Document Versioning Using Feature Space Distances

Wei Lee Woon[1], K. Daniel Wong[2], Zeyar Aung[1], Davor Svetinovic[1]

[1]EECS, Masdar Institute of Science and Technology,
P.O. Box 54224 Abu Dhabi, UAE.
[2]Daniel Wireless Software Pte Ltd, #03-04, 3 Science Park Drive,
Singapore Science Park, Singapore 118223
wwoon@masdar.ac.ae,ksdwong@gmail.com,{zaung,dsvetinovic}@masdar.ac.ae

Abstract. The automated analysis of documents is an important task given the rapid increase in availability of digital texts. 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.

Keywords: String Matching, Text Processing, Data Mining, Versioning, Information Retrieval

1 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. In an earlier work [1], we demonstrated a procedure for inferring the version histories of unlabelled document collections using the edit distances between these documents. However, 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.

1.1 Relevant background

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 [2]; 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 [3]. 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 [4, 5]. The original study examined the use of character level kernels but similar kernels calculated at the level of words have also been proposed [6]. 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 [7]. 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.

2 Experimental Methodology

2.1 Feature extraction

Edit distances between pairs of documents and strings are found using dynamic programming. However, this is computationally expensive, and the addition of new documents requires comparisons with all other documents in the set. To address this, we propose a method by which the number of document comparison required may be greatly reduced. The proposed method exploits the kernel trick [8]; 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:

$$K(x_1, x_2) = \langle \Phi_K(x_1), \Phi_K(x_2) \rangle \,, \tag{1}$$

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.

In the present context, we note that the edit distance measure possesses the characteristics of a Mercer kernel [9]. 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 can be empirically tested by calculating the eigenvalues of the closely related similarity matrix $\mathbf{\Sigma}$:

$$\mathbf{\Sigma} = \begin{bmatrix} d_{max} - d_{1,1} & \dots & d_{max} - d_{1,n} \\ \vdots & \ddots & \vdots \\ d_{max} - d_{n,1} & \dots & d_{max} - d_{n,n} \end{bmatrix}; \tag{2}$$

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. Also, since the collection of documents being studied would occupy only a subspace $\mathcal{S}$ of this space, we 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. Finally, we obtain the coordinates of a given document in $\mathcal{S}$ by projecting it onto $\mathcal{B}$ via the edit distances:

$$\boldsymbol{\mu}_i = \{d_{max} - d_{i,j} : j \in \mathcal{B}\}\,. \tag{3}$$

Distances between documents i and j can now be found quickly by calculating the Euclidean distances in this transformed space.

For the sake of comparison, two conventional vector-space representations, Term Frequency (*TF*) and Term Frequency Inverse Document Frequency (*TF-IDF*) [10] have been included in our experiments. In these cases, the cosine distance is used to compare different documents.

2.2 Scoring of results

The version history of the documents can now be inferred by finding the *Hamiltonian Path* through all the documents then comparing them to the true version histories as follows [1]:

Algorithm *Score*$(\mathrm{w}, d_1, d_2, \ldots, d_n)$
1. Accuracy $\leftarrow$0
2. **for** $i \leftarrow$1 **to** n
3. TmpAccuracy $\leftarrow$0
4. **for** $j \leftarrow \max\{1, i - \mathrm{w}\}$ **to** $(i-1)$
5. **if** *Earlier*(d_j, d_i) **then**
6. TmpAccuracy $\leftarrow$TmpAccuracy+1
7. **for** $j \leftarrow (i+1)$ **to** $\min\{n, i + \mathrm{w}\}$
8. **if** *Earlier*(d_i, d_j) **then**
9. TmpAccuracy $\leftarrow$TmpAccuracy+1
10. Accuracy $\leftarrow$Accuracy+$\frac{\mathrm{TmpAccuracy}}{\min\{n, i+\mathrm{w}\} - \max\{1, i-\mathrm{w}\} - 1}$
11. **return** $100 \times$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.

2.3 Data

Two sets of data are used in our current experiments [1]:

1. **Linux Kernel source code** - In the first test, the proposed method is used to version C source files taken from the linux kernel. The source distribution consists of many files so testing was done with the following three randomly selected files: *loop.c*,*fork.c* and *eth.c*, for which 127, 216 and 58 unique versions were extracted respectively.
2. **"Wiki" pages** - versioned Wikipedia pages were collected using the "Wayback Machine"[1]. For our tests, we chose Wikipedia entries for *Fourier Transform* and *Perl*, as these are popular topics with regularly updated and comparatively lengthy entries.
   For our tests, we downloaded all available (at the time of writing) 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.

3 Results

3.1 Notation

For the results presented here, the following notation is used:

1. **Feature extraction schemes**: The method of feature extraction is written as <*feature*>, where *feature* is one of {DA (document alignment), FS (feature space), TFIDF, TF}. 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**: As mentioned, the scoring scheme is tunable to different window sizes. These are denoted as **Global**, **Local-5**,**Local-1** for the case with all documents, window size of 5, and of 1.

3.2 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 in Tables 1, to 3.

The following observations could be made:

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

[1] From the "Internet Archive" - http://www.archive.org

means that the proposed method allows comparable performance with less than a third of the computational effort of calculating the full distance matrix. However, as with the full distance matrices, local accuracy values are consistently less accurate compared to the global accuracy values.
2. 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.
3. Interestingly, for the file *fork.c*, the accuracy obtained using FS *exceeded* that obtained with DA, even though the feature space projections uses 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 the relevant subspace of the full feature space.
4. The larger the set of documents being analyzed, the better the accuracy of the method.

	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-30%	98.7	93.2	88.7	93.5
	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-30%	**99.0**	95.7	90.6	95.1
	FS-50%	**99.0**	96.7	93.7	96.5
	FS-75%	**99.0**	96.7	93.7	96.5

Table 1. Versioning scores (**loop.c**). Scores in bold denote highest scoring items in the respective columns.

3.3 Wikipedia entries

The same experiments were then repeated using the Wikipedia versions, and results have been tabulated in Table 4.

	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-30%	**99.6**	93.7	92.4	95.2
	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-30%	**99.6**	93.5	90.3	94.5
	FS-50%	**99.6**	93.6	91.2	94.8
	FS-75%	**99.6**	93.5	90.3	94.5

Table 2. 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-30%	98.1	91.7	83.7	91.2
	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-30%	**98.7**	**94.2**	86.2	93.0
	FS-50%	95.3	91.8	85.3	90.8
	FS-75%	**98.7**	**94.2**	86.2	93.0

Table 3. Versioning scores (**eth.c**). Scores in bold denote highest scoring items in the respective columns.

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. 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.
2. 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.

	Method/ Class	Fourier; Scores (%)				Perl; Scores (%)			
		Global	Local-5	Local-1	Average	Global	Local-5	Local-1	Average
Original	DA	97.5	94.2	87.5	93.0	95.5	91.1	**87.2**	91.3
	TFIDF	97.1	92.0	75.0	88.0	95.6	74.3	62.8	77.6
	TF	97.1	92.4	79.2	89.6	94.1	78.9	68.3	80.4
Randomized	FS-10%	73.3	63.6	62.3	66.4	93.1	82.3	75.6	83.6
	FS-30%	96.1	89.9	77.5	87.8	96.1	89.2	82.1	89.1
	FS-50%	96.2	90.2	79.2	88.5	96.8	90.9	81.7	89.8
	FS-75%	97.6	94.2	86.7	92.8	**97.0**	**92.2**	82.7	90.6
Uniform	FS-10%	**97.8**	**94.5**	87.5	**93.3**	96.6	90.9	81.1	89.5
	FS-30%	97.1	93.3	83.3	91.2	96.9	91.5	86.0	**91.4**
	FS-50%	89.1	86.6	75.0	83.6	96.9	91.5	84.8	91.0
	FS-75%	90.2	89.8	**89.6**	89.9	96.5	89.8	78.7	88.3

Table 4. Document versioning scores: Wikipedia entries. Scores in bold denote highest scoring items in the respective columns.

4 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. For example, a method based on clustering of the nodes, then selecting the cluster centers as the basis vectors might be more appropriate

2. Optimisation of the basic dynamic programming algorithm. For e.g. 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).

References

1. Wei Lee Woon and Kuok-Shoong Wong. String alignment for automated document versioning. *Knowledge and Information Systems*, 2008.
2. Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966.
3. William R. Soukoreff and Scott I. Mackenzie. 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, 2001. ACM Press.
4. Huma Lodhi, John S. Taylor, Nello Cristianini, and Christopher J. C. H. Watkins. Text classification using string kernels. In *Advances in Neural Information Processing Systems (NIPS)*, pages 563–569, 2000.
5. Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. Text classification using string kernels. *J Mach Learn Res*, 2:419–444, 2002.
6. Nicola Cancedda, Eric Gaussier, Cyril Goutte, and Jean M. Renders. Word sequence kernels. *J Mach Learn Res*, 3:1059–1082, 2003.
7. N. Cristianini and S. Taylor. *An introduction to support vector machines.* Cambridge University Press, Cambridge, UK, 2000.
8. John Mercer. 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, 1909.
9. H. Aradhye and C. Dorai. 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, 2002.
10. Man Lan, Chew-Lim Tan, Hwee-Boon Low, and Sam-Yuan Sung. 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, 2005. ACM Press.