International Journal on Artificial Intelligence Tools

XML Clustering and Retrieval through Principal Component Analysis

Jason T. L. Wang*, Jianghui Liu and Junhan Wang
Department of Computer Science, New Jersey Institute of Technology
University Heights, Newark, NJ 07102, USA

Received (12 September 2004)
Revised (23 March 2005)
Accepted (20 April 2005)

XML is increasingly important in data exchange and information management. A great deal of efforts have been spent in developing efficient techniques for storing, querying, indexing and accessing XML documents. In this paper we propose a new approach to clustering XML data. In contrast to previous work, which focused on documents defined by different DTDs, the proposed method works for documents with the same DTD. Our approach is to extract features from documents, modeled by ordered labeled trees, and transform the documents to vectors in a high-dimensional Euclidean space based on the occurrences of the features in the documents. We then reduce the dimensionality of the vectors by principal component analysis (PCA) and cluster the vectors in the reduced dimensional space. The PCA enables one to identify vectors with co-occurrent features, thereby enhancing the accuracy of the clustering. We also discuss an extension of our techniques to XML retrieval. Experimental results based on documents obtained from Wisconsin's XML data bank show the effectiveness and good performance of the proposed techniques.

Keywords: Document clustering; information retrieval; data mining.

1. Introduction

XML processing is a problem of continuing interest.[1,2,3,4,5,6] For example, Jagadish *et al.*[7] presented an algebraic language for XML querying. Chen *et al.*[8] studied techniques for selectivity estimation in XML retrieval. Garofalakis and Kumar[9] developed algorithms for comparing and correlating XML data streams. Much of the work models an XML document using an ordered labeled tree, in which each node has a label and the left to right order among siblings is important.[10,11] For example, Figure 1 shows an XML document and its tree representation. The simplicity and scalability brought out by the mechanism of the tree-like information organization contribute significantly to the success of XML technology.

Since an XML document is represented as a tree structure, one can explore the relationship among XMLs using various tree matching algorithms.[12,13,14,15,16] A

*Contact author: wangj@njit.edu

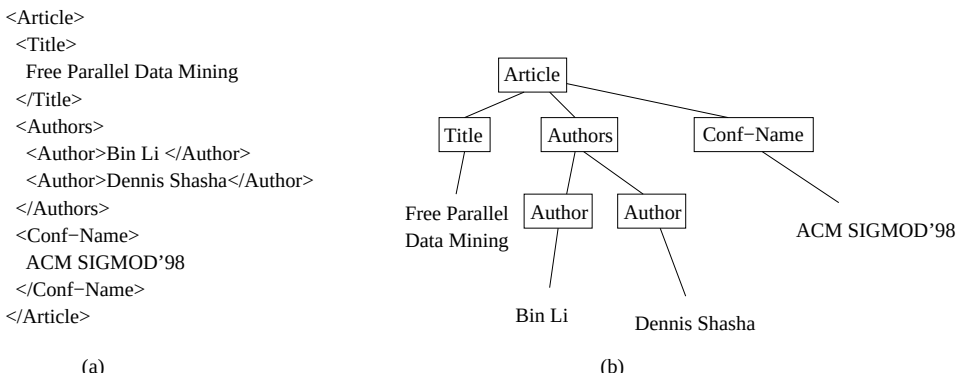

Fig. 1. Illustration of an XML document and its tree representation; (a) a sample XML document and (b) the tree representation of the document in (a).

closely related problem is to find trees in a database that "match" a given pattern or query tree.[17] This type of retrieval often exploits various filters that eliminate unqualified data trees from consideration at an early stage of retrieval. The filters accelerate the retrieval process.[10] Another approach to facilitating a search is to cluster XMLs into appropriate categories, either before or after the search is conducted. Clustering is also helpful in information organization, data mining and integration, among others.[18]

One natural way to cluster XMLs is based on the DTD (document type definition) used for governing the construction of a specific type of XML documents. Recently, Lee *et al.*[19] represented DTDs as trees and proposed a technique, called XClust, to cluster the DTDs by calculating the similarities among the DTD trees. In XClust, the similarity between two DTDs is computed by measuring pairwise element similarities, which are obtained from summing up three weighted component similarities associated with the element pairs in the two DTDs.

While Lee *et al.*'s work is useful in categorizing XML documents with different DTDs, they do not address how to cluster XML documents sharing the same DTD. Here we propose a new approach to clustering these documents. The XMLs we are interested in have the same type (e.g. they all are movie documents), and share the same DTD. We propose different algorithms to extract "features" from these XMLs, where a feature may refer to a path or a node pair in an XML tree. Then we transform an XML tree into a vector of features and build a high-dimensional matrix to represent the XML trees at hand. Next, we apply principal component analysis (PCA) to the matrix to reduce its dimensionality. Finally we use a K-means algorithm[20] to cluster the vectors residing in the reduced dimensional space and place them in appropriate categories. We also discuss an extension of the PCA method to support XML retrieval by accessing the vectors in the reduced dimensional space.

The rest of the paper is organized as follows. Section 2 presents several algo-

rithms for extracting features from XML. Section 3 describes the principal component analysis method used for XML clustering. Section 4 reports experimental results concerning the proposed clustering techniques. Section 5 discusses the extension of the PCA method for XML retrieval. Section 6 concludes the paper and points out some future work.

2. Feature Extraction in XML

A feature extraction algorithm can be considered as a function, g, that maps an XML tree T into a set of features $\mathcal{F} = \{a_1, a_2, \cdots, a_n\}$, viz $g : T \rightarrow \{a_1, a_2, \cdots, a_n\}$. In general, g is not a bijective mapping since two different trees can be mapped into the same set of features. In reality, however, the possibility for this to happen is small. We present here three methods for feature extraction: *path-based*, *nodepair-based* and *hybrid*.

The path-based method works by extracting all paths of lengths less than or equal to L from the tree T, where L is a user-specified parameter value. These paths will be used as features for T. We refer to a path that contains the root of T as a *root-beginning* path and refer to a path that contains a leaf of T as a *leaf-ending* path. A path that starts with the root and ends at a leaf of T is called a *complete* path. Figure 2(a) shows a complete path of length 3 extracted from the XML tree in Figure 1(b). Thus, a path extracted by the path-based method may be a portion of a complete path. The time complexity of the path-based method is $\mathcal{O}(n^2 \times h)$ where n is the number of nodes in the tree T and h is the height of T.

The nodepair-based method uses node pairs extracted from the tree T as features. A pair of nodes u, v could have a sibling relationship, a parent-child relationship, a grandparent-grandchild relationship, or a cousin relationship, etc.[a] We model these various kinds of relationship by considering the number of edges separating u from v. Specifically, ignoring the direction of each edge in T, we say a node pair is of type K if the number of edges on the shortest path connecting u and v is K. Thus, for example, a parent-child node pair is of type 1; a sibling pair is of type 2; a cousin pair is of type 4, etc. Figure 2(b) shows a node pair of type 4 extracted from the XML tree in Figure 1(b).

The nodepair-based method extracts all node pairs of types ranging from 1 to K from the tree T, where K is a user-specified value. These node pairs will be used as features for T. The time complexity of the nodepair-based method is $O(K \times n^2)$, where n is the number of nodes in the tree T. Since K is a constant, the time complexity is asymptotically $\mathcal{O}(n^2)$.

Finally, the hybrid method is to combine all the distinct features extracted by the path-based method and the nodepair-based method and use them altogether as the features for the tree T.

[a] u is a cousin of v if u's parent is a sibling of v's parent.

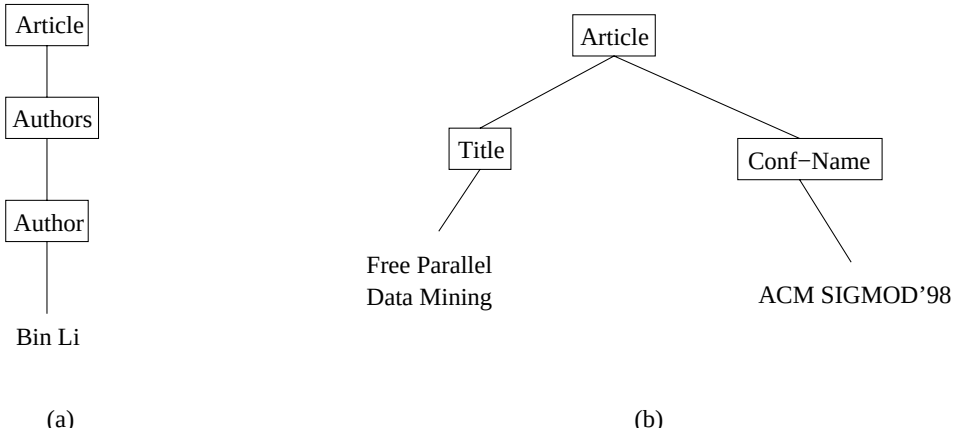

Fig. 2. Two features extracted from the XML tree in Figure 1(b).

3. XML Clustering

3.1. *Vector Representation of an XML Tree*

Let $\mathcal{S}$ be a set of N XML trees that share the same DTD. Each XML tree in $\mathcal{S}$ can be mapped to a set of features $\mathcal{F}$ using the feature extraction algorithms described in Section 2. Without numerical representation, it would be difficult to perform clustering on the feature sets directly. We propose here to represent an XML tree by a vector of integer numbers using the features extracted from the trees in $\mathcal{S}$.

Specifically, let $\mathcal{E}$ contain the distinct features extracted by one of the methods described in Section 2 from all the trees in $\mathcal{S}$ with duplicate features removed. Let C be the size of $\mathcal{E}$. Treat each feature (path or node pair) as a character string and sort the features based on their lexicographical order. Then we represent each tree T as a C-dimensional vector V in Euclidean space based on two models, *boolean* and *weighted*. With the boolean model, the nth component of V is 1 if the nth feature f of $\mathcal{E}$ exists in T, and the nth component of V is 0 otherwise. With the weighted model, the nth component of V is the number of occurrences of the nth feature f in T. If this feature does not appear in T at all, the nth component of V is 0. It is likely that many components (or dimensions) of V have zero values.

Once we are able to represent an XML tree by a vector in a high-dimensional Euclidean space, we can apply vector-based clustering algorithms (e.g. K-means)[20] to cluster these XML trees. Alternatively, one can first apply existing dimensionality reduction algorithms[21,22,23] to these vectors and then cluster the XML trees in a reduced dimensional space. A shortcoming of these dimensionality reduction algorithms is that they may introduce inaccuracy in performing clustering.

We propose here to use principal component analysis (PCA) to reduce the dimensionality of the vectors. When one combines the vectors into a $C \times N$ matrix X where C is the total number of distinct features in $\mathcal{E}$ and N is the total number

of XML trees in the given set $\mathcal{S}$, one obtains a very sparse matrix in which many entries are zeros. The PCA helps to identify significant dimensions and condense the matrix X.

3.2. *Dimensionality Reduction through PCA*

Consider the $C \times N$ matrix X with rank r where the rank represents the maximum number of uncorrelated column vectors in X. Through singular value decomposition (SVD),[24] we can represent X as

$$X = U \cdot S \cdot V^T \tag{1}$$

Both U and V are orthogonal matrices. Each column of U is one of the eigenvectors of the covariance matrix $X \cdot X^T$ where X^T is the transpose of X. Each column of V is one of the eigenvectors of the matrix $X^T \cdot X$. The $r \times r$ matrix S contains eigenvalues of X on S's diagonal line.

In our case, each column vector of the matrix X represents an XML tree, i.e. each XML tree is a vector in the C-dimensional Euclidean space. The dot product between two column vectors reflects the extent to which the two corresponding XML trees share similar feature occurrences. Thus, we can use the dot product to get pairwise tree distances. Let M contain pairwise tree distances, i.e. M_{ij} is the dot product distance between tree T_i and tree T_j. Then M can be derived by:

$$M = X^T \cdot X \tag{2}$$

which can be generalized as:

$$\begin{aligned} M &= (U \cdot S \cdot V^T)^T \cdot (U \cdot S \cdot V^T) && (3)\\ &= (V \cdot S \cdot U^T) \cdot (U \cdot S \cdot V^T) && (4)\\ &= V \cdot S^2 \cdot V^T && (5)\\ &= (V \cdot S) \cdot (V \cdot S)^T && (6) \end{aligned}$$

The derived, new representation of M tells us that the pairwise tree comparison matrix M can be obtained through the dot product between matrices $(V \cdot S)$ and $(V \cdot S)^T$. That is, the ith row of the $N \times r$ matrix $(V \cdot S)$ is a r-dimensional vector, which represents the ith XML tree in the given set $\mathcal{S}$. (This r-dimensional vector is also referred to as a tree-vector.) This result indicates that after performing projection transformation with respect to the matrix V, we can keep pairwise distances between tree-vectors as in the original setting.

Notice that even though the SVD helps to reduce the dimensionality (from C to r where $r < \min(C, N)$) of a tree-vector, there may still exist some redundant dimensions in the reduced r-dimensional tree-vector. In practice, it is possible to further reduce the dimensionality of these tree-vectors without losing the characteristics of the tree-vectors in the original C-dimensional Euclidean space. Specifically,

based on the PCA, the optimal solution conditioned on the squared-error criterion to represent a r-dimensional vector D by a k-dimensional vector $D^{'}$, $k < r$, is to project the r-dimensional vector onto the subspace spanned by the eigenvectors corresponding to the k largest eigenvalues of the covariance matrix $X \cdot X^T$.[25] Consequently, we can obtain an approximation, X_k, of the original matrix X by keeping the k largest eigenvalues of the covariance matrix $X \cdot X^T$ and replacing the remaining eigenvalues with zeros. This is shown pictorially in Figure 3. Only the shaded parts in the figure are used to construct the new matrix X_k.

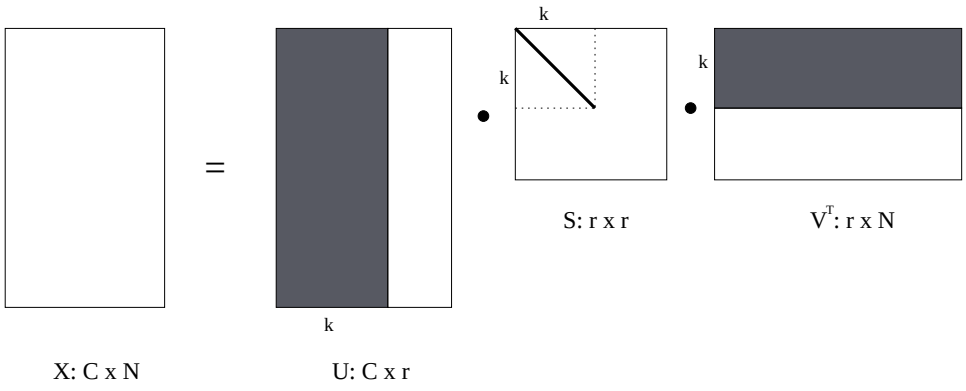

Fig. 3. Dimensionality reduction based on PCA.

By keeping the shaded parts in Figure 3 and the k largest eigenvalues of the covariance matrix $X \cdot X^T$, we rewrite Eq. (1) as:

$$X_k = U_k \cdot S_k \cdot V_k^T \tag{7}$$

As a result, the $N \times k$ matrix $(V_k \cdot S_k)$ replaces the matrix $(V \cdot S)$ in Eq. (6). The ith row of the matrix $(V_k \cdot S_k)$ is a k-dimensional vector, which represents the ith XML tree in the given set $\mathcal{S}$. Ideally, the value k is selected in such a way that the sum of the k largest eigenvalues contributes to roughly 85% of the sum of all eigenvalues of the covariance matrix $X \cdot X^T$.[26]

4. Clustering Experiments

4.1. *Data*

We conducted a series of experiments to evaluate the effectiveness of the proposed approach. The data used were two sets of XML documents taken from Wisconsin's XML data bank.[27] The first set contained 870 articles published in ACM SIGMOD conferences between 1993 and 1999. The second set contained XML documents

for 240 movies. All the documents were parsed into ordered labeled trees using the parser developed by Zhang *et al.*[6] For lengthy attributes/values of the XML elements such as the abstracts in the SIGMOD conference papers, we selected 4 to 10 keywords depending on the length of the attribute values and used the keywords to represent the original content.[b]

Each of the document sets was associated with a DTD file used to define the structure of the XML trees in that document set. This DTD file was not considered in our experiments. Our focus was to cluster XML documents within each document set that shared the same DTD.[c]

We used the three methods described in Section 2 to extract features from the XML documents, namely the path-based, nodepair-based, and hybrid method. For the path-based method, we extracted paths from an XML tree with lengths less than or equal to a user-specified parameter value L and used the paths as the features for the corresponding XML document. For the nodepair-based method, we extracted node pairs of types ranging from 1 to a user-specified parameter value K and used the node pairs as the features for the XML document. The hybrid method combined both the paths and the node pairs and used all of them to represent the XML document.

Table 1 summarizes the parameters and their default values used in the experiments. In practice, it is hard to determine the exact number for k used to construct the approximation, X_k, of X, cf. Eq. (7). Instead, in the experiments we used a parameter p—after choosing the k largest eigenvalues of the covariance matrix $X \cdot X^T$, the sum of the k largest eigenvalues is equal to $p \times G$, where G is the sum of all eigenvalues of $X \cdot X^T$.

Table 1. Parameters in our algorithms, their meanings and default values used in the experiments.

Parameter	Meaning	Value
L	the maximum length for a path	3
K	the maximum type for a node pair	4
p	ratio of the sum of the k largest eigenvalues over the sum of all eigenvalues	0.85

Table 2 shows some statistics concerning the data used in the experiments based on the default parameter values given in Table 1. For each feature extraction method described in Section 2, Table 2 lists the smallest and largest number of features

[b]Our experimental results showed that the number of keywords chosen to represent the content was insensitive to the relative performance of the proposed algorithms. Changing the number of keywords yielded the same qualitative conclusion concerning the algorithms.

[c]It is straightforward to cluster XML trees constructed by different DTDs because the extracted features would be significantly distinctive between trees defined by different DTDs. The resulting tree-vectors would be perpendicular to each other in calculating their similarities using the dot product distance. Consequently the clustering result would be always 100% correct.

extracted from the XML trees in each document set. For example, consider the nodepair-based method and SIGMOD conference papers. The number of distinct node pairs extracted by this method from the conference papers ranged from 70 to 220.

Table 2. Statistics concerning the XML trees and features extracted from these trees used in the experiments.

XML documents	Number of trees	Nodepair-based method	Path-based method	Hybrid method
SIGMOD papers	870	[70, 220]	[40, 110]	[110, 330]
Movie documents	240	[140, 160]	[80, 100]	[220, 260]

The 240 movie documents were categorized into six groups based on how they are displayed in the blockbuster video store. These six groups included (1) *action/adventure/war*, (2) *comedy/drama*, (3) *horror/mystery/thriller*, (4) *scientific fiction*, (5) *musical*, and (6) *others*. The 870 SIGMOD conference papers were categorized into 45 clusters. Papers in the same cluster shared a similar category, subject description, terms and phrases according to ACM classification.

4.2. *Performance Measure*

To evaluate the effectiveness of the proposed approach, we did experiments by performing clustering with and without the PCA method. The clustering algorithm used in the experiments was the K-means method.[20] The distance between two vectors in Euclidean space is measured by their dot product. To reduce the impact of the lengths of the vectors when calculating their dot product, we normalize each vector Z and denote its normalized vector as Z_{norm} where

$$Z_{norm} = \frac{Z}{||Z||} \tag{8}$$

$||Z||$ is the length of Z. Thus, after normalization, every vector becomes a unit vector. The dot product of two normalized vectors equals their cosine value.

We denote each group of documents as D_i, $1 \leq i \leq J$. Here, J is 6 for the movie documents and 45 for the conference papers. We mix all the documents and then use the K-means method to cluster the documents based on their vector representations. The clustering process also produces J groups, denoted by C_i, $1 \leq i \leq J$. For each unordered pair of distinct documents $m, n \in C_i$, define

$$F(m, n, C_i) = \begin{cases} 0 & \text{if } \exists j, m, n \in D_j \\ 1 & \text{if } \nexists j, m, n \in D_j \end{cases} \tag{9}$$

The number of clustering errors, denoted ε_b, is calculated as follows:

$$\varepsilon_b = \sum_{i=1}^{J} \sum_{\substack{m,n \in C_i \\ m \neq n}} F(m, n, C_i) \tag{10}$$

It is obvious that when a cluster C_i is equal to or is a subset of one of the predefined clusters D_j, then C_i contributes nothing to ε_b. On the other hand, if no pair of the members of C_i comes from the same predefined cluster, then the amount contributed by C_i to ε_b is $\binom{|C_i|}{2}$.

Finally, we define the error rate ε to be:

$$\varepsilon = \frac{\varepsilon_b}{\sum_{i=1}^{J} \binom{|C_i|}{2}} \times 100\% \tag{11}$$

Clearly, $0 \leq \varepsilon \leq 1$. This ε will be used as the performance measure to evaluate the effectiveness of the proposed clustering techniques.

4.3. *Results*

Figure 4 shows the error rate as a function of L with the PCA method and the path-based feature extraction algorithm. The other parameter values are as shown in Table 1. It can be seen that the weighted model outperforms the boolean model for both conference papers and movie documents. When L is too small (e.g. $L = 1$), too many short non-discriminative paths are generated, which dominate the feature set. The effect of discriminative paths is reduced, yielding a high error rate. On the other hand, when L is too large (e.g. $L = 6$), many long paths are generated, which are shared by few document trees. These long paths contribute little to clustering while weakening the effect made by discriminative features. It was also found that there is no difference between subpaths extracted from root-beginning paths and those extracted from leaf-ending paths.

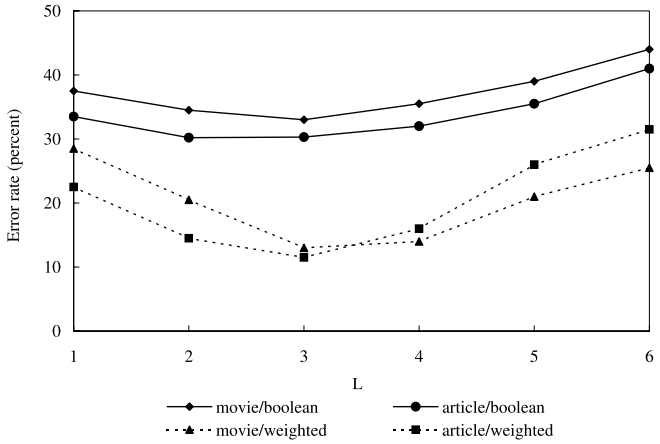

Fig. 4. Effect of L.

Figure 5 shows the error rate as a function of K with the PCA method and the nodepair-based feature extraction algorithm. The other parameter values are as

shown in Table 1. Again, when K is too small or too large, the performance is poor. For example, for a small K, an extracted feature would be a node pair (author, name), which helps little in the clustering process. On the other hand, for a large K, an extracted feature would be (author, year), which doesn't help either. More discriminative features would be node pairs of a reasonably large K type (e.g. $K = 4$), such as (Jack Harris, John Smith) that includes two co-authors of an article.

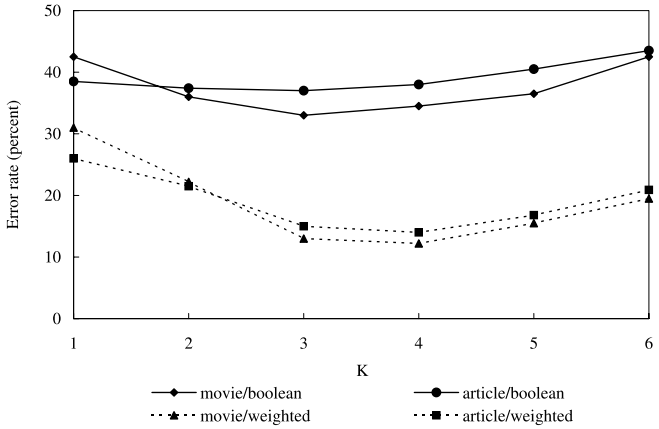

Fig. 5. Effect of K.

Figure 6 shows the impact of p for the PCA method with the nodepair-based feature extraction algorithm. Again, the weighted model is superior to the boolean model. The best performance is achieved when p is approximately 0.85, i.e., when the sum of the k largest eigenvalues of the covariance matrix $X \cdot X^T$ is $0.85 \times G$, where G is the sum of all eigenvalues of $X \cdot X^T$. This result is consistent with the discussion concerning the selection of the best k value for PCA given in Section 3. A similar result was obtained for the PCA method with the path-based feature extraction algorithm.

Figure 7 compares the proposed approach employing the PCA method with the approach without using PCA for the movie documents. Figure 8 shows the result for the conference papers. The parameter values used by the algorithms are as shown in Table 1. In the figures, the white bar represents our approach with PCA and the shaded bar represents the approach without PCA.

The figures show that the PCA method significantly improves clustering performance. This happens because the discriminative information in the XML documents is mainly associated with the principal components of the matrix X in Eq. (1). Thus, when PCA is used to highlight the discriminative features while at the same time eliminating ambiguous portions (i.e. those with eigenvalues approaching 0), the performance of the clustering is improved. Notice also that for the movie documents, the path-based method yields the best performance. On the other hand, for the

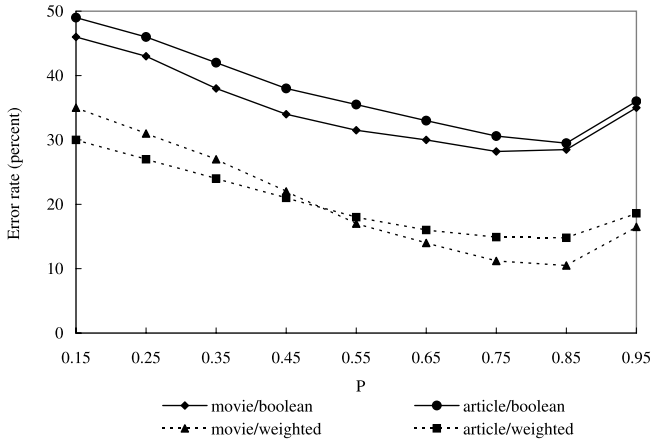

Fig. 6. Effect of P.

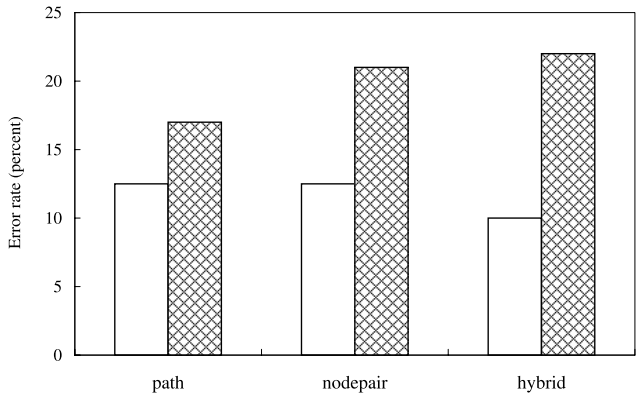

Fig. 7. Comparison between our approach with PCA and the approach without PCA for movie documents.

conference papers, the hybrid method is best. This happens due to the inherent structural characteristics existing in the different document sets. In our case, the discriminative information for the conference papers is distributed to both paths and node pairs of the XML trees for the articles. On the other hand, for the movie documents, the discriminative information is mainly associated with the paths of the document trees.

5. XML Retrieval

Besides clustering, PCA can be applied to structured-based retrieval for XML documents.[28,29,30] Given a query tree Q and a database of XML trees $\mathcal{D}$, the goal is to find XML trees in the database that are "closest" to the query tree and rank those XML trees based on their closeness to the query tree. The closeness here is

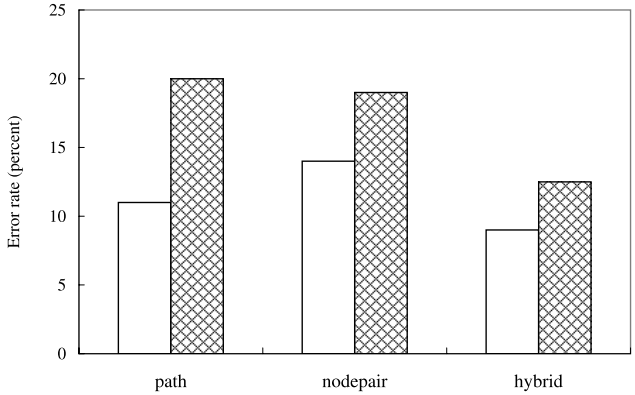

Fig. 8. Comparison between our approach with PCA and the approach without PCA for conference papers.

measured by the similarity score calculated through the PCA method.

Specifically, the retrieval process consists of two phases. In the first phase, which is the pre-processing phase, we extract features from the XML document trees in the database $\mathcal{D}$ using the feature extraction algorithms described in Section 2. Let $\mathcal{E}$ contain the distinct features extracted from the XML document trees in $\mathcal{D}$. Let N be the size of the database $\mathcal{D}$ and let C be the size of $\mathcal{E}$. We build the $C \times N$ matrix X for the database $\mathcal{D}$ as described in Section 3.

We then project each vector onto the subspace spanned by the $C \times k$ matrix U_k shown in Eq. (7), which consists of k eigenvectors corresponding to the k largest eigenvalues of the covariance matrix $X \cdot X^T$. The projected result gives a new representation of the original XML tree within the subspace spanned by the matrix U_k. Let X_k be the new matrix containing the projected vectors. We obtain

$$X_k = U_k^T \cdot X \tag{12}$$

Each column of X_k corresponds to a projected tree-vector with dimension k.

In the second phase, which is the online search phase, the query tree Q is given, and this tree is first transformed to a vector based on the weighted model described in Section 3 by checking which features in $\mathcal{E}$ exist in Q. Call the resulting vector $Q^{'}$. We then project $Q^{'}$ onto the same k-dimensional subspace spanned by the matrix U_k. The representation of Q in that k-dimensional space is $U_k^T \cdot Q^{'}$.

Then we calculate the similarity scores between the query tree Q and the XML trees in the database $\mathcal{D}$ using the formula:

$$S = (U_k^T \cdot Q^{'})^T \cdot (U_k^T \cdot X) \tag{13}$$

where S contains all the similarity scores, in which the nth component is the similarity score between Q and the nth tree in the database $\mathcal{D}$. The XML trees in $\mathcal{D}$ can be sorted based on their similarity scores to Q stored in the matrix S. The

XML trees with larger similarity scores are ranked higher than those with smaller similarity scores when search results are displayed.

We have implemented the above algorithms into an XML retrieval system. Figure 9 shows a query and search results for retrieving XML movie documents. Here, the weighted model and nodepair-based method with the default parameter values in Table 1 were used for transforming an XML document tree to a vector. In the left, upper window in Figure 9, an XML query is displayed which is to find all XML documents in the database in which Orson Welles is an actor. Four search results are displayed, with the XML document closest to the query ranked highest. In the figure, three movie documents in which both Orson Welles and Laurence Olivier are actors are shown in the three windows on the right side of the screen. These movie documents are ranked higher than the movie documents in which Orson Welles, but not Laurence Olivier, is an actor. (One of the latter documents is displayed in the left, lower window.) This happens because Orson Welles and Laurence Olivier co-occur frequently in the XML movie documents in the database. These co-occurring features are captured by PCA and consequently the documents containing both actors are very similar to the query.

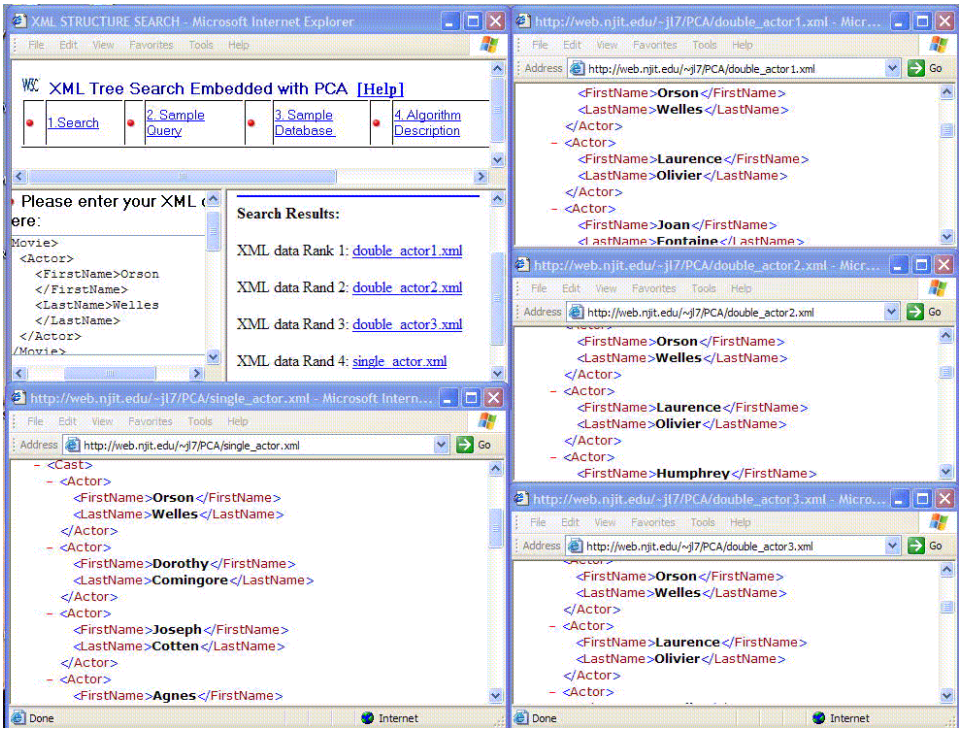

Fig. 9. A query and search results for retrieving XML movie documents.

Figure 10 shows a query and search results for retrieving SIGMOD conference papers. The query is to find all SIGMOD conference papers in which Jason Tsong-Li Wang is a co-author. Since Jason Tsong-Li Wang and Dennis Shasha have co-published several papers, those papers with both Wang and Shasha are ranked highest (shown in the three windows on the right side of the screen). Interestingly, the search results also contain papers in which Shasha, but not Wang, is a co-author (one such result is displayed in the left, lower window). These papers are ranked higher than those papers in which neither Wang nor Shasha is a co-author.

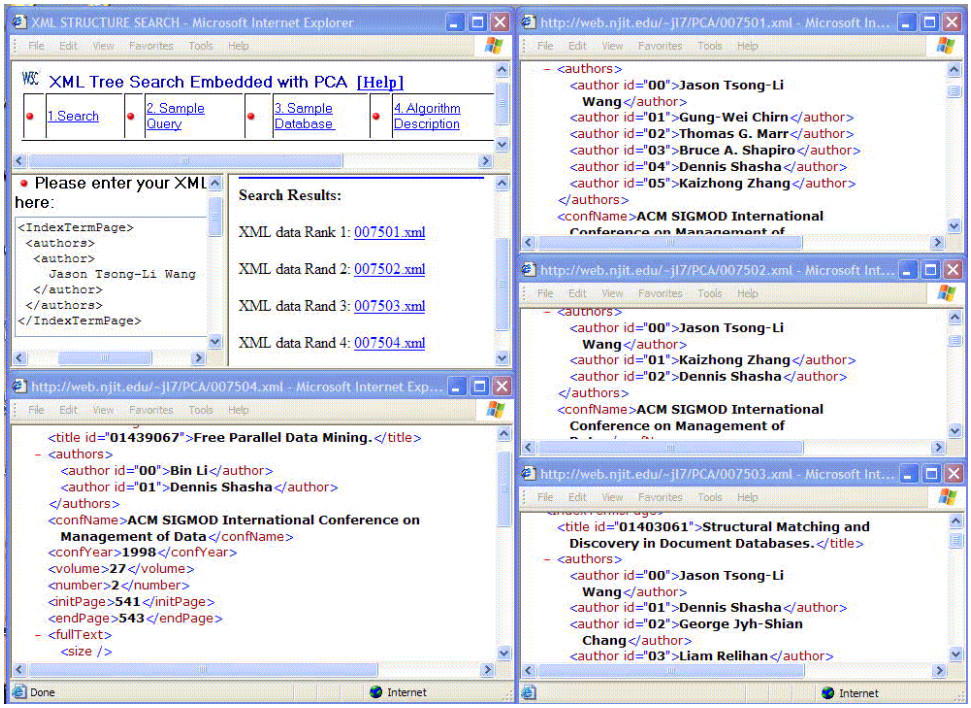

Fig. 10. A query and search results for retrieving XML conference papers.

In summary, with the PCA method, we project each XML vector into a subspace spanned by the k eigenvectors corresponding to the k largest eigenvalues of the covariance matrix $X \cdot X^T$. This way, the significantly discriminative features in the XML documents are captured by PCA while redundant and trivial features are ignored. Here, the redundant features refer to those having linear correlations among them. The trivial features refer to those that cannot contribute to separating one vector from another. In contrast to existing approaches to structure-based

retrieval,[28,29,30] the proposed approach measures the closeness of XML trees based on the dot product of their corresponding vectors in a high-dimensional Euclidean space and therefore is capable of finding documents with co-occurrent features.

6. Conclusion and Future Work

In this paper, we have proposed a new approach to clustering XML documents. Our method is based on the tree representation of the XML documents and extracting features including paths and node pairs from the XML trees. We then transform the XML trees to vectors in a high-dimensional Euclidean space according to the occurrences of the features in the trees. By applying the principal component analysis (PCA) method to the data, our approach identifies significant dimensions in the vectors and clusters the vectors in the subspace spanned by the significant dimensions. The distance function employed in the PCA method measures the similarity between two vectors based on the co-occurrence of features in the vectors. Thus, two documents with many co-occurrent features are more similar than those with few co-occurrent features. Experimental results showed that the PCA method enhances the accuracy of clustering. We also discussed an extension of the proposed approach to XML retrieval.

Our future work includes (i) applying our techniques to other types of tree-structured data such as phylogenies, and (ii) developing new feature extraction algorithms, possibly based on domain-specific substructures in trees, to improve the effectiveness of clustering and retrieval. For example, Yoon *et al.*[31] recently proposed to use bitmap indexing in combination with XML-element paths, terms and words to cluster XML documents. Pluempitiwiriyawej and Hammer[32] considered XML documents as complex objects and defined a set of distance functions for their sub-elements and attribute values. The authors then used hierarchical clustering methods to cluster the XML documents based on these distance functions. When compared to the two existing methods, our approach yields slightly higher error rates. However, by including all node labels of an XML tree and using these labels as features, the proposed PCA method outperforms the two previous methods. We expect even better performance can be achieved by combining the PCA techniques with domain-specific knowledge concerning XML documents.

Acknowledgments

This work was supported in part by NSF grant IIS-9988636. We thank Sen Zhang for his efforts spent in implementing the XML retrieval system and for his help in generating the screenshots in Figures 9 and 10.

References

1. N. Mamoulis, Mining frequent patterns in XML documents, http://www.csis.hku.hk/~minexml/Progress/Plan/unordtree.shtml.

2. C. Moh, E. Lim and W. Ng, DTD-Miner: A tool for mining DTD from XML documents, in *Proceedings of the 2nd International Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems* (2000).
3. R. Nayak, R. Witt and A. Tonev, Data mining and XML documents, in *Proceedings of the International Conference on Internet Computing* (2002).
4. W3C recommendation, http://www.w3.org/TR/REC-xml.
5. M. J. Zaki and C. Aggarwal, XRules: An effective structural classifier for XML data, in *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2003).
6. S. Zhang, J. T. L. Wang and K. G. Herbert, XML query by example, *International Journal of Computational Intelligence and Applications* **2**(2002) 329–337.
7. H. V. Jagadish, L. V. S. Lakshmanan, D. Srivastava and K. Thompson, TAX: A tree algebra for XML, in *Proceedings of the International Workshop on Database Programming Languages* (2001), pp. 149–164.
8. Z. Chen, H. V. Jagadish, F. Korn, N. Koudas, D. Srivastava, R. Ng and S. Muthukrishnan, Counting twig matches in a tree, in *Proceedings of the 17th International Conference on Data Engineering* (2001).
9. M. Garofalakis and A. Kumar, Correlating XML data streams using tree-edit distance embeddings, in *Proceedings of the 22nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems* (2003).
10. D. Shasha, J. T. L. Wang and R. Giugno, Algorithmics and applications of tree and graph searching, in *Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems* (2002), pp. 39–52.
11. K. Zhang and D. Shasha, Simple fast algorithms for the editing distance between trees and related problems, *SIAM J. Computing* **18**(1989) 1245–1262.
12. D. Chase, An improvement to bottom-up tree pattern matching, in *Proceedings of the 14th Annual ACM Symposium on Principles of Programming Languages* (1987), pp. 168–177.
13. R. Cole, R. Hariharan and P. Indyk, Tree pattern matching and subset matching in deterministic $o(nlog^3m)$ time, in *Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms* (1999), pp. 245–254.
14. M. Dubiner, Z. Galil and E. Magen, Faster tree pattern matching, *J. ACM* **14**(1994) 205–213.
15. P. Kilpelainen and H. Mannila, Ordered and unordered tree inclusion, *SIAM J. Computing* **24**(1995) 340–356.
16. R. Shamir and D. Tsur, Faster subtree isomorphism, *J. Algorithms* **33**(1999) 267–280.
17. D. Shasha, J. T. L. Wang, H. Shan and K. Zhang, ATreeGrep: Approximate searching in unordered trees, in *Proceedings of the 14th International Conference on Scientific and Statistical Database Management* (2002), pp. 89–98.
18. K. G. Herbert, J. T. L. Wang and J. Liu, Information retrieval and data mining, in *Handbook of Computer Science and Engineering*, 2nd edn., ed. A. B. Tucker (CRC Press, 2004), pp. 75.1–75.16.
19. M. L. Lee, L. Yang, W. Hsu and X. Yang, XClust: Clustering XML schemas for effective integration, in *Proceedings of the 11th ACM International Conference on Information and Knowledge Management* (2002).
20. J. Han and M. Kamber, *Data Mining: Concepts and Techniques* (Morgan Kaufmann Publishers, San Francisco, 2001).
21. C. Faloutsos and K.-I. Lin, FastMap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets, in *Proceedings of the ACM SIGMOD International Conference on Management of Data* (1995), pp. 163–174.

22. J. T. L. Wang, X. Wang, K.-I. Lin, D. Shasha, B. A. Shapiro and K. Zhang, Evaluating a class of distance-mapping algorithms for data mining and clustering, in *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (1999), pp. 307–311.
23. X. Wang, J. T. L. Wang, K.-I. Lin, D. Shasha, B. A. Shapiro and K. Zhang, An index structure for data mining and clustering, *Knowledge and Information Systems* **2**(2000) 161–184.
24. G. Golub and C. V. Loan, *Matrix Computation* (Johns Hopkins University Press, Baltimore, 2nd edn., 1989).
25. C. H. Q. Ding, A similarity-based probability model for latent semantic indexing, in *Proceedings of the 22nd ACM SIGIR Conference on Research and Development in Information Retrieval* (1999), pp. 58–65.
26. R. C. Gonzales and C. Rafael, *Digital Image Processing* (Addison-Wesley Publishing Company, 1993).
27. Niagara query engine, http://www.cs.wisc.edu/niagara/data.html.
28. H. Shan, K. G. Herbert, W. Piel, D. Shasha and J. T. L. Wang, A structure-based search engine for phylogenetic databases, in *Proceedings of the 14th International Conference on Scientific and Statistical Database Management* (2002), pp. 7–10.
29. J. T. L. Wang, H. Shan, D. Shasha and W. H. Piel, TreeRank: A similarity measure for nearest neighbor searching in phylogenetic databases, in *Proceedings of the 15th International Conference on Scientific and Statistical Database Management* (2003).
30. J. T. L. Wang, X. Wang, D. Shasha, B. A. Shapiro, K. Zhang, Q. Ma and Z. Weinberg, An approximate search engine for structural databases, in *Proceedings of the ACM SIGMOD International Conference on Management of Data* (2000).
31. J. Yoon, V. Raghavan and V. Chakilam, Bitmap indexing-based clustering and retrieval of XML documents, in *Proceedings of the ACM SIGIR Workshop on Mathematical/Formal Methods in IR* (2001).
32. C. Pluempitiwiriyawej and J. Hammer, Element matching across data-oriented XML sources using a multi-strategy clustering model, *Data and Knowledge Engineering* **48**(2004) 297–333.