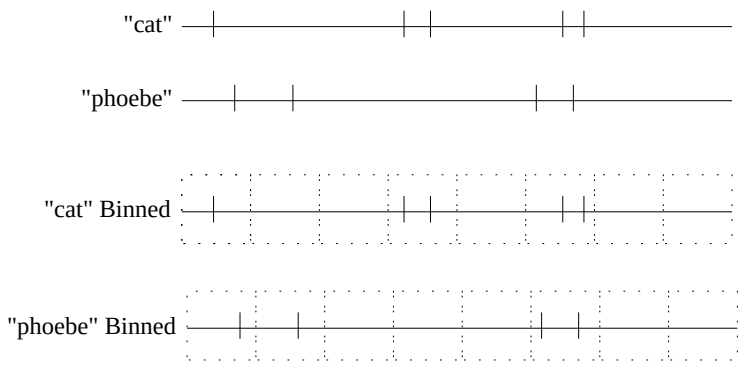

Fig. 1. A visual example of how the word signals are obtained. The top two lines, labeled "cat" and "phoebe" show the positions of the words cat and phoebe in some document (the position is signified by the vertical stroke through the line). The bottom half shows the binned positions of the words.

existing ranking methods. Finally, section VII contains the conclusion.

II. Fourier Domain Scoring

Vector space similarity measures will only give a score based on the count of each word in the document. FDS tries to capture the appearance of the word through the document. Words can then be compared to find if they are relevant to the document.

The main steps of FDS are:

- Collect words into spatial bins
- Create inverted index
- Perform preweighting
- Perform Fourier transform
- Calculate document scores

Some initial work on this topic can be found in [15]. We discuss each of these steps in brief as it is essential to understand how FDS works before we can examine any of its properties.

A. Collect words into spatial bins

Rather than mapping a document to a vector that contains the count of each word, FDS maps each document into a set of word vectors. These word vectors show the position of the word throughout the document, where the position of the element represents the position of the word in the document. This vector can also be thought of as the word signal in the document. The simplest word vector would contain elements equal to the word count of the document. For every position where the word appeared in the document, a '1' would appear in the vector in its corresponding position. Where the word does not appear a '0' would occur. This leads to a very large vector. To reduce the word vector size (and hence the calculations needed), the words should be grouped into bins. If the number of bins is set to B, a document containing W words would have W/B words in each bin. Therefore the first element of the word vector would contain the number of times the word appeared in the first W/B words. The second element would be the number of times the word appeared in the second W/B words, and so on. For example, the top half of figure 1 gives us the positions of the words "cat" and "phoebe" throughout a document (signified by the vertical bars). If we choose $B = 8$, we obtain the word signal $[1\ 0\ 0\ 2\ 0\ 2\ 0\ 0]$ for the word "cat" in that document and $[1\ 1\ 0\ 0\ 0\ 2\ 0\ 0]$ for the word "phoebe" in that document. A visual example of this can be seen in figure 1.

B. Create inverted index

Just as in any vector space model, an inverted index can be created to enable quick retrieval of word vectors. In this case, a little more information needs to be stored due to the documents being split into bins. In this case the words in each document were represented as:

$$< n >< b_1, f_1 >< b_2, f_2 > \ldots < b_n, f_n > \tag{1}$$

where n is the number of non-zero bins, b_a is the bin number and f_a is the count of the word in bin b_a. A table was also set up containing the words and references to the position in the inverted index which the word begins (for fast retrieval).

C. Weighting

The cosine similarity measure can be improved considerably by adding weighting to the document vectors before the score is calculated [16]. The weighting which we will be using when performing the cosine measure (shown in Witten *et al.* [6]) is the typical TF×IDF rule:

$$\mathrm{S}_{\mathsf{cos}}(d,T) = \frac{1}{W_d W_t} \sum_{t \in T \cap d} (1 + \log_e f_{d,t}) \cdot \log_e \left(1 + \frac{N}{f_t}\right) \tag{2}$$

where the $\mathrm{S}_{\mathsf{cos}}(d,T)$ is the score given to the dth document using query terms T, the TF (term frequency) is $1 + \log_e f_{d,t}$, which depends on the count of term t in document d ($f_{d,t}$). The IDF (inverse document frequency) is $\log_e \left(1 + \frac{N}{f_t}\right)$ which depends on the number of documents in which term t appears (f_t) and the number of documents (N). Rather than calculating this at query time, the weighting is done when the document index is created. The term frequency weighting is performed to reduce the effect of more than one appearance of the keyword on the score (eg. if a word appears twice in a document, it does not mean that that document is twice as relevant as a document in which the key word only appears once). The inverse document frequency weighting reduces the effect of common words to the score (eg. if we are searching for 'large dog' and 'large' appears in many documents, but 'dog' only in a few, 'dog' should have more effect on the score).

FDS also requires an index to be built, therefore preweighting (applying weights before performing the Fourier transform) can be performed to increase the accuracy of the relevance scores. Since we will be comparing FDS to the TF×IDF cosine measure, we will also use TF×IDF for preweighting. But the words are split into spatial bins, so how do we apply this weighting? The methods proposed are a modification of the TF weighting scheme:

- Apply this weighting to each spatial bin to become term bin frequency × inverse document frequency (TBF×IDF):

$$\mathrm{TBF}: \ 1 + \log_e f_{d,t,b} \tag{3}$$

  where $f_{d,t,b}$ is the count of term t in spatial bin b of document d.
- Or apply the weighting to the document frequency values and divide this proportionally amongst the spatial bins to become proportional term frequency × inverse document frequency (PTF×IDF):

$$\mathrm{PTF}: \ (1 + \log_e f_{d,t}) \left(\frac{f_{d,t,b}}{f_{d,t}}\right) \tag{4}$$

The IDF will be used with these TF modifications in the latter experiments. The original IDF is used because the binning of words does not affect them.

D. Perform Fourier transform

The Fourier transform has been used in many areas of electrical engineering. It is designed to change the basis of a signal (or function) to linearly independent sinusoidal waves. The discrete form of the transform (DFT) is of the form:

$$\nu_{d,t,\beta} = \sum_{b=0}^{B-1} \omega_{d,t,b} \exp\left(\frac{-i2\pi\beta b}{B}\right) \tag{5}$$

in our case the signal will be the weighted word signal consisting of elements $\omega_{d,t,b}$ where $b \in \{0, 1, \ldots, B-1\}$. Since each $\nu_{d,t,b}$ is the projection of the word signal $\omega_{d,t,\cdot}$ onto a sinusoidal wave of frequency β, the signal $\nu d, t\cdot$ is the spectrum of the given word signal. The spectral component value β is an element of the set $\{0, 1, \ldots, B-1\}$.

Therefore, the Fourier transform maps a signal from the time or spatial domain to the frequency domain. Once the spectrum of a signal can be seen, we are able to identify the major frequency components which give the signal its shape. The Fourier transform produces the following mapping:

$$\{\omega_{d,t,b}\} \rightarrow \{\nu_{d,t,b}\} = \{H_{d,t,b} \exp(i\phi_{d,t,b})\}$$

where $\omega_{d,t,b}$ is the weight of term t in bin b of document d, $\nu_{d,t,b}$ is the bth frequency component of term t in document d, $H_{d,t,b}$ and $\phi_{d,t,b}$ are the magnitude and phase of frequency component $\nu_{d,t,b}$ respectively (and are therefore real values) and i is $\sqrt{-1}$.

Shannon's sampling theorem states that the greatest frequency component to be found in a real signal is equal to half of the sampling rate. This implies that if we choose B bins for the word signal, we will only need to examine frequency components 0 to $\frac{B}{2}$ to analyse all of the information found in the spatial word signal.

By performing the Fourier transform on the spatial word bins, we are able to see how the word flows through the document. Each frequency component contains magnitude and phase information which can be interpreted as the effect and shift of the component respectively. The effect gives us an idea of the shape of the word signal. If a lower frequency component magnitude is large with respect to the other components, then the word should appear clustered in a few places in the document. If a higher frequency component magnitude is large with respect to the other components, then the word clusters would appear more frequently. The shift represents the position of the word throughout the document and is measured in radians. The shift is useful when comparing word signals. If two signals have the same phase then they are said to be 'in phase', meaning they appear together. But if two signals have opposite phase (different by π radians), then they are 'out of phase'. Therefore, if two or more word signals are in phase, this implies that they appear together throughout the document. If two of more words are not in phase, it implies that they appear in the document but not together most of the time.

For example, if we consider the word signals for the words "cat" and "phoebe" at the top of figure 2, we can see that they appear together in bins 0 and 5. If we perform the DFT on each signal we can examine the magnitude and phase of the spectral components. If a component c from word 1 has a similar phase to component c from word 2, these two components are considered in phase, which implies that the words 1 and 2 appear together in that region of the spectrum. This is the case for components 0, 3 and 4 when comparing the words "cat" and "phoebe". If the magnitude for those components are high (relative to the other components), it implies that these are major frequency components of the word signal. This is also the case with components 0, 3 and 4 when comparing the words "cat" and "phoebe". Therefore this document can be considered relevant to the query "cat" and "phoebe". An example of the effect of the Fourier transform on a word signal can be seen in figure 2.

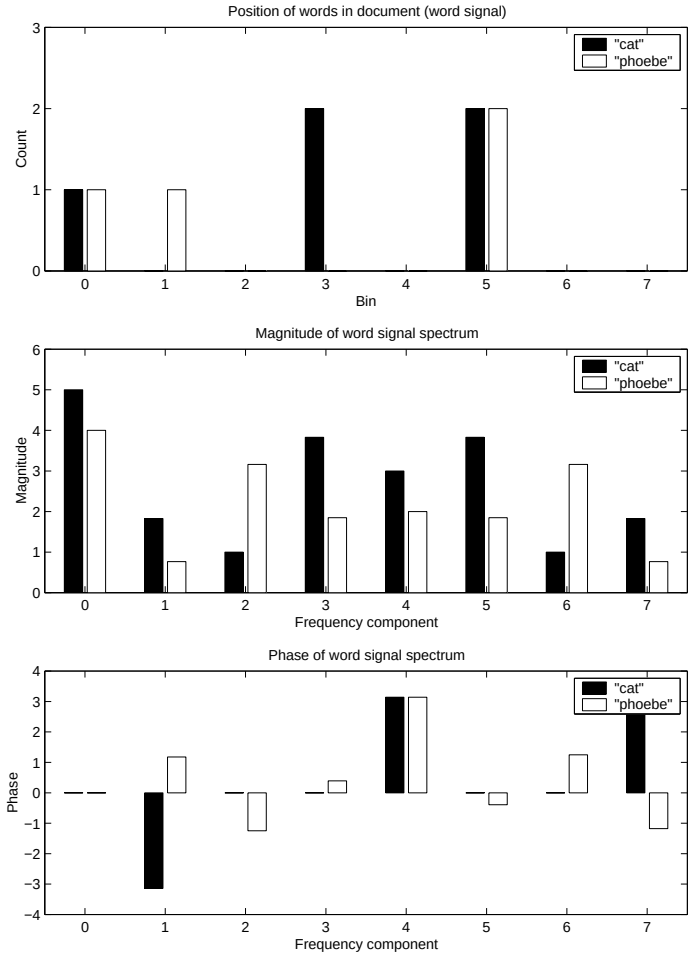

Fig. 2. An example of before and after the Fourier transform. We can see the binned positions of word 1 and word 2 from a certain document in the top plot. The bottom two plots show the information obtained after computing the Fourier transform on each of the binned word signals. Notice the reflection of the spectral information about component 4 (as stated in Shannon's sampling theorem).

E. Calculate document scores

The final step to the FDS process is to combine the spectral information of the keywords from the document to obtain a single relevance score. From intuition, a relevant document should have large magnitudes and the corresponding phases from each word should be similar (in phase). But how do we combine the spectral components so that we can obtain a score to give precise rankings? There are many methods which can be used. These are explained in detail in section III

III. CALCULATING DOCUMENT SCORE

This section will provide a thorough investigation of some of the methods of word spectrum and component combination.

A. Combining word spectrums

Once we have performed the Fourier transform on the B length vectors, we have a set of $\frac{B}{2}+1$ independent complex numbers for each query term. We can combine these vectors using either the dot product or

the magnitude×phase precision method.

A.1 Dot product

This method treats each set of frequency components as though it is a separate index. The dot product (of the complex values) is performed and the results are summed to get a document score. The score vector contains elements:

$$s_{d,b} = \left| \sum_{t \in T} \nu_{d,t,b} \right| \tag{6}$$

where $\nu_{d,t,b}$ is the complex value of frequency component b of word t in document d.

The motivation behind this idea is to give a high score to the documents which contain all similar word spectrums for all terms in the query. A higher score will also be given to the documents which contain more occurrences of the query terms (which would lead to higher magnitude of the spectral components).

A.2 Magnitude×Phase precision

If we treat the magnitude and phase information of the word signal spectrum as separate entities, we can design another method of combining the word spectrums which uses the phase information as a weighting term. The score vector will contain the elements:

$$s_{d,b} = \Phi_{d,b} \sum_{t \in T} H_{d,t,b} \tag{7}$$

where the magnitudes ($H_{d,t,b}$) of each component in each word signal are added obtain the magnitude components of the query terms. The phase precision ($\Phi_{d,b}$) is a measure of similarity of the phase of a component across a set of terms. If the phase precision is 1 for a particular component, this implies that every term has the same phase for that component.

Some variations on calculating the phase precision have been included in this experiment. These are as follows:

Phase precision is calculated by taking a set of unit vectors with phases equal to to the phases included in the term vector and finding the average phase. The phase precision is the magnitude of this average.

$$\text{Phase precision} := \Phi_{d,b} = \left| \frac{\sum_{t \in T} \phi_{d,t,b}}{\#(T)} \right| \tag{8}$$

where

$$\nu_{d,t,b} = H_{d,t,b} \exp\left(i\theta_{d,t,b}\right) \tag{9}$$

$$\phi_{d,t,b} = \frac{\nu_{d,t,b}}{|\nu_{d,t,b}|} = \exp\left(i\theta_{d,t,b}\right) \tag{10}$$

and $\theta_{d,t,b}$ is the phase of the bth frequency component of the tth query term in the dth document $\phi_{d,t,b}$ is the unit phase of the bth component of the tth query term in the dth document, $\#(\cdot)$ is a function which gives the cardinality of the set $\cdot$, and T is the set of query terms.

For example, if we have the following three word signal spectrums:

	Frequency Component (magnitude, phase)				
word	0	1	2	3	4
michelle	$(4.0, 0)$	$(2.7, -2.2)$	$(1.4, 0.7)$	$(2.1, -1.7)$	$(2, 3.1)$
cat	$(0, 0)$	$(0, 0)$	$(0, 0)$	$(0, 0)$	$(0, 0)$
phoebe	$(3.0, 0)$	$(1.0, -2.3)$	$(1.0, -1.6)$	$(1.0, -0.8)$	$(3.0, 3.1)$
$\sum_{t \in T} H_{d,t,b}$	7.0	3.7	2.4	3.1	5.0
$\Phi_{d,b}$	1.0	0.5	0.6	0.8	0.3
$s_{d,b}$	7.0	1.9	1.4	2.4	1.7

Active phase precision is similar to phase precision except it only includes the frequency components which have a non zero magnitude. If a vector has zero magnitude, the phase is considered irrelevant, therefore these components are excluded from the calculations.

$$\begin{aligned} \text{Active phase precision} &:= \hat{\Phi}_{d,b} \\ &= \left| \frac{\sum_{t \in T : H_{d,t,b} \neq 0} \phi_{d,t,b}}{\#(\hat{T}_{d,b})} \right| \end{aligned} \tag{11}$$

where $\hat{T}_{d,b}$ is the set of query terms which do not have zero magnitude in frequency component b in document d.

Using the same query term spectrums as in the phase precision example, we can show an example calculation of active phase precision:

	Frequency Component				
	0	1	2	3	4
$\hat{\Phi}_{d,b}$	1.0	1.0	0.4	0.9	1.0
$s_{d,b}$	7.0	3.7	0.9	2.8	5.0

Selective phase precision is again similar to phase precision but only includes the frequency components with nonzero magnitude and averages over the total number of query terms. This total averaging is performed so that if any query terms do not have a frequency component b, then the score will be reduced.

$$\begin{aligned} \text{Selective phase precision} &:= \bar{\Phi}_{d,b} \\ &= \left| \frac{\sum_{t \in T : H_{d,t,b} \neq 0} \phi_{d,t,b}}{\#(T)} \right| \end{aligned} \tag{12}$$

Using the same query term spectrums as in the phase precision example, we can show an example calculation of selective phase precision:

	Frequency Component				
	0	1	2	3	4
$\bar{\Phi}_{d,b}$	0.7	0.7	0.3	0.6	0.7
$s_{d,b}$	4.7	2.5	0.6	1.9	3.3

B. Combining spectral components

Once the word spectrums have been combined into score vectors, we are left with a vector with $B/2+1$ elements, containing how relevant each query term was to that frequency component. This section will explain a few methods to help us obtain a single score from this vector and allow us to rank a set of documents in terms of relevance to the query terms. Methods which were experimented with were:

- Sum all components
- Sum largest score vector elements
- Sum largest phase precision components
- Sum largest magnitude components
- Sum threshold phase precision components

B.1 Sum all components

Since each element of the score vector represents the content of the query terms in that specific frequency component, a document which contains high values in each element should be more relevant than one that contains lower values. Therefore the most obvious operation to perform is summation.

$$S_d = \sum_{b=1}^{B/2+1} s_{d,b} \tag{13}$$

B.2 Sum largest score vector elements

It could also be stated that if there are high values contained in any of the elements of the score vector then the the document should be considered relevant the the query terms. To implement this, only the components with the two greatest values were added to create the score.

$$S_d = s_{d,b_1} + s_{d,b_2} \tag{14}$$

$$s_{d,b_1}, s_{d,b_2} \geq \max_{\forall b \neq b_1,b_2} (s_{d,b})$$

The larger of the two components will be the zeroth component (DC component). The DC component magnitude of a word signal spectrum will be equal to the sum of all bins of the word signal (and hence the value used in vector space measures). This largest component also is a real value and therefore has zero phase (leading to a precision of 1 if all words are present). This is explained in detain in section IV.

B.3 Sum largest phase precision components

Rather than finding elements of the score vector which have high values, we might want to find the elements which have the largest phase precision component. Components which have high phase precision will identify a sinusoidal pattern of all of the query terms appearing together. Again the component with the largest phase precision will be the zeroth component (as explained in section IV). For this reason, the two components with the highest phase precision will be summed to give the score. The score equation will be as in equation 14, but the components b_1 and b_2 are chosen according to the following equation:

$$\Phi^{\star}_{d,b_1}, \Phi^{\star}_{d,b_2} \geq \max_{\forall b \neq b_1,b_2} (\Phi^{\star}_{d,b})$$

where $\Phi^{\star}_{d,b}$ represents which ever of the three phase precision methods we choose.

B.4 Sum largest magnitude components

Another score calculation method is to include only the score vector elements with larger magnitude components. The magnitude of the frequency components of the query term vectors gives us some understanding of how many query terms appeared in the pattern of the sinusoidal wave (to the corresponding frequency component). As in the last scoring method, the components with the two greatest magnitudes will be taken to create the score for each document. Again, the component with the greatest magnitude will be the zeroth component.

$$H_{d,b_1}, H_{d,b_2} \geq \max_{\forall b \neq b_1, b_2} (H_{d,b})$$

The score components added will be s_{b_1} and s_{b_2}.

B.5 Sum threshold phase precision components

The final score calculation method is to include only the score vector elements with a phase precision greater than some predefined constant. The idea behind this method is that the components with low phase precision will only disturb the quality of the final score, if the phase precision is low, it must mean that the set of query terms were not in phase for that component, and is therefore considered noise.

$$S_d = \sum_{b \in \{c | \Phi^{\star}_{d,c} > P\}} s_{d,b}$$

IV. DC Analysis

If we examine the DFT equation (5), we can see that if $\beta = 0$ the transform reduces to:

$$\nu_{d,t,0} = \sum_{b=0}^{B-1} \omega_{d,t,b} \tag{15}$$

So we can clearly see that the zeroth component is equal to the sum of all the spatial bins. If no weighting is performed this zeroth component is also equal to the sum of the word count in each bin, which is the document word count (the value used in vector space measures). This implies that the vector space similarity measures are a sub-method of FDS due to them using only a subset of the data available. We will see later that FDS produces better results than the vector space similarity measures because FDS uses the full spectrum of data, rather than just the zeroth (DC) component.

We can also see that if the spatial signal $(\omega_{d,t,b})$ is real, the zeroth component is also real giving it a phase of zero. We want to show that the zeroth element of the score vector is the largest, but before we can show this we must observe the following lemmas:

Lemma 1: The magnitude of the zeroth frequency component of a real positive signal will be greater than or equal to every other component.

Proof:

$$\begin{aligned}
|\nu_{d,t,\beta}| &= \left| \sum_{b=0}^{B-1} \omega_{d,t,b} \exp\left(\frac{-i2\pi\beta b}{B}\right) \right| \\
&\leq \sum_{b=0}^{B-1} \left| \omega_{d,t,b} \exp\left(\frac{-i2\pi\beta b}{B}\right) \right| \\
&\leq \sum_{b=0}^{B-1} \omega_{d,t,b} \\
&\leq \nu_{d,t,0}
\end{aligned}$$

■

Lemma 2: The phase precision of the zeroth frequency component for any set of query terms will be greater than or equal to every other component.

Proof: From the definition of the unit phase, we can see that:

$$|\phi_{d,t,\beta}| = 1$$

and from the definition of the Fourier transform, we know that:

$$\phi_{d,t,0} = 1$$

We will look first at the case of phase precision:

$$\begin{aligned}\Phi_{d,\beta} &= \left|\frac{\sum_{t\in T}\phi_{d,t,\beta}}{\#(T)}\right| \\ &\leq \frac{\sum_{t\in T}|\phi_{d,t,\beta}|}{\#(T)} \\ &\leq \frac{\sum_{t\in T}1}{\#(T)} = 1\end{aligned}$$

We can also see that in the case of $\beta = 0$, $\Phi_{d,0} = 1$. Therefore, $\Phi_{d,\beta} \leq \Phi_{d,0}$.

In the case of active phase precision:

$$\begin{aligned}\hat{\Phi}_{d,b} &= \left|\frac{\sum_{t\in T:H_{d,t,b}\neq 0}\phi_{d,t,b}}{\#(\hat{T}_{d,b})}\right| \\ &\leq \frac{\sum_{t\in T:H_{d,t,b}\neq 0}|\phi_{d,t,b}|}{\#(\hat{T}_{d,b})} = 1\end{aligned}$$

by definition of $\hat{T}_{d,b}$. By observing the case of $\beta = 0$ we can see that $\hat{\Phi}_{d,0} = 1$.

And finally, selective phase precision:

$$\bar{\Phi}_{d,b} = \left|\frac{\sum_{t\in T:H_{d,t,b}\neq 0}\phi_{d,t,b}}{\#(T)}\right|$$

$$\phi_{d,t,0} = \begin{cases} 1 & \text{if query term } t \text{ exists in document } d \\ 0 & \text{otherwise} \end{cases}$$

Here we have to use the fact shown in Lemma 1. If the magnitude of the zeroth frequency component is greater than or equal to every other component and the zeroth component is 0, this implies all component are equal to 0 and hence have no phase. Once this is understood, we can follow the same process as the last two phase precision methods to complete the proof. ■

This leads to the following theorem:

Theorem 1: The zeroth element in the score vector will be greater than or equal to every other element in the score vector.

$$s_{d,0} = \max_{\forall\beta}(s_{d,\beta}) \tag{16}$$

Proof: To begin the proof we must examine the cases which can occur. This paper covers two simple methods to combine the word spectrum into a score vector, namely the dot product and magnitude×phase precision.

In the case where the word spectrums are combined using the dot product (Section A.1):

$$\begin{aligned} s_{d,b} &= \left| \sum_{t \in T} \nu_{d,t,b} \right| \\ &\leq \sum_{t \in T} |\nu_{d,t,\beta}| \\ &\leq \sum_{t \in T} \nu_{d,t,0} = s_{d,0} \end{aligned}$$

In the case where the word spectrums are combined using magnitude×phase precision (Section A.2):

$$\begin{aligned} s_{d,b} &= \Phi_{d,b} \sum_{t \in T} H_{d,t,\beta} \\ &= \Phi_{d,b} \sum_{t \in T} |\nu_{d,t,\beta}| \\ &\leq \Phi_{d,b} \sum_{t \in T} \nu_{d,t,0} \\ &\leq \Phi_{d,0} \sum_{t \in T} \nu_{d,t,0} = s_{d,0} \end{aligned}$$

It can be seen that this proof applies for all three cases of phase precision by applying Lemma 2. ■

This theorem shows us that the DC component is the most dominant component when calculating the score. We have also seen that the DC component is used to calculate the relevance in vector space similarity measures. Therefore the smaller AC components (used only in FDS) can be considered as a refinement to the score in relation to the positions of the query terms.

V. AC Analysis

Given that the DC component of a word signal spectrum is the most dominant component, how do the rest of the components affect the score? All of the methods are based on the idea that if the words appear in phase, then the document should be more relevant than a document out of phase. This can be easily seen for a few query terms, but what about when there are many query terms (as in the TREC query shown in figure 4).

A. Case 1 : Dot Product

Since the DC component is the largest component and always has zero phase, any other component will also have to have a common phase across the set of query terms in order to be comparable to the DC component. For a component to have common phase across the entire set of query terms, the query terms must occur in the same positional bins. If the number of query terms is greater than the number of words per bin, the query terms have no chance of appearing in the same bins. For each new query term added, the commonality between phases will reduce. Therefore if the number of query terms is much greater than the number of words per bin, then the AC components will be insignificant when compared to the DC component.

B. Case 2 : Magnitude×Phase precision

If we look at the basic expression for magnitude×phase precision (equation 7), we can see that the phase precision acts as a weighting term. If we consider component b (not equal to 0), we can see that the phase precision will remain at value 1 as long as all terms in the query are in the same positional bins in the document. Once the number of query terms passes the number of words per bin, it is impossible for all words in the query appear in the same positional bins. Therefore the phase precision will reduce with every new word added to the query term set. Therefore:

$$\#(T) \gg B \Rightarrow \Phi_{d,b} < \epsilon \quad \forall b \in \{1, 2, \ldots, B-1\} \tag{17}$$

where ϵ is a small and positive. This implies that if a large number of query terms are used, the DC component will be the dominant factor in the score.

$$\#(T) \to \infty \Rightarrow S_d \to s_{d,0} \tag{18}$$

We have also found in the DC analysis (section IV) the DC component is related to the value used in vector space similarity measures. Based on these two observations, it follows that if the number of query terms is large, FDS will be comparable to the similarity measure using the weighting method chosen. In our case, we have chosen TF×IDF weighting as the preweighting scheme for FDS. Therefore, for a large number of query terms FDS should approach the cosine measure with TF×IDF weighting.

VI. Experiments

Initial results have shown that FDS improves the accuracy of search results on small document sets (containing about 100 to 1000 documents) [15]. In these experiments we will show that FDS can effectively improve the results of large document sets. By doing so, we will also show that using FDS in a Web search engine would be a great enhancement.

A. Methods Performed

Before we can begin comparing FDS to the existing method, we must perform experiments to find which of the combinations of the three steps shown will perform the best. We should see that the results will be dependent on the data (document set and queries) but we should be able to see which method gives overall better results. The methods chosen to compare in this experiment are shown in table I.

B. Experiment 1 : Long queries

The document sets used were part of the TIPSTER project and have now been taken over by TREC. The TIPSTER documents come from many sources and are split on to several disks to allow the users to compare specific sets. The documents used in these experiments are from the Associated Press Newswire (1988) disk 2 (AP2). This information can be found at the TREC web site [17]. Each document in the set is separated by SGML tags and comes in the form shown in figure 3. The AP2 collection contains approximately 80,000 news articles.

Before the documents were used, preprocessing was performed. This preprocessing consisted of case folding, removing stop words (the stop word list contained about 400 common English words), and stemming was performed using Porter's stemming algorithm. The FDS methods used eight bins to record the approximate position of the words throughout the document. The number of bins chosen gives a trade off between accuracy and speed. For example if only four bins were chosen, the results would be less accurate, but the processing time would be faster. Trials were run with different numbers of bins and eight seemed to give quick and accurate results.

TABLE I
METHODS CHOSEN TO COMPARE IN THE QUEST TO FIND THE MOST ACCURATE FORM OF FDS

Method	Weighting	Combine word spectrum	Combine spectral components
3.1.1	TBF×IDF	dot product	add all components
3.2.1	TBF×IDF	magnitude×phase precision	add all components
3.2.2	TBF×IDF	magnitude×phase precision	add largest phase precision components
3.3.1	TBF×IDF	magnitude×active phase precision	add all components
4.1.1	PTF×IDF	dot product	add all components
4.2.1	PTF×IDF	magnitude×phase precision	add all components
4.2.2	PTF×IDF	magnitude×phase precision	add largest phase precision components
4.3.1	PTF×IDF	magnitude×active phase precision	add all components
3.4.1	TBF×IDF	magnitude×selective phase precision	add all components
3.4.2	TBF×IDF	magnitude×selective phase precision	add largest phase precision components
3.4.4	TBF×IDF	magnitude×selective phase precision	add largest score components
4.4.1	PTF×IDF	magnitude×selective phase precision	add all components
4.3.2	PTF×IDF	magnitude×active phase precision	add largest phase precision components
4.3.3	PTF×IDF	magnitude×active phase precision	add largest magnitude components
4.3.4	PTF×IDF	magnitude×active phase precision	add largest score components
3.3.2	TBF×IDF	magnitude×active phase precision	add largest phase precision components
3.3.3	TBF×IDF	magnitude×active phase precision	add largest magnitude components
3.3.4	TBF×IDF	magnitude×active phase precision	add largest score components
3.4.5	TBF×IDF	magnitude×selective phase precision	add threshold phase precision components
4.4.5	PTF×IDF	magnitude×selective phase precision	add threshold phase precision components

The queries used were those corresponding to the ad-hoc tasks in the TREC-1 (queries 51 to 100), TREC-2 (queries 101 to 150) and TREC-3 (queries 151 to 200) conferences. These can also be found on the TREC web site [17]. A typical query is shown in figure 4. As we can see by the sample query shown, a typical TIPSTER query is not equivalent to a typical Web search engine query. These queries were used because there are lists of document classifications for each of the queries. After each of the TREC conferences the top ranking documents were examined by judges and given a classification of relevant or irrelevant, to create a relevance file for others to use. The queries were applied to the document sets to give an understanding of which FDS methods were effective and which were not. Each method generates a list of the documents in order of relevance. The precision[1] and recall[2] values were generated from the ranked list by the `trec_eval` program (set up by the organisers of TREC). The results are located in table II. One of the stronger methods can be seen compared with existing vector space similarity measures in figure 5. The TF×IDF method is compared to the FDS method to see the effect of examining the spatial information found in the documents (since FDS 3-4-1 uses a variant of TF×IDF as a preweighting scheme). The BD-ACI-BCA method is compared because it is suggested by Zobel and Moffat [18] that this method is generally the best of the vector space similarity measures.

The results show us that the methods $3.4.x$ and $4.3.2$ are superior amongst the FDS methods for this document set. These are the methods which use TBF×IDF weighting and magnitude × selective phase

[1] Precision is the proportion of relevant documents to total documents retrieved.
[2] Recall is the proportion of relevant documents retrieved to the total number of relevant documents.

```
<DOC>
<DOCNO> AP880212-0001 </DOCNO>
<FILEID>AP-NR-02-12-88 2344EST</FILEID>
<FIRST>u i AM-Vietnam-Amnesty 02-12 0398
</FIRST><SECOND>AM-Vietnam-Amnesty,0411
</SECOND><HEAD>Reports Former Saigon
Officials Released from Re-education Camp
</HEAD><DATELINE>BANGKOK, Thailand (AP)
</DATELINE>
<TEXT>
   More than 150 former officers of the
overthrown South Vietnamese government
have been released from a re-education
camp after 13 years of detention, the
official Vietnam News Agency reported
Saturday.
   The report from Hanoi, monitored in
Bangkok, did not give

                                    ⋮

Vietnamese capital of Saigon.
   The amnesties apparently are part of
efforts by Communist Party chief Nguyen
Van Linh to heal internal divisions and
improve Vietnam's image abroad.
</TEXT>
</DOC>
```

Fig. 3. A typical document from the TIPSTER document set

precision, and the method which uses PTF×IDF weighting and magnitude × active phase precision taking only the maximum phase precision values.

Also we can see in figure 5, the FDS method gives a greater precision for all values of recall over the cosine TF×IDF measure, but the results are very similar to the BD-ACI-BCA method. This is probably due to the effects of large query term sets as discussed in section V.

C. Experiment 2 : Short queries

Now that we have observed which FDS methods produce more precise results for long queries, we can compare them with the existing vector space methods (the cosine measure) for smaller query term sets. This comparison will use queries which are more like those which are found on Web search engines (a few key words, eg. "Weather Related Fatalities") . To generate the queries, the titles of the TIPSTER queries will be extracted and used as queries themselves.

Since the queries have changed, it implies that the relevance of each document has changed as well. Therefore each document must be reclassified for each query. We make the assumption that most people using a Web search engine will not look past the first twenty results (the first two pages), and so only

```
<top>
<head> Tipster Topic Description
<num> Number:  059
<dom> Domain:  Environment
<title> Topic:  Weather Related Fatalities
<desc> Description:
Document will report a type of weather
event which has directly caused at least
one fatality in some location.
<smry> Summary:
Document will report a type of weather
event which has directly caused at least
one fatality in some location.
<narr> Narrative:
A relevant document will include the number
of people killed and injured by the weather
event, as well as reporting the type of
weather event and the location of the
event.
<con> Concept(s):
1.  lightning, avalanche, tornado, typhoon,
                                    ⋮
4.  NOT earthquakes, NOT volcanic eruptions
<fac> Factor(s):
<def> Definition(s):
</top>
```

Fig. 4. A sample query from the TIPSTER collection

manually classify the top twenty document of each query. To eliminate any bias in the results (and to speed up the classification process), a small program was written to take the top twenty documents of each method, remove any of the duplicates and sort them in alphabetical order of document number. Each document was then manually classified according to the query, and stored in a relevance file.

The document set used was the AP2 set (as in the previous experiment), since this was the most similar to actual web data. The preprocessing which was performed in the experiment in section A was also performed here. The results can be seen in figure 6 and table III. Note that only the cosine TF$\times$IDF results are shown in comparison to FDS 3-4-1. The short query experiment was performed for the BD-ACI-BCA method, but the results were either similar to or worse than that of the shown cosine TF$\times$IDF measure. Figure 6 shows us the positions of the relevant documents (represented by a black block) amongst the top 20 documents considered relevant by each method. For high precision we want all of the black blocks to be higher (closer to rank 1) in the order.

We can clearly see that the FDS method produces more relevant documents in the top 20 results of each search and that the relevant documents appear higher in the order. We can also see in figure 6 that FDS produces a relevant document as the highest ranked document in all but one case. The same figure

TABLE II

ALL FDS METHODS TESTED WITH LONG QUERIES. SHOWN ARE THEIR PRECISION VALUES AT RECALLS OF 0%, 10%, 20%, 30% AND 40% AND THEIR AVERAGE PRECISION AND R-PRECISION VALUES.

Method	Precision at Recall					Avg. Prec.	R-Prec.
	00%	10%	20%	30%	40%		
3.1.1	0.4682	0.2815	0.2025	0.1489	0.1093	0.1100	0.1586
3.2.1	0.6707	0.5748	0.5230	0.4477	0.3999	0.3190	0.3466
3.2.2	0.6836	0.5783	0.5192	0.4410	0.3900	0.3061	0.3397
3.3.1	0.6626	0.5859	0.5197	0.4491	0.3967	0.3208	0.3462
4.1.1	0.3956	0.1920	0.1263	0.0863	0.0572	0.0655	0.1101
4.2.1	0.5923	0.4762	0.4089	0.3395	0.3035	0.2570	0.2987
4.2.2	0.6764	0.5645	0.5023	0.4290	0.3820	0.3131	0.3367
4.3.1	0.6497	0.5460	0.4680	0.4102	0.3711	0.3116	0.3362
3.4.1	0.7400	0.6453	0.5762	0.5201	0.4626	0.3814	0.3991
3.4.2	0.7161	0.6196	0.5523	0.4967	0.4454	0.3512	0.3737
3.4.4	0.7279	0.6303	0.5627	0.5163	0.4531	0.3659	0.3832
4.4.1	0.6958	0.5686	0.4993	0.4396	0.3990	0.3300	0.3576
4.3.2	0.7174	0.6195	0.5574	0.4922	0.4495	0.3661	0.3834
4.3.3	0.6542	0.5510	0.4934	0.4300	0.3889	0.3193	0.3408
4.3.4	0.6542	0.5510	0.4934	0.4300	0.3889	0.3193	0.3408
3.3.2	0.6623	0.5860	0.5211	0.4483	0.3990	0.3086	0.3448
3.3.3	0.6648	0.5722	0.5036	0.4394	0.3898	0.3040	0.3395
3.3.4	0.6697	0.5771	0.5125	0.4429	0.3917	0.3109	0.3429
3.4.5	0.7069	0.6115	0.5494	0.4890	0.4356	0.3527	0.3808
4.4.5	0.6575	0.5167	0.4591	0.4087	0.3557	0.2995	0.3350

TABLE III

SHORT QUERIES APPLIED TO THE AP2 DOCUMENT SET. WE CAN SEE THAT THE FDS METHOD GIVES MORE RELEVANT DOCUMENTS OUT OF THE TOP 20 RETURNED BY EACH METHOD.

Query term	Relevant documents in top 20	
	Cosine TF×IDF	FDS 3.4.1
Airbus Subsidies	10	14
Satellite Launch Contracts	6	12
Rail Strikes	12	17
Weather Related Fatalities	3	11
Information Retrieval Systems	5	7
Attempts to Revive the SALT II Treaty	5	12
Bank Failures	16	18
U.S. Army Acquisition of Advanced Weapons Systems	2	4
International Military Equipment Sales	6	11
Fiber Optics Equipment Manufacturers	6	8
Total	71	114

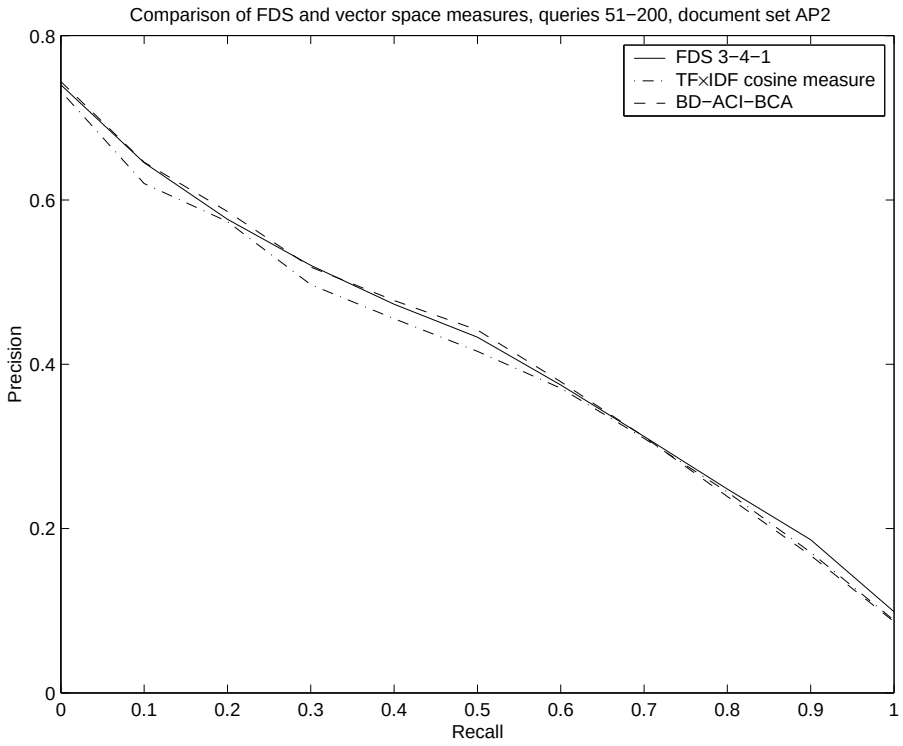

Fig. 5. Plot of precision vs recall which compares an FDS method (3.4.1) with the TF×IDF cosine measure and BD-ACI-BCA using query sets 51 to 200 on data set AP2. We can see that the FDS method has a higher precision for every value of recall.

shows that FDS has a higher relative precision than the cosine measure using TF×IDF weighting for web-like queries. Table III shows that on average FDS produces just over four more documents (a 60% improvement) in the top twenty results when compared to the cosine measure. This implies that FDS would greatly improve the quality of a Web search engine by giving the users more relevant documents to choose from.

VII. Conclusion

Internet search engines have become an essential tool for locating resources and information on the Web. Due to the large majority of Web information being textual, a need arises for a superior text search engine. We have presented a novel and simple document ranking method called Fourier Domain Scoring (FDS) which provides more precise results than the currently used vector space similarity methods. We also showed that the existing vector space similarity methods can be considered as a special case of FDS. The results show that FDS is a superior method because it makes use of the spatial information within a document rather than the count of each query term.

Results were given for two different experiments, the first involving long queries and the second using shorter queries. Both experiments were performed on the TREC data set. We have shown that FDS gives a higher precision for all values of recall when compared to the cosine measure for the long queries. When considering the shorter queries, we looked at the top 20 ranked documents from each method and compared the results. The top 20 were selected due to the limited patience of most Web search engine users. FDS improves performance on average 60% better than the cosine measure when observing the top 20 ranked documents. This implies that FDS would enhance the performance of a Web search engine if

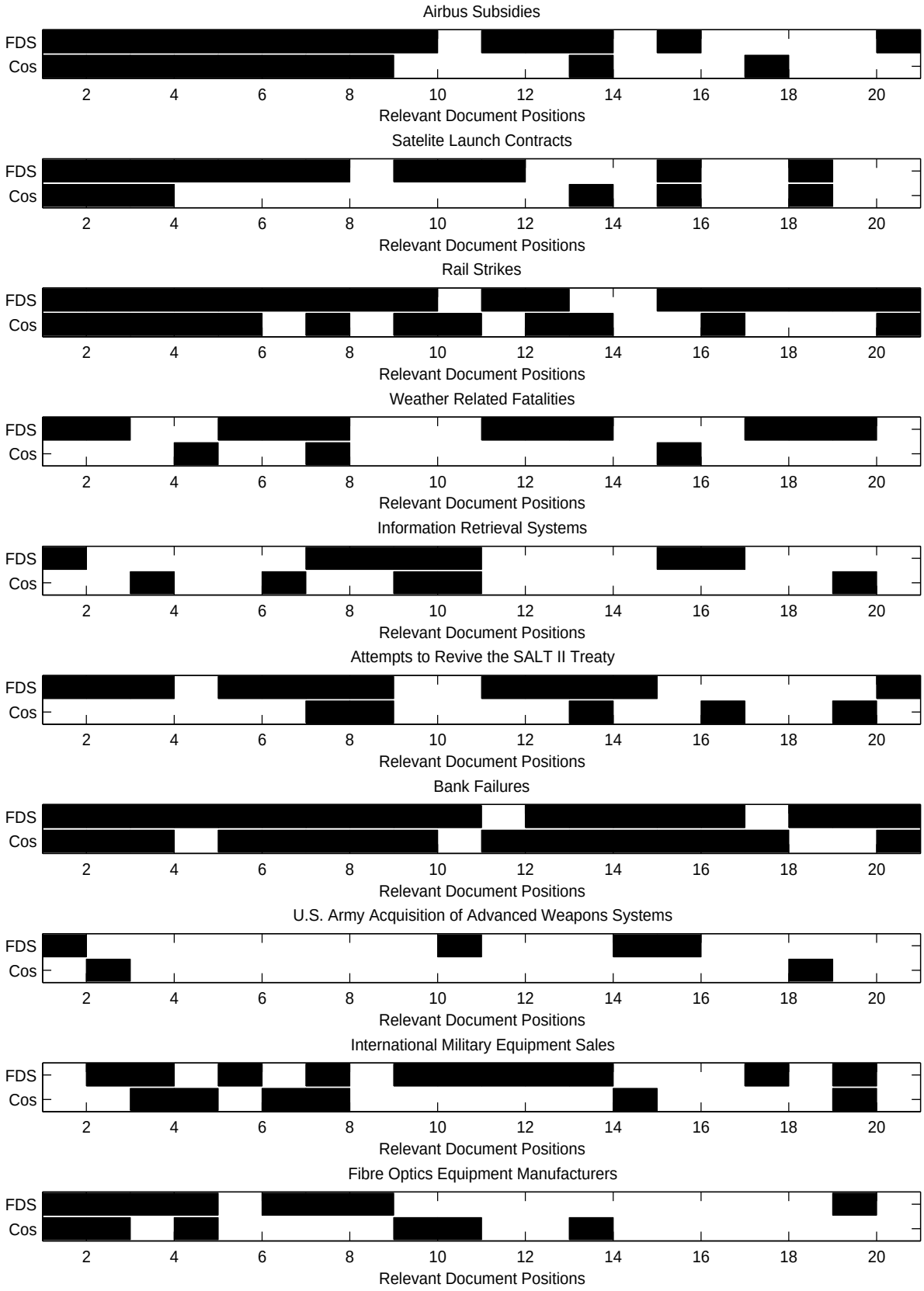

Fig. 6. The positions of relevant documents in the top 20 documents ranked by FDS and the TF$\times$IDF cosine measure. A relevant document is identified by a black block. FDS has more relevant documents for all queries, and the relevant documents are higher in the order when compared to the cosine measure.

used in the place of the cosine measure as the text classification method.

ACKNOWLEDGEMENTS

We would like to thank the ARC Special Research Centre for Ultra-Broadband Information Networks for their support and funding of this research. We would also like to thank Alistair Moffat and his research team for use of their resources and help.

REFERENCES

[1] Massimo Marchiori, "The quest for correct information on the web: hyper search engines," *Computer Networks and ISDN Systems*, vol. 29, pp. 1225–1235, 1997.

[2] 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*, vol. 29, pp. 1249–1255, 1997.

[3] S. Jeromy Carrière and Rick Kazman, "Webquery: searching and visualising the web through connectivity," *Computer Networks and ISDN Systems*, vol. 29, pp. 1257–1267, 1997.

[4] Ellen Spertus, "Parasite: mining structural information on the web," *Computer Networks and ISDN Systems*, vol. 29, pp. 1205–1215, 1997.

[5] Oren Etzioni, "Moving up the information food chain," in *AI Magazine*, vol. 18, pp. 11–18. American Association for Artificial Intelligence, Summer 1997.

[6] Ian H. Witten, Alistair Moffat, and Timothy C. Bell, *Managing gigabytes : compressing and indexing documents and images*, Morgan Kaufmann Publishers, 1999.

[7] Chris Buckley and Janet Walz, "Smart in trec 8," in *NIST Special Publication 500-246: The Eighth Text REtrieval Conference (TREC-8)*, E. M. Voorhees and D. K. Harman, Eds., November 1999, pp. 577–582.

[8] Chris Buckley, Amit Singhal, Mandar Mitra, and Gerard Salton, "New retrieval approaches using smart : Trec 4," in *NIST Special Publication 500-236: The Fourth Text REtrieval Conference (TREC-4)*, D. K. Harman, Ed., November 1995, pp. 25–48.

[9] S. E. Robertson and S. Walker, "Okapi/keenbow at trec-8," in *NIST Special Publication 500-246: The Eighth Text REtrieval Conference (TREC-8)*, E. M. Voorhees and D. K. Harman, Eds., November 1999, pp. 151–162.

[10] Kiduk Yang and Kelly Maglaughlin, "Iris at trec-8," in *NIST Special Publication 500-246: The Eighth Text REtrieval Conference (TREC-8)*, E. M. Voorhees and D. K. Harman, Eds., November 1999, pp. 645–656.

[11] James Allan, Jamie Callan, Fang-Fang Feng, and Daniella Malin, "Inquery and trec-8," in *NIST Special Publication 500-246: The Eighth Text REtrieval Conference (TREC-8)*, E. M. Voorhees and D. K. Harman, Eds., November 1999, pp. 637–644.

[12] Sergey Brin and Lawrence Page, "The anatomy of a large-scale hypertextual Web search engine," *Computer Networks and ISDN Systems*, vol. 30, no. 1–7, pp. 107–117, 1998.

[13] David S. Ebert, Amen Zwa, and Ethan L. Miller, "Two-handed volumetric document corpus mangement," *IEEE Computer Graphics and Applications*, vol. 17, no. 4, pp. 60–62, July/August 1997.

[14] M. W. Berry, S. T. Dumais, and G. W. O'Brien, "Using linear algebra for intelligent information retrieval," Tech. Rep., Computer Science Department, The University of Tennessee, Knoxville, TN, December 1994.

[15] Laurence A. F. Park, Marimuthu Palaniswami, and Ramamohanarao Kotagiri, "Internet document filtering using fourier domain scoring," in *Principles of Data Mining and Knowledge Discovery*,

Luc de Raedt and Arno Siebes, Eds. September 2001, number 2168 in Lecture Notes in Artificial Intelligence, pp. 362–373, Springer-Verlag.

[16] Susan T. Dumais, "Improving the retrieval of information from external sources," *Behaviour Research Methods, Instruments & Computers*, vol. 23, no. 2, pp. 229–236, 1991.

[17] National Institute Of Standards and Technology, "Text retrieval conference (trec) http://trec.nist.gov/," World Wide Web, 2001.

[18] Justin Zobel and Alistair Moffat, "Exploring the similarity space," in *ACM SIGIR Forum*, Spring 1998, vol. 32, pp. 18–34.