Feature Space Approximation of Document Edit Distances: An Application in Document Versioning

Wei Lee Woon
K. Daniel Wong

Technical Report DNA #2012-06

April 2012

Data & Network Analytics Research Group (DNA)
Computing and Information Science Program,
Masdar Institute of Science and Technology,
PO Box 54224, Abu Dhabi, UAE.

Feature space approximations of document edit distances: an application in document versioning

W.L.Woon*, K.S.D.Wong†
*Masdar Institute of Science and Technology,
P.O. Box 54224 Abu Dhabi, United Arab Emirates.
†Daniel Wireless LLC, PO Box 20181,
Stanford, CA 94309-0181, USA.
wwoon@masdar.ac.ae, ksdwong@gmail.com

***Abstract*— The automated analysis of documents is an important task given the rapid increase in availability of digital texts. Traditionally, text documents are encoded as vectors of term occurrence frequencies, a representation which facilitates the classification and clustering of documents. In an earlier publication, we had presented a framework where the edit distances between documents was used to reconstruct the version history of a set of documents. However, one problem which we encountered was the high computational costs of calculating these edit distances. In addition, the number of document comparisons which need to be done scales quadratically with the number of documents. In this paper we propose a simple approximation which retains many of the benefits of the method, but which greatly reduces the time required to calculate these edit distances. To test the utility of this method, the accuracy of the results obtained using this approximation is compared to the original results.**

***Index Terms*— String Matching, Text Processing, Data Mining, Versioning, Information Retrieval**

I. Introduction

The proliferation of network connectivity and web content have led to rapid increases in the quantity and availability of digital texts, motivating the development of new tools to cope with this sudden increase. However, document mining research has focused on the data-mining derived operations of classification (Tan, 2006; Lan et al., 2005; Liu and Lu, 2002; Zaiane and Antonie, 2002) (and the related operations of search/retrieval) and clustering (Hung and Wermter, 2004; Muthukrishnan, 2002; Larsen and Aone, 1999). While these are important capabilities, document spaces do not always match the clustered model typical of data mining applications.

The instance of a set of interrelated documents or source code versions provides an example where the documents do not occur in clusters but rather fall along a gradual continuum. In an earlier work (Woon and Wong, 2008), we demonstrated a procedure for inferring the version histories of unlabelled document collections using the edit distances between these documents. A number of experiments were conducted which indicated that the method was able to recover the version histories of the collections to a high level of accuracy.

However, one problem was that the calculation of the edit distances was computationally expensive. In this paper, we present a procedure for approximating inter-document distances without calculating the edit distances between all pairs of documents.

A. Related work

The underlying theme in our work in this area (Woon and Wong, 2008; Woon and Wong, 2006) has been the the study of alternative approaches to document analysis; this is in contrast to the standard "vector space" model (VSM), also known as the bag-of-words representation, in reference to traditional document representations which do not utilize the position of words within documents. This approach will be discussed at greater length in section II but in particular, we are interested in representations which exploit the *distances* between pairs of documents, as exemplified by the edit distance measure which forms the basis of this study.

Alternative text representations have been presented in the past. The idea of using string alignment to study documents is related to the *Levenshtein's distance* between documents (Levenshtein, 1966); however, this method uses alignments at the character level, and is more appropriate for lower-level applications, for example in the study of text entry methods for mobile phones (Soukoreff and Mackenzie, 2001). Our method operates at the level of words, which we believe represent the appropriate level of resolution for the majority of text documents.

Another similar approach in terms of document representation is the string kernel technique (Lodhi et al., 2000; Lodhi et al., 2002). The original study examined the use of character level kernels but similar kernels calculated at the level of words have also been proposed (Cancedda et al., 2003). Briefly, the string kernel method projects each document to a high dimensional feature space where each "dimension" is the weighted number of occurrences of a particular substring (which need not be contiguous) in a document. As the number of such substrings is potentially huge, the dimensionality of the resulting subspace is also large, motivating the use of the kernel procedure (Cristianini and Taylor, 2000). With respect to document alignments, these methods are similar to the related operation of *local* alignment, which have also been used for biological sequence analysis; in contrast, the method proposed here performs a global alignment of the documents concerned.

Apart from that, other research efforts which deviate from the standard VSM based document mining schemes include the use of contextual features of words to derive additional knowledge. The context in this case could refer to a variety of local word groupings; examples include sentence level contextual information (Pullwitt, 2002) down to consecutive words (Honkela and Hyvarinen, 2004). In (Park et al., 2005), a method for calculating the wavelet transform of term occurrences in text documents is proposed, though the resulting wavelet transform is again used for document retrieval.

One notable example of an approach that focuses on the exploratory aspect of document mining (as opposed to classification and retrieval) is the WEBSOM technique (Lagus et al., 2004); via the use of Self-Organizing-Maps (SOMs), this technique has been used to visualize web search results to help elucidate themes and subject areas. Another similar study uses hierchical feature maps to decompose text into natural divisions such as chapters and sections (Merkl, 1997) while in (Yang and Lee, 2004) a combination of SOMs and an agglomerative clustering algorithm is used to automatically organize a collection of web-documents into a coherent hierarchy.

In short, while still relatively uncommon, the study of alternative frameworks for analysing documents has already attracted significant attention. However, the challenge of extracting and fully exploiting the information encoded in the massive volumes of digital text is still a big one, and there is much scope for further progress in this area.

The rest of this paper is structured as follows: in section II, we describe the document versioning procedure, as well as the experimental procedures observed and also the evaluation metric used to evaluate the results. Section III presents the results of these experiments as well as some key observations, while section IV concludes the paper and highlights avenues for future research.

II. Automatic document versioning

A. Overview

The basic methodology used for determining document version histories is now described briefly (Woon and Wong, 2008). The overall idea is summarized in the following two steps:

1) For a set of documents $\mathcal{D}$, calculate the differences between all pairs of documents in terms of the edit distance, defined as the number of steps required to transform one of the documents to the other.
2) Once all the distances have been calculated in this way, the shortest path connecting all the documents is found, giving an approximation of the temporally correct sequence of document versions.

B. Feature extraction

1) Edit distance: Edit distances between pairs of documents and strings are found using a dynamic programming algorithm similar to the Needleman-Wunsch method used to align biological sequences (Needleman and Wunsch, 1970). A dynamic programming matrix is used, as shown in figure 1, where x_i and y_j are elements in the two strings to be compared.

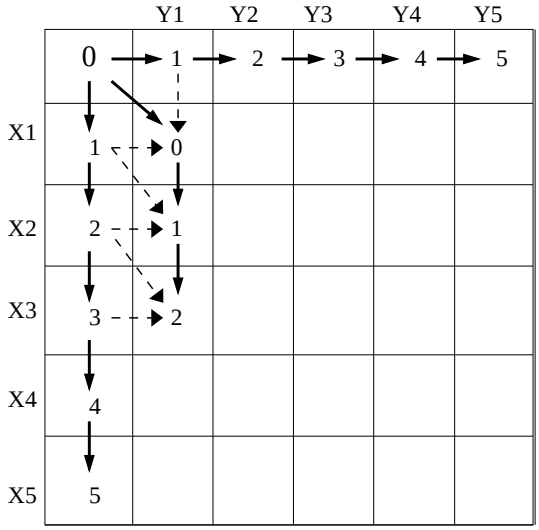

Fig. 1. An example of a dynamic programming matrix. In this example, $x_1 = y_1$,$x_2 \neq y_1$ and $x_3 \neq y_1$; the gap and substitution costs for the words are both 1. The bold arrows denote the actual paths taken to arrive at a particular cell.

As shown in the figure, the algorithm starts from the top left corner with a score of 0, and proceeds to fill out each element in that column. When it reaches the bottom, the next column is filled out and so on. Each cell is scored using the following set of rules:

$$m_{i,j} = argmax \begin{cases} m_{i-1,j-1} + \textit{match}(x_i, y_j) \\ m_{i-1,j} + \textit{gap cost} \\ m_{i,j-1} + \textit{gap cost} \end{cases}, \quad (1)$$

where $m_{i,j}$ refers to the element in the ith row and jth column. *match*(x_i, y_j) is a function which returns 0 if $x_i = y_j$, or a pre-defined substitution cost if $x_i \neq y_j$; this represents the penalty incurred for substituting a word with another word. Similarly, the *gap cost* is the penalty incurred when inserting a gap into the string. Due to the lack of any prior information about the frequencies of these events, both the costs were set to 1 for all the experiments described here.

Note that the cells on the leftmost column or top row can only be reached by the cell immediately above or to the left; as such, these cells may be filled out in advance, as is done in figure 1. The intuitive interpretation of this is that each position within the matrix corresponds to a specific alignment configuration, and could have been arrived at via any of the following three operations:

1) A gap inserted into string x
2) A gap inserted into string y
3) A substitution of x_i with y_j (or vice-versa)

The choice amongst the three alternatives is determined by which will provide the highest score after taking into consideration the pre-defined gap and substitution penalties, as well as the match scores of the adjacent matrix cells. A fuller

description of the dynamic programming algorithm may be obtained from (Mount, 2001).

2) Feature space approximation to inter-document edit distance: As a measure of similarity between pairs of documents, the edit distance was shown to be a valuable technique. However, calculating this minimum sequence of edits is an expensive operation (the time complexity is $\mathcal{O}(l_1.l_2)$, where l_1 and l_2 are the lengths of the documents being compared). A more serious problem is that adding a new document to the test set required comparisons with all other documents already in the set. This imposes a practical limit on the total number of documents which can realistically be analyzed.

To address this issue, in this paper we propose a method by which the number of document comparison required may be greatly reduced. The proposed method exploits the kernel trick which is based on Mercer's theorem (Mercer, 1909); given a symmetric, positive semi-definite measure of similarity or distance $K(,)$, the theorem states that there is a high dimensional Hilbert space $\mathcal{H}$ (also known as the feature space) such that:

$$K(x_1, x_2) = \langle \Phi_K(x_1), \Phi_K(x_2) \rangle \,, \tag{2}$$

where $x_1, x_2 \in \mathcal{X}$, the input space, Φ_K is the mapping $\mathcal{X} \rightarrow \mathcal{H}$ and $<,>$ denotes the inner-product in the feature space. Importantly, there is no need to use or express the function Φ_K explicitly; this is a critical point as the exact form of Φ_K is often unknown, and as directly calculating the inner product is often very costly.

To apply this formulation to the current problem, we premise that the edit distance measure possesses the characteristics of a Mercer kernel (Aradhye and Dorai, 2002). A theoretical proof is beyond the scope of this paper, but this intuition is supported by a number of observations. As the minimum edit path is independent of the order of the documents in $K(,)$, the symmetry condition is satisfied. Positive semi-definiteness is harder to establish but can be empirically tested by calculating the eigenvalues of the closely related similarity matrix $\mathbf{\Sigma}$:

$$\mathbf{\Sigma} = \begin{bmatrix} d_{max} - d_{1,1} & \dots & d_{max} - d_{1,n} \\ \vdots & \ddots & \vdots \\ d_{max} - d_{n,1} & \dots & d_{max} - d_{n,n} \end{bmatrix} ; \tag{3}$$

where $d_{i,j}$ is the edit distance between documents i and j, $n = |\mathcal{D}|$ and $d_{max} = \max_{ij} \{d_{i,j} : i, j \in \mathcal{D}\}$. Our observations have been that the vast majority of the eigenvalues of Σ are positive, with a number of very large positive eigenvalues. A small number of negative eigenvalues were observed but these were extremely small. As an example, for the 216×216 similarity matrix corresponding to *fork.c*, there were 2 negative eigenvalues which were 6.95×10^-1 and 2.94×10^-13. In comparison, the ten largest eigenvalues were in the range of 6.93×10^3 to 6.94×10^5. Allowing for some noise in $\mathbf{\Sigma}$, this at least gives us some confidence that the edit distance approximately matches Mercer's conditions.

Next, note that while the feature space may be very high dimensional, the collection of documents being studied would occupy only a subspace $\mathcal{S}$ of this space, and it is on this subspace that we seek to focus our efforts. As $\mathcal{H}$ is not directly observable, we instead select a subset $\mathcal{B} \subset \mathcal{D}$ for use as basis vectors in $\mathcal{H}$, and define $\tilde{\mathcal{S}}$ as the subspace spanned by these basis vectors. Provided that the choice of $\mathcal{B}$ provides a good coverage of the entire set of documents, it may be assumed that $\tilde{\mathcal{S}} \approx \mathcal{S}$.

Finally, we obtain the coordinates of a given document in $\mathcal{S}$ by projecting it onto $\mathcal{B}$ via the edit distances:

$$\boldsymbol{\mu}_i = \{d_{max} - d_{i,j} : j \in \mathcal{B}\} \,. \tag{4}$$

Distances between documents i and j can now be found quickly by calculating the Euclidean distances in $\tilde{\mathcal{S}}$ as follows:

$$d^*_{i,j} = \sum_{k=1}^{|\mathcal{B}|} \left[\boldsymbol{\mu}_i(k) - \boldsymbol{\mu}_j(k) \right]^2 . \tag{5}$$

As the computational requirements of calculating the Euclidean distance are minimal compared to those of calculating the edit distance, the savings are considerable. To obtain the full distance matrix for a corpus of n documents, a total of $\frac{1}{2}n(n-1)$ edit-distance calculations are needed. In comparison, for a given ratio $\nu = \frac{|\mathcal{B}|}{n}$, the number of edit-distance calculations needed is:

$$\begin{aligned} \text{\# Calcs.} &= \nu n \left(n - \nu n\right) + \frac{1}{2} \nu n \left(\nu n - 1\right) \\ &= \frac{\nu n}{2} \left[2n - \nu n - 1\right] \\ \therefore \text{Savings} &= 1 - \frac{\frac{\nu n}{2} \left[2n - \nu n - 1\right]}{\frac{n}{2}(n-1)} \\ &= 1 - \nu \left[\frac{2n-1}{n-1} - \frac{\nu n}{n-1} \right] \\ \text{(for } n \gg 1) &\approx 1 - \nu(2 - \nu) \end{aligned}$$

The relationship between the reduction in the number of edit distance calculations and the ratio ν is depicted in Fig. 2, in which it can be seen that a choice of ν between 0.2 and 0.4 will allow a reduction of around 50% in the number of required edit distance calculations. One last issue is the selection of the documents to be used as basis vectors where, ideally, the documents chosen should be representative of all of the documents in $\mathcal{D}$. For the current study two selection schemes were used: the first method was to sample randomly from the collection of documents, while the second approach was to select documents which were evenly spaced along the actual version history. Note that the second method is only intended to serve as a demonstration since in a real application the true version history would not be available. Therefore, an issue for future research would be to devise a clustering scheme which could be used to select documents that are representative of the underlying distribution in the document space.

3) Vector Space representations: For the sake of comparison, two conventional vector-space representations have been included in our experiments. A number of VSM-based features are available (Lan et al., 2005); we choose two common variants:

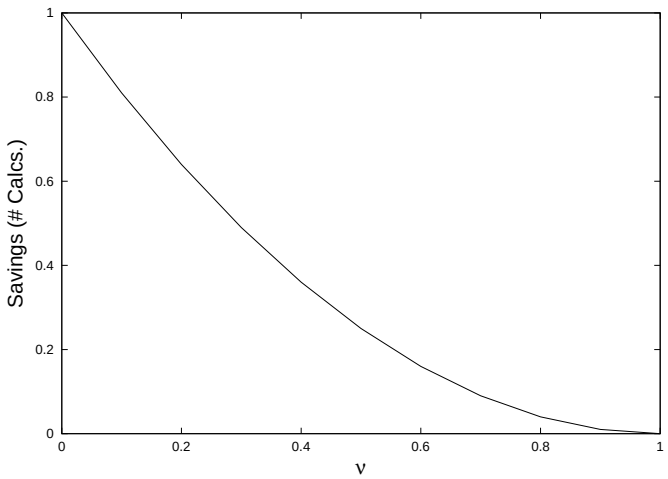

Fig. 2. Plot of reduction in # calculations versus proportion of documents used as basis vectors

1) **TF** - **T**erm **F**requency. This is given by:

$$TF(i) = \frac{n_i}{\sum_{j=1}^{k} n_j}, \tag{6}$$

where n_i is the number of occurrences of term i in the document and k is the length of the document.

2) **TFIDF** - **T**erm **F**requency **I**nverse **D**ocument **F**requency. For term i, this is defined as:

$$TFIDF(i) = TF(i).\log \frac{n}{|\{d : d \in \mathcal{D}, t_i \in d\}|}. \tag{7}$$

where $\mathcal{D}$) is the set of documents being studied.

Finally, the cosine distance, given by:

$$CosDist(d_1, d_2) = 1 - \frac{\vec{d_1}.\vec{d_2}}{|\vec{d_1}| \times |\vec{d_2}|}, \tag{8}$$

is used to represent distances between pairs of documents d_1 and d_2, where $\vec{d}$ is the vector representation of document d.

C. Shortest path method for inferring version history

Once the inter-document distances for all pairs of documents in the collection have been calculated, the version history of the collection can be inferred by finding the minimum *Hamiltonian Path*. In graph theory, a Hamiltonian path is a path in an undirected graph which visits each vertex of the graph exactly once (figure 3 is an illustration of this). The document versioning problem maps easily into this context, where each document is regarded as a vertex, with the weight of the edges connecting the documents given by the edit distances between them. Based on the principle of parsimony, we assume that the path with the least number of edits, i.e. the path with minimum total weight, is the one most likely to represent the actual version history.

The problem of finding this minimal path is of exponential complexity and is not solvable via a full enumeration of all possible paths; however, many techniques exist for finding both approximate and exact solutions. For our experiments we used the following greedy search algorithm:

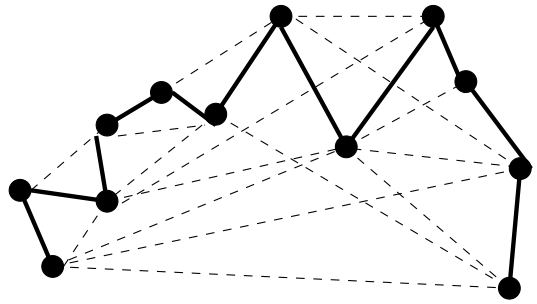

Fig. 3. Densely connected graph, with the heavy line denoting a likely candidate for the minimum Hamiltonian Path

Algorithm *GreedySearch*($\{\mathcal{V}, \mathcal{E}\}$)
Input: A set $\{\mathcal{V}, \mathcal{E}\}$ of vertices and weighted edges (i.e.: distance)
Output: Shortest Hamiltonian path found $[v_1, v_2, \ldots, v_n]$

```
v←[]
Select random v ∈ V and insert into list v
repeat
    V ← V − {v}
    Select v ∈ V with minimum distance to the vertices
    on either end of v
    Append v to this end of v
until |V| == 0
```

To test the reliability of the algorithm, we also used a branch-and-bound solver which was found to produce identical results for subsets of the data at a time. We concede that a more optimised implementation is certainly possible, as is a solution using other heuristic methods (for example genetic algorithms), which might have produced slightly better results. However, it was felt that for the present preliminary investigation the method chosen was adequate.

D. Scoring of results

To evaluate the accuracy of the methods tested, we use the following procedure (Woon and Wong, 2008):

Algorithm *Score*(w, $d_1, d_2, \ldots, d_n$)

```
Accuracy ←0
for i ←1 to n
        TmpAccuracy ←0
        for j ←max{1, i − w} to (i − 1)
                if Earlier(d_j, d_i) then
                TmpAccuracy ←TmpAccuracy+1
        for j ←(i + 1) to min{n, i + w}
                if Earlier(d_i, d_j) then
                TmpAccuracy ←TmpAccuracy+1
        Accuracy ←Accuracy+TmpAccuracy / (min{n, i+w} − max{1, i−w} − 1)
return 100×Accuracy/n%
```

where the function $Earlier(d_i, d_j)$ returns TRUE if d_1 occurs before d_j in the actual version history of the document. As can be seen, the procedure works by sliding a window of length w over the inferred document sequence. The score for the document in the center of this window is given by the proportion of the other documents in the window which are correctly sequenced with respect to the the document in

question. The total score for the entire version history is the average over the scores of all the individual documents.

E. Data

To test the capabilities of the procedure, labelled sets of document versions were required. As there appear to be no standard test collections for such experiments, we had previously described simple methods by which appropriate data sets may be constructed (Woon and Wong, 2008). Two sets of data are used in our current experiments, as described briefly below:

1) Linux Kernel source code: In the first test, the proposed method is used to version C source files taken from the linux kernel. This provides a suitable test set as it has a large number of versions (at the time of writing 656 versions in total) - making it suitably challenging - and it is a well known and publicly available resource[1].

As the source distribution consists of many files, presenting results for all of them would be redundant and excessive. Instead, we focus on the following three randomly selected files:

- /linux/drivers/block/loop.c
- /linux/kernel/fork.c
- /linux/net/ethernet/eth.c

However, in almost all the files surveyed, there were also duplicated files as not all files are modified with each version of the kernel source code. After removing duplicate copies, the number of versions for each of the files above is 127, 216 and 58 for *loop.c*,*fork.c* and *eth.c* respectively.

2) "Wiki" pages: Besides source code files, we also test the performance of the method when applied to other types of files. One convenient option was to work with webpages. To obtain a versioned set of webpages, the "Wayback Machine"[2] was used to retrieve previous versions of webpages.

Ideally we would like to use webpages which are regularly updated and which have a large proportion of text content (as opposed to javascript, "active" components, etc). It was decided that "Wikipedia"[3] entries were a good match to this criteria. For our tests, we chose Wikipedia entries for the *Fourier Transform* and *Perl*, as these are popular topics with regularly updated and comparatively lengthy entries.

For our tests, we downloaded all available versions of these pages from the Internet Archive. For the Fourier Transform page, there were 24 versions available while for the Perl page, we were able to obtain 82 versions.

III. Results

A. Notation

The experiments discussed in the previous section were conducted as described and the results are tabulated in tables I to V. The following notation is used:

1) **Feature extraction schemes**: The method of feature extraction used is indicated in the leftmost column of each table and is written as <*feature*>, where *feature* is one of {DA (document alignment), FS (feature space), TFIDF, TF}, as discussed in section II-B. FS features are additionally written as FS-$< x\% >$, where $x\%$ refers to the percentage of documents in the collections which are used as "basis vectors" in the feature space. Further, for results corresponding to the Wikipedia pages, feature names will be written as <*feature*>-<*format*>, where *format* is one of {html,text}, and indicates if the procedure was applied to pure HTML, or if the HTML tags had been stripped in advance.
2) **Scoring schemes**: The scoring scheme used is shown in the lower-half of the header row of each table. As explained in section II-D, the method used is tunable to different window sizes, allowing different resolutions to be taken into account. Rather than focus on any one window size, a small selection of sizes were evaluated, and are denoted in the tables as:
   - **Global** - Window size was set to the number of documents in the collection
   - **Local-5** - Window size of five
   - **Local-1** - Window size of one
3) Finally, there are two sets of FS scores, one in which random sampling of basis documents is used (for this set, experiments were repeated 10 times and the results averaged), and one in which documents were sampled uniformly along the version history. The two sets have been clearly marked in the relevant tables and figures.

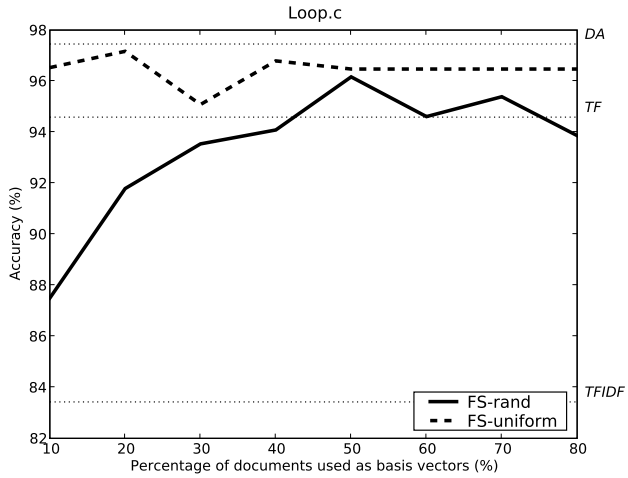

Fig. 4. Accuracy of versioning vs. number of basis vectors used: *loop.c*

B. Source code versioning task

First, we study the performance of the algorithm with the Kernel source code. For the files *loop.c*, *fork.c* and *eth.c*, the results are presented graphically in Figures 4, 5 and 6 respectively, which allow the general trends to be discerned. Detailed results are provided in Tables I, to III.

The following observations could be made:

1) The FS approach is able to produce very accurate results which approach those obtained using DA. Broadly

[1]Linux Kernel Archive - http://www.kernel.org
[2]From the "Internet Archive" - http://www.archive.org
[3]Wikipedia - http://www.wikipedia.org

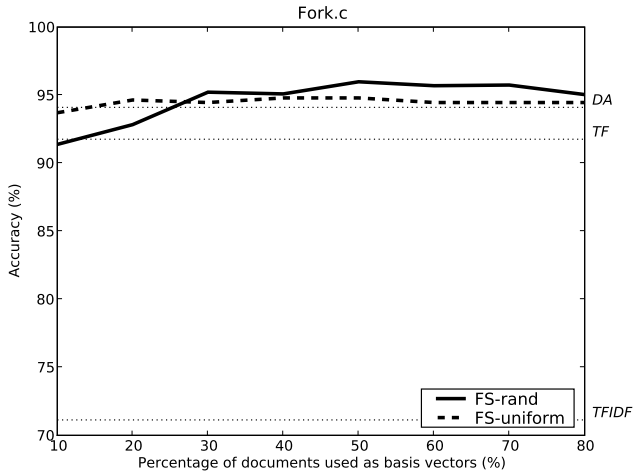

Fig. 5. Accuracy of versioning vs. number of basis vectors used: *fork.c*

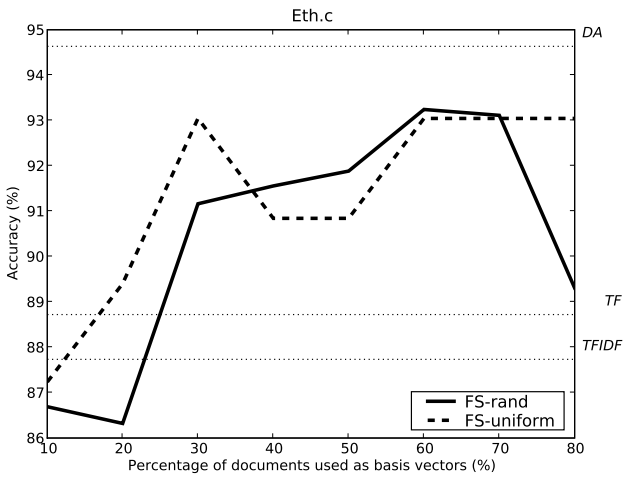

Fig. 6. Accuracy of versioning vs. number of basis vectors used: *eth.c*

speaking, accuracy was observed to improved as the number of basis vectors used was increased, as is expected, though the incremental benefit of increasing the number of basis vectors diminshed after a certain point (generally around $20 - 30\%$ of $|\mathcal{D}|$). This means that the proposed method allows comparable performance with less than a third of the computational effort of calculating the full distance matrix.

2) In common with the results obtained using the full distance matrices, local accuracy values are consistently less accurate compared to the global accuracy values. However, this drop in performance appears to be more pronounced when using FS. This suggests that the feature space projection was able to capture the overall structure of the document space, but that finer details are still quite difficult to capture. This is one aspect in which the uniform sampling seemed to help - when using random sampling, global scoring was often quite good but dropped off quite rapidly when narrowed scoring windows were used.

	Method/ Class	Scores(%)			
		Global	Local-5	Local-1	Average
Original	DA	99.0	**97.3**	**96.1**	**97.5**
	TFIDF	97.1	82.7	70.5	83.4
	TF	98.7	93.7	91.3	94.6
Randomized	FS-10%	94.4	86.0	82.1	87.5
	FS-20%	98.3	91.3	85.7	91.8
	FS-30%	98.7	93.2	88.7	93.5
	FS-40%	98.7	93.7	89.8	94.1
	FS-50%	99.0	96.0	93.5	96.2
	FS-75%	98.8	95.0	92.2	95.3
Uniform	FS-10%	99.0	96.9	93.7	96.5
	FS-20%	**99.1**	97.2	95.3	97.2
	FS-30%	99.0	95.7	90.6	95.1
	FS-40%	99.0	96.9	94.5	96.8
	FS-50%	99.0	96.7	93.7	96.5
	FS-75%	99.0	96.7	93.7	96.5

TABLE I
VERSIONING SCORES (**LOOP.C**). SCORES IN BOLD DENOTE HIGHEST SCORING ITEMS IN THE RESPECTIVE COLUMNS.

3) The accuracy of the FS approach was significantly improved when the basis vectors were uniformly spaced along the version history. This makes sense as in this way there is a higher chance of obtaining a better coverage of the distribution of $\mathcal{D}$ in the feature space.
4) Interestingly, for the file *fork.c*, the accuracy obtained using FS *exceeded* that obtained with DA. This was quite surprising since the feature space projections use information that is already contained in the full distance matrix. One possible explanation is that using the projections in this way helped to eliminate noise in the data by restricting the analysis to only a subspace of the full feature space, while excluding spurious components which fall outside this subspace.
5) Another observation was that the larger the set of documents being analyzed, the better the accuracy of the method.

C. Wikipedia entries

As with the kernel source code, results are presented both graphically (Figures 7 and 8 for the *Fourier Transform* and *Perl* Wikipedia pages respectively), and in tabular form (Tables IV and V).

1) The results exhibit the same broad trends as with the Kernel source code, though the performance of the FS-based methods showed some overall decline. As before, accuracy improved when the number of basis vectors was increased, but again this was only up to a point before saturation was observed. Also, this improvement was by no means smooth, as there was a great degree

	Method/ Class	Scores(%)			
		Global	Local-5	Local-1	Average
Original	DA	99.5	92.1	90.7	94.1
	TFIDF	71.7	70.8	70.8	71.1
	TF	99.1	89.4	86.8	91.8
Randomized	FS-10%	98.8	88.6	86.8	91.4
	FS-20%	98.8	90.6	89.1	92.8
	FS-30%	**99.6**	93.7	92.4	95.2
	FS-40%	**99.6**	93.6	92.1	95.1
	FS-50%	**99.6**	**94.6**	**93.7**	**96.0**
	FS-75%	**99.6**	93.9	92.8	95.4
Uniform	FS-10%	99.4	92.4	89.4	93.7
	FS-20%	**99.6**	93.6	90.7	94.6
	FS-30%	**99.6**	93.5	90.3	94.5
	FS-40%	**99.6**	93.6	91.2	94.8
	FS-50%	**99.6**	93.6	91.2	94.8
	FS-75%	**99.6**	93.5	90.3	94.5

TABLE II

VERSIONING SCORES (**FORK.C**). SCORES IN BOLD DENOTE HIGHEST SCORING ITEMS IN THE RESPECTIVE COLUMNS.

	Method/ Class	Scores(%)			
		Global	Local-5	Local-1	Average
Original	DA	**98.7**	93.9	**91.4**	**94.6**
	TFIDF	97.5	88.2	77.6	87.7
	TF	96.9	86.5	82.8	88.7
Randomized	FS-10%	96.4	86.5	77.2	86.7
	FS-20%	92.6	87.4	79.0	86.3
	FS-30%	98.1	91.7	83.7	91.2
	FS-40%	98.1	91.6	84.9	91.6
	FS-50%	98.2	92.2	85.3	91.9
	FS-75%	98.3	92.7	85.9	92.3
Uniform	FS-10%	97.2	86.0	78.4	87.2
	FS-20%	97.5	88.0	82.8	89.4
	FS-30%	**98.7**	**94.2**	86.2	93.0
	FS-40%	95.3	91.8	85.3	90.8
	FS-50%	95.3	91.8	85.3	90.8
	FS-75%	**98.7**	**94.2**	86.2	93.0

TABLE III

VERSIONING SCORES (**ETH.C**). SCORES IN BOLD DENOTE HIGHEST SCORING ITEMS IN THE RESPECTIVE COLUMNS.

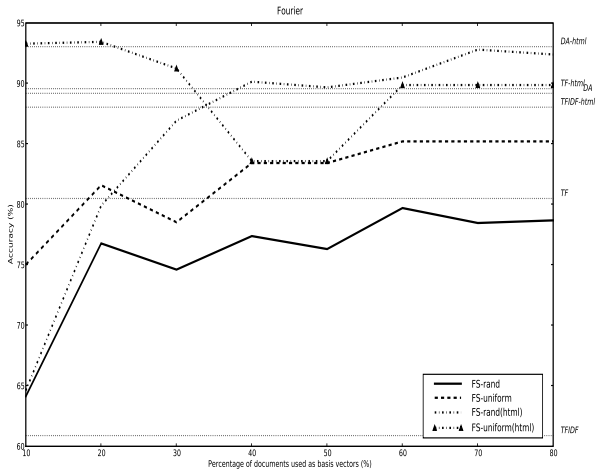

Fig. 7. Accuracy of versioning vs. number of basis vectors used: *Wikipedia-Fourier*

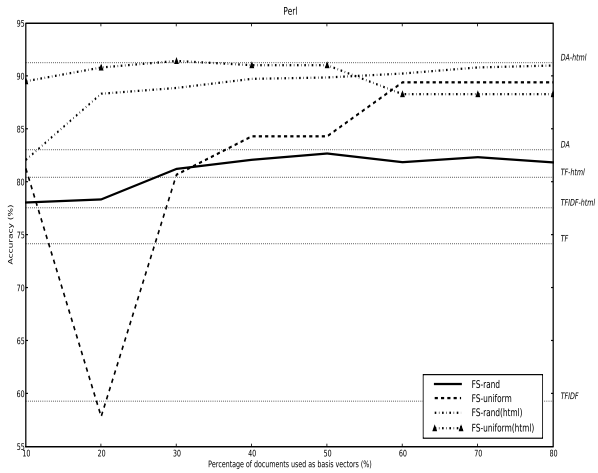

Fig. 8. Accuracy of versioning vs. number of basis vectors used: *Wikipedia-Perl*

of fluctuation. Also, the accuracy of FS methods when using uniformly spaced basis vectors was superior to the randomly chosen vectors.

2) Of the two Wikipedia files studied, better results were obtained when versioning the *Perl* wikipedia page. This is consistent with what was observed when versioning the Kernel source files where the scores obtained with largest collection of file - in that case *fork.c* - were the highest.
3) However, the size of the document collection was not the only factor - for example, the Wikipedia *Perl* dataset contained 82 versions compared to 58 for the *eth.c* collection, yet the versioning accuracy of the former was still markedly lower.
4) For both the *Fourier Transform* and *Perl* pages, the overall highest scores were obtained using FS.

	Method/ Class	Scores(%)			
		Global	Local-5	Local-1	Average
Original	DA-html	97.5	94.2	87.5	93.0
	DA-text	90.9	91.2	85.4	89.2
	TFIDF-html	97.1	92.0	75.0	88.0
	TFIDF-text	63.4	56.8	62.5	60.9
	TF-html	97.1	92.4	79.2	89.6
	TF-text	87.0	81.6	72.9	80.5
Randomized	FS-10%	70.6	62.8	59.0	64.1
	FS-30%	86.1	78.0	59.8	74.6
	FS-50%	87.2	81.1	60.6	76.3
	FS-75%	87.5	82.4	61.7	77.2
Rand.(html)	FS-10%	73.3	63.6	62.3	66.4
	FS-30%	96.1	89.9	77.5	87.8
	FS-50%	96.2	90.2	79.2	88.5
	FS-75%	97.6	94.2	86.7	92.8
Uniform	FS-10%	80.4	78.0	66.7	75.0
	FS-30%	87.7	83.3	64.6	78.5
	FS-50%	89.5	87.9	72.9	83.4
	FS-75%	89.9	88.7	77.1	85.2
Unif.(html)	FS-10%	**97.8**	**94.5**	87.5	**93.3**
	FS-30%	97.1	93.3	83.3	91.2
	FS-50%	89.1	86.6	75.0	83.6
	FS-75%	90.2	89.8	**89.6**	89.9

TABLE IV

DOCUMENT VERSIONING SCORES: WIKIPEDIA FOURIER TRANSFORM ENTRIES. SCORES IN BOLD DENOTE HIGHEST SCORING ITEMS IN THE RESPECTIVE COLUMNS.

	Method/ Class	Scores(%)			
		Global	Local-5	Local-1	Average
Original	DA-html	95.5	91.1	87.2	91.3
	DA-text	77.5	85.7	86.0	83.1
	TFIDF-html	95.6	74.3	62.8	77.6
	TFIDF-text	60.6	60.0	57.3	59.3
	TF-html	94.1	78.9	68.3	80.4
	TF-text	77.6	76.7	68.3	74.2
Randomized	FS-10%	77.0	82.6	74.6	78.1
	FS-30%	77.6	86.3	79.8	81.2
	FS-50%	77.8	87.5	82.8	82.7
	FS-75%	77.8	88.0	84.1	83.3
Rand.(html)	FS-10%	93.1	82.3	75.6	83.6
	FS-30%	96.1	89.2	82.1	89.1
	FS-50%	96.8	90.9	81.7	89.8
	FS-75%	**97.0**	92.2	82.7	90.6
Uniform	FS-10%	77.6	86.3	79.9	81.3
	FS-30%	77.6	85.8	78.7	80.7
	FS-50%	78.0	89.0	86.0	84.3
	FS-75%	78.5	**94.6**	**95.1**	89.4
Unif.(html)	FS-10%	96.6	90.9	81.1	89.5
	FS-30%	96.9	91.5	86.0	**91.4**
	FS-50%	96.9	91.5	84.8	91.0
	FS-75%	96.5	89.8	78.7	88.3

TABLE V

DOCUMENT VERSIONING SCORES: WIKIPEDIA PERL ENTRIES. SCORES IN BOLD DENOTE HIGHEST SCORING ITEMS IN THE RESPECTIVE COLUMNS.

IV. CONCLUSIONS AND FUTURE PLANS

The results presented here demonstrate that the feature space approximation is a useful and efficient alternative to aligning all pairs of documents in the study set. Potential avenues for further development include:

1) A more systematic method could be found for the selection of the basis nodes. In the present study, two methods were demonstrated for selecting the nodes: random and selection of evenly spaced versions of the file, where the results indicated that the method performed better when evenly spaced versions were used. However this has two main problems: firstly, in a real versioning scenario, we would not have access to the underlying sequence of documents to start with. Secondly, even this method could be suboptimal as the versions may not be uniformly distributed in the feature space. A method based on clustering of the nodes, then selecting cluster centers might be more appropriate.
2) While this method reduces the number of alignments required to analyze a set of documents, there is still the issue of the heavy computational requirements of aligning a pair of documents. For a full alignment of a pair of documents, the complexity is $\mathcal{O}(mn)$, which will become very time consuming when a large collection is to be analysed. However, many options exist for optimising the basic dynamic programming algorithm; indeed bioinformatics tools such as BLAST and FASTA have been devised which can search massive biological databases very efficiently using a variety of simplifications (such as only focussing on promising regions of the alignment matrix). There is no reason why similar simplifications could not be applied to the analysis of text databases. This is certainly an avenue which should be explored further if the general approach described here is to be scaled to very large collections of documents.

Experiments to test the proposed method were conducted with two different classes of documents, and with at least two sets of versions each. In general, the results obtained were very favourable and pointed to the usefulness of the proposed approach.

ACKNOWLEDGEMENT

The work described in the paper was carried out with financial support from the Ministry of Science, Technology and Innovation, Malaysia, via the ScienceFund grant 01-02-

05-SF0003. The authors would like to thank the Ministry for its support for this research.

References

Aradhye, H. and Dorai, C. (2002). New kernels for analyzing multimodal data in multimedia using kernel machines. In *Multimedia and Expo, 2002. ICME '02. Proceedings. 2002 IEEE International Conference on*, volume 2, pages 37–40 vol.2.

Cancedda, N., Gaussier, E., Goutte, C., and Renders, J. M. (2003). Word sequence kernels. *J Mach Learn Res*, 3:1059–1082.

Cristianini, N. and Taylor, S. (2000). *An introduction to support vector machines*. Cambridge University Press, Cambridge, UK.

Honkela, T. and Hyvarinen, A. (2004). Linguistic feature extraction using independent component analysis. volume 1.

Hung, C. and Wermter, S. (2004). Neural network based document clustering using wordnet ontologies. *Int J Hybrid Intel Syst*, 1(3,4):127–142.

Lagus, K., Kaski, S., and Kohonen, T. (2004). Mining massive document collections by the websom method. *Inf Sci*, 163(1-3):135–156.

Lan, M., Tan, C.-L., Low, H.-B., and Sung, S.-Y. (2005). A comprehensive comparative study on term weighting schemes for text categorization with support vector machines. In *WWW '05: Special interest tracks and posters of the 14th international conference on World Wide Web*, pages 1032–1033, New York, NY, USA. ACM Press.

Larsen, B. and Aone, C. (1999). Fast and effective text mining using linear-time document clustering. In *SIGKD99D*, pages 16–22, San Diego, USA.

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710.

Liu, R. L. and Lu, Y. L. (2002). Incremental context mining for adaptive document classification. In *KDD*, pages 599–604. ACM.

Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., and Watkins, C. (2002). Text classification using string kernels. *J Mach Learn Res*, 2:419–444.

Lodhi, H., Taylor, J. S., Cristianini, N., and Watkins, C. J. C. H. (2000). Text classification using string kernels. In *Advances in Neural Information Processing Systems (NIPS)*, pages 563–569.

Mercer, J. (1909). Functions of positive and negative type, and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 209:415–446.

Merkl, D. (1997). Exploration of text collections with hierarchical feature maps. In *SIGIR '97: Proceedings of the 20th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 186–195, New York, NY, USA. ACM Press.

Mount, D. W. (2001). *Bioinformatics: Sequence and Genome Analysis*. Cold Spring Harbor Laboratory Press, Cold Spring Harbor, New York.

Muthukrishnan, S. (2002). Efficient algorithms for document retrieval problems. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 657–666, San Francisco, USA.

Needleman, S. B. and Wunsch, C. D. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J Mol Biol*, 48(3):443–453.

Park, L. A. F., Ramamohanarao, K., and Palaniswami, M. (2005). A novel document retrieval method using the discrete wavelet transform. *ACM T Inform Syst*, 23(3):267–298.

Pullwitt, D. (2002). Integrating contextual information to enhance som-based text document clustering. *Neural Netw*, 15(8-9):1099–1106.

Soukoreff, W. R. and Mackenzie, S. I. (2001). Measuring errors in text entry tasks: an application of the levenshtein string distance statistic. In *CHI '01: CHI '01 extended abstracts on Human factors in computing systems*, pages 319–320, New York, NY, USA. ACM Press.

Tan, S. (2006). An effective refinement strategy for knn text classifier. *Expert Syst Appl*, 30(2):290–298.

Woon, W. L. and Wong, K. D. (2006). A self organising approach to document space visualisation. *Multimedia Cyberscape Journal (Special Issue on Multimedia Data Mining*, 4(1).

Woon, W. L. and Wong, K.-S. (2008). String alignment for automated document versioning. *Knowledge and Information Systems*.

Yang, H.-C. and Lee, C.-H. (2004). A text mining approach on automatic generation of web directories and hierarchies. *Expert Syst Appl*, 27(4):645–663.

Zaiane, O. and Antonie, M.-L. (2002). Classifying text documents by associating terms with text categories. volume 5 of *Conferences in Research and Practice in Information Technology*, pages 215–222, Melbourne, Australia. ACS. Thirteenth Australasian Database Conference (ADC2002).