A Graph-Based Framework for Web Document Mining

Adam Schenker[1], Horst Bunke[2], Mark Last[3], and Abraham Kandel[1,4]

[1] University of South Florida, Tampa FL 33620, USA
[2] University of Bern, CH-3012 Bern, Switzerland
[3] Ben-Gurion University of the Negev, Beer-Sheva 84105, Israel
[4] Tel-Aviv University, Tel-Aviv 69978, Israel

Abstract. In this paper we describe methods of performing data mining on web documents, where the web document content is represented by graphs. We show how traditional clustering and classification methods, which usually operate on vector representations of data, can be extended to work with graph-based data. Specifically, we give graph-theoretic extensions of the k-Nearest Neighbors classification algorithm and the k-means clustering algorithm that process graphs, and show how the retention of structural information can lead to improved performance over the case of the vector model approach. We introduce several different types of web document representations that utilize graphs and compare their performance for clustering and classification.

1 Introduction

Web document mining [1] is the application of data mining techniques to web-related documents. Web mining methodologies can generally be classified into one of three categories: web usage mining, web structure mining, and web content mining [2]. In web usage mining the goal is to examine web page usage patterns in order to learn about a web system's users or the relationships between the documents. Web usage mining is useful for providing personalized web services, an active area of web mining research. In the second category of web mining methodologies, web structure mining, only the relationships between web documents are examined, usually by utilizing the information conveyed by each document's hyperlinks. Like web usage mining, the actual content of the web pages is often ignored.

In the current paper we are concerned with the third category of web mining, web content mining. In web content mining we examine the actual content of web pages (most often the text contained in the pages) and then perform some web mining procedure, most typically clustering or classification. Content-based classification of web documents is useful because it allows users to more easily navigate and browse collections of documents [3][4]. Such classifications are often costly to perform manually, as it requires a human expert to examine the content of each web document. Due to the large number of documents available on the Internet in general, or even when we consider smaller collections of web documents, such as those associated with corporate or university web sites, an automated system which performs web document classification is desirable in order to reduce costs and increase the speed with which new documents are classified. Clustering is an unsupervised method which attempts to separate web documents into similar groups, while classification is a supervised learning technique which aims to assign a specific label to each web document. Clustering is done to organize web documents into related groups. This has benefits when the classes are not known a priori, such as in web search engines [5], since it allows systems to display results grouped by cluster (topic), in comparison to the usual "endless" ranked list, making browsing easier for the user. Using classification techniques with these types of systems is difficult due to the highly dynamic nature of the Internet; creating and maintaining a training set would be challenging and costly. For this reason, it is necessary to create clusters automatically from the data.

Traditional information retrieval and data mining methods represent documents with a vector model, which utilizes a series of numeric values associated with each document. Each value is associated with a specific term (word) that may appear on a document, and the set of possible terms is shared across all documents. The values may be binary, indicating the presence or absence of the corresponding term. The values may also be non-negative integers, which represent the number of times a term appears on a document (i.e. term frequency). Non-negative real numbers can also be used, in this case indicating the importance or weight of each term. These values are derived through a method such as the popular inverse document frequency model [6], which reduces the importance of terms that appear on many documents. Regardless of the method used, each series of values represents a document and corresponds to a point (i.e. vector) in a Euclidean feature space; this is called the vector-space model of information retrieval. This model is often used when applying data mining techniques to documents, as there is a strong mathematical foundation for performing distance measure and centroid calculations using vectors. However, this method of document representation does not capture important structural information, such as the order and proximity of term occurrence, or the location of term occurrence within the document.

In order to overcome this problem we have introduced several methods of representing web document content using graphs instead of vectors, and have extended existing data mining methods to work with these graphs. These approaches have two main benefits: 1. they allow us to keep the inherent structural information of the original document without having to discard information as we do with the vector model representation, and 2. they intuitively extend existing, well-known data mining algorithms rather than create new algorithms, whose properties and behavior are unknown.

Only recently have a few papers appeared in the literature that deal with graph representations of documents. Lopresti and Wilfong compare web documents using a graph representation that primarily utilizes HTML parse information, in addition to hyperlink and content order information [7]. In their approach they use *graph probing*, which extracts numerical feature information from the graphs, such as node degrees or edge label frequencies, rather than comparing the graphs themselves. In contrast, our representation uses graphs created solely from the content, and we use the graphs themselves rather than a set of extracted features. Liang and Doermann represent the physical layout of document images as graphs [8]. In their *layout graphs* nodes represent elements on the page of a document, such as columns of text or headings, while edges indicate how these elements appear together on the page (i.e. spatial relationships). This method is based on the formatting and appearance of the documents when rendered, not the textual content (words) of a document as in our approach.

Earlier work by the authors dealing with graph-based classification [9][10] and clustering [11][12] of web documents has been presented. In this paper we aim to give a synopsis of our previous work and a general framework for web document mining using a graph model of web document content. We also introduce a new graph representation model and compare it to the ones previously proposed. The paper is organized as follows. In Sect. 2 we will describe several techniques for representing web document content by graphs. We demonstrate how the popular k-Nearest Neighbors classification method can be easily extended to work with these graphs in Sect. 3. Similarly, in Sect. 4, we explain the graph-theoretic extension of the k-means clustering algorithm. Experimental results for both methods comparing the graph and vector approaches are presented in Sect. 5. Finally, some concluding remarks are provided in Sect. 6.

2 Graph Representations of Web Documents

In this section we describe methods for representing web documents using graphs instead of the traditional vector representations. All representations are based on the adjacency of terms in a web document. These representations are named: *standard*, *simple*, *n-distance*, *n-simple distance*, *raw frequency* and *normalized frequency*.

Under the *standard* method each unique term (word) appearing in the document, except for *stop words* such as "the", "of", and "and" which convey little information, becomes a node in the graph representing that document. Each node is labeled with the term it represents. Note that we create only a single node for each word even if a word appears more than once in the text. Second, if word a immediately precedes word b somewhere in a "section" s of the document, then there is a directed edge from the node corresponding to term a to the node corresponding to term b with an edge label s. We take into account certain punctuation (such as periods) and do not create an edge when these are present between two words. Sections we have defined for the standard representation are: *title*, which contains the text related to the document's title and any provided keywords (meta-data); *link*, which is text that appears in hyper-links on the document; and *text*, which comprises any of the visible text in the document (this includes text in links, but not text in the document's title and keywords). Next we remove the most infrequently occurring words on each document, leaving at most m nodes per graph (m being a user provided parameter). This is similar to the dimensionality reduction process for vector representations [6]. Finally we perform a simple stemming method and conflate terms to the most frequently occurring form by re-labeling nodes and updating edges as needed. An example of this type of graph representation is given in Fig. 1. The ovals indicate nodes and their corresponding term labels. The edges are labeled according to title (TI), link (L), or text (TX). The document represented by the example has the title "YAHOO NEWS", a link whose text reads "MORE NEWS", and text containing "REUTERS NEWS SERVICE REPORTS". Note there is no restriction on the form of the graph and that cycles are allowed. While this approach to document representation appears superficially similar to the bigram, trigram, or N-gram methods, those are statistically-oriented approaches based on word occurrence probability models [13]. The methods presented here, with the exception of the frequency representations described below, do not require or use the computation of term probability relationships.

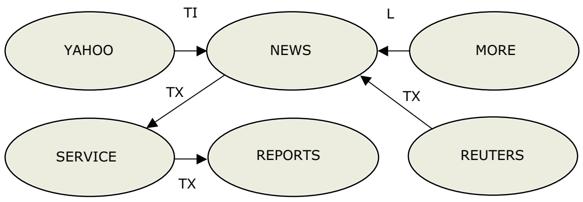

Fig. 1. Example of a *standard* graph representation of a document.

The second type of graph representation we will look at is what we call the *simple* representation. It is basically the same as the standard representation, except that we look at only the visible text on the page, and do not include title and meta-data information (the *title* section). Further, we do not label the edges between nodes so there is no distinction between *link* and *text* sections. An example of this type of representation is given in Fig. 2.

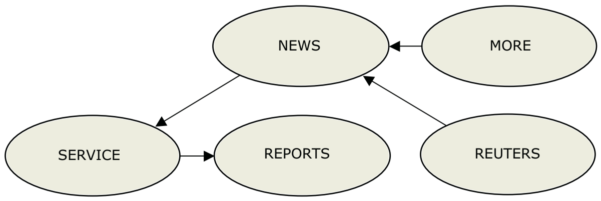

Fig. 2. Example of a *simple* graph representation of a document.

The third type of representation is called the *n-distance* representation. Under this model, there is a user-provided parameter, n. Instead of considering only terms immediately following a given term in a web document, we look up to n

terms ahead and connect the succeeding terms with an edge that is labeled with the distance between them (unless the words are separated by certain punctuation marks). For example, if we had the following text on a web page, "AAA BBB CCC DDD", then we would have an edge from term AAA to term BBB labeled with a 1, an edge from term AAA to term CCC labeled 2, and so on. The complete graph for this example is shown in Fig. 3.

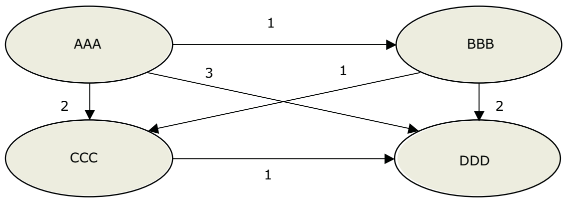

Fig. 3. Example of an *n-distance* graph representation.

Similar to n-distance, we also have the fourth graph representation, *n-simple distance*. This is identical to n-distance, but the edges are not labeled, which means we only know that the distance between two connected terms is not more than n.

The fifth graph representation is what we call the *raw frequency* representation. This is similar to the simple representation (adjacent words, no section-related information) but each node and edge is labeled with an additional frequency measure. For nodes this indicates how many times the associated term appeared in the web document; for edges, this indicates the number of times the two connected terms appeared adjacent to each other in the specified order. The raw frequency representation uses the total number of term occurrences (on the nodes) and co-occurrences (edges).

A problem with this representation is that large differences in document size could lead to skewed comparisons, similar to the problem encountered when using Euclidean distance with vector representations of documents. Under the *normalized frequency* representation, instead of associating each node with the total number of times the corresponding term appears in the document, a normalized value in $[0,1]$ is assigned by dividing each node frequency value by the maximum node frequency value that occurs in the graph; a similar procedure is performed for the edges. Thus each node and edge has a value in $[0,1]$ associated with it, which indicates the normalized frequency of the term (for nodes) or co-occurrence of terms (for edges).

3 *k*-Nearest Neighbors Classification with Graphs

In this section we describe the k-Nearest Neighbors (k-NN) classification algorithm and how we can easily extend it to work with the graph-based representations of web documents described above. The basic k-NN algorithm is given as follows [14]. First, we have a set of training examples; in the traditional k-NN approach these are numerical feature vectors. Each of these training instances is associated with a label which indicates to what class the instance belongs. Given a new, previously unseen instance, called an input instance, we attempt to estimate which class it belongs to. Under the k-NN method this is accomplished by looking at the k training instances closest (i.e. with least distance) to the input instance. Here k is a user provided parameter and distance is usually defined to be the Euclidean distance:

$$dist_{EUCL}(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \ , \tag{1}$$

where x_i and y_i are the ith components of vectors $x = [x_1, x_2, \ldots, x_n]$ and $y = [y_1, y_2, \ldots, y_n]$, respectively. However, for applications in document classification, the cosine or Jaccard similarity measures [6] are often used due to their length invariance property. We can convert these to a distance measure by the following:

$$dist_{COS}(x,y) = 1 - \frac{x \bullet y}{\|x\| \cdot \|y\|} \ , \tag{2}$$

$$dist_{JAC}(x,y) = 1 - \frac{\sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n} x_i^2 + \sum_{i=1}^{n} y_i^2 - \sum_{i=1}^{n} x_i y_i} \ . \tag{3}$$

In Eq. (2) $\bullet$ indicates the dot product operation and $\| \cdots \|$ indicates the magnitude (length) of a vector.

Once we have found the k nearest training instances using some distance measure, such as one of those defined above in Eqs. (1–3), we estimate the class by the majority among the k instances. This class is then assigned as the predicted class for the input instance. If there are ties due to more than one class having equal numbers of representatives amongst the nearest neighbors we can either choose one class randomly or we can break the tie with some other method, such as selecting the tied class which has the minimum distance neighbor. For the experiments in this paper we will use the latter method, which in our experiments has shown a slight improvement over random tie breaking. In order to extend the k-NN method to work with graphs instead of vectors, we only need a distance measure which computes the distance between two graphs instead of two vectors. This graph-theoretic distance measure, which utilizes the concept of the maximum common subgraph, is [15]:

$$dist_{MCS}(G_1,G_2) = 1 - \frac{|mcs(G_1,G_2)|}{\max(|G_1|,|G_2|)} \ , \tag{4}$$

where G_1 and G_2 are graphs, $mcs(G_1, G_2)$ is their maximum common subgraph, $\max(\cdots)$ is the standard numerical maximum operation, and $| \cdots |$ denotes the size of the graph. Usually, this is taken to be the sum of the number of nodes and edges in the graph. However, for the frequency-based graph representations described in Sect. 2, the graph size is defined as the sum of the node frequency values added to the sum of the edge frequency values. Other similar graph distance measures have been proposed as well [16][17].

For general graphs the computation of the maximum common subgraph is NP-Complete. However, for the graph representations of web documents presented in Sect. 2, the computation of the maximum common subgraph is $O(n^2)$, with n being the number of nodes, due to the existence of unique node labels in the graph representations (i.e. we need only examine the intersection of the nodes, since each node has a unique label) [18]. We will give some examples of actual run times when using the graph-based methods in Sect. 5.

4 *k*-Means Clustering with Graphs

The k-means clustering algorithm is a simple and straightforward method for clustering data [14]. The basic algorithm is given in Fig. 4. As with the k-NN classification algorithm, the typical procedure is to represent each item to be clustered as a vector in the Euclidean space $\Re^m$. The distance measure used by the algorithm is usually one of Eqs. (1–3).

```
Inputs:   the set of $n$ data items and a parameter, $k$, defining the number of clusters to create
Outputs:  the centroids of the clusters and for each data item the cluster (an integer in [1,$k$]) it
          belongs to

Step 1.   Assign each data item randomly to a cluster (from 1 to $k$).
Step 2.   Using the initial assignment, determine the centroids of each cluster.
Step 3.   Given the new centroids, assign each data item to be in the cluster of its closest centroid.
Step 4.   Re-compute the centroids as in Step 2. Repeat Steps 3 and 4 until the centroids do not
          change.
```

Fig. 4. The basic k-means clustering algorithm.

As we saw in Sect. 3, we have available to us a method of computing distances between graphs [Eq. (4)]. However, note that the k-means algorithm, like many clustering algorithms, requires not only the computation of distances, but also of cluster representatives. In the case of k-means, these representatives are called centroids. Thus we need a graph-theoretic version of the centroid, which itself must be a graph, if we are to extend this algorithm to work with graph representations of web documents. Our solution is to compute the representatives (centroids) of the clusters using *median graphs* [19]. The median of a set of graphs is defined as the graph from the set which has the minimum average distance to all the other graphs in the set. Here the distance is computed with the graph-theoretic distance measure mentioned above [Eq. (4)].

We wish to clarify a point that may cause some confusion. Clustering with graphs is well established in the literature. However, with those methods the entire clustering problem is treated as a graph, where nodes represent the items to be clustered and weights on edges connecting nodes indicate the distance between the objects the nodes represent. The goal is to partition this graph, breaking it up into several connected components which represent clusters. The usual procedure is to create a minimal spanning tree of the graph and then remove the remaining edges with the largest weight until the number of desired clusters is achieved [20]. This is very different than the technique we described in this section, since it is the data (in this case, the web documents) themselves which are represented by graphs, not the overall clustering problem.

5 Experiments and Results

In order to evaluate the performance of the graph-based k-NN and k-means algorithms as compared with the traditional vector methods, we performed experiments on two different collections of web documents, called the *F-series* and the *J-series* [21][1]. These two data sets were selected because of two major reasons. First, all of the original HTML documents are available, which is necessary if we are to represent the documents as graphs; many other document collections only provide a pre-processed vector representation, which is unsuitable for use with our method. Second, ground truth assignments are provided for each data set, and there are multiple classes representing easily understandable groupings that relate to the content of the documents. Some web document collections are not labeled or are presented with some other task in mind than content-related classification (e.g. building a predictive model based on user preferences).

The F-series originally contained 98 documents belonging to one or more of 17 sub-categories of four major category areas: *manufacturing*, *labor*, *business & finance*, and *electronic communication & networking*. Because there are multiple sub-category classifications from the same category area for many of these documents, we have reduced the categories to just the four major categories mentioned above in order to simplify the problem. There were five documents that had conflicting classifications (i.e., they were classified to belong to two or more of the four major categories) which we removed in order to create a single class classification problem, which allows for a more straightforward way of assessing classification accuracy. The J-series contains 185 documents and ten classes: *affirmative action*, *business capital*, *information systems*, *electronic commerce*, *intellectual property*, *employee rights*, *materials processing*, *personnel management*, *manufacturing systems*, and *industrial partnership*. We have not modified this data set. Additional results on a third, larger data set can be found in [9][10][12].

For the vector-model representation experiments there were already several pre-created term–document matrices available for our experiments at the same location where we obtained the two document collections. We selected the matrices with the smallest number of dimensions. For the F-series documents there are 332 dimensions (terms) used, while the J-series has 474 dimensions. We performed some preliminary experiments and observed that other term-weighting

[1] The data sets are available under these names at: `ftp://ftp.cs.umn.edu/dept/users/boley/PDDPdata/`

schemes (i.e., $tf \cdot idf$, see [6]) improved the accuracy of the vector-model representation for these data sets either only very slightly or in many cases not at all. Thus we have left the data in its original format.

For our graph-based experiments we used a maximum graph size of 30 nodes per graph; this corresponds to setting $m = 30$ (see Sect. 2). This parameter value was selected based on previous experimental results, and has been shown to work adequately for both data sets (see also the study by Turney [22]). Further results are omitted for brevity. For the distance related graph representations, n-distance and n-simple distance, we used $n = 5$ (i.e. 5-distance and 5-simple distance). Classification accuracy was assessed by the leave-one-out method. Clustering performance is measured using two performance indices which indicate the similarity of obtained clusters to the "ground truth" clusters. The first performance index is the *Rand index* [23], which is computed by examining the produced clustering and checking how closely it matches the ground truth clustering. It produces a value in the interval $[0, 1]$, with 1 representing a clustering that perfectly matches ground truth. The second performance index we use is *mutual information* [24], which is an information-theoretic measure that evaluates the overall degree of agreement between the clustering under consideration and ground truth, with a preference for clusters that have high purity. Higher values of mutual information indicate better performance. The clustering experiments were repeated ten times to account for the random initialization of the k-means algorithm, and the average of these experiments is reported.

The results for the k-NN classification experiments are given in Table 1. The first column, *DS*, indicates which data set (F or J-series) the experiments were performed on. The next column, k, is the number of nearest neighbors used in that experiment. The next six columns are the results of using our graph-theoretic approach when utilizing the various types of graph representations of web documents described in Sect. 2. These are, from left to right: standard, simple, 5-distance, 5-simple distance, raw frequency, and normalized frequency. The final column is the accuracy of the vector representation approach using a distance based on Jaccard similarity [6], which is the best performing vector distance measure we have worked with. On our system, a 2.6 GHz Pentium 4 with 1 gigabyte of memory, the graph method took an average of 0.2 seconds to classify a document for the F-series, and 0.45 seconds for the J-series, both when using $k = 1$ and the standard representation.

Similarly, the result for the k-means clustering experiments are given in Table 2. The columns are identical to Table 1, with the exception of the second column, which indicates the performance measures (*PM*) used: Rand index (*R*) or mutual information (*MI*). The average time to create clusters for the F-series using the graph-based method and the standard representation was 22.7 seconds, while it took 59.5 seconds on average for the J-series.

From the results we see that the standard representation, in all experiments, exceeded the equivalent vector procedure. In 11 out of 12 experiments, the simple representation outperformed the vector model. The 5-distance representation was better in 8 out of 12 experiments. The 5-simple distance representation was

Table 1. Experimental results for k-NN classification. Classification accuracy is given as percent of documents correctly classified using leave-one-out. The best results of each experiment are shown in bold.

DS	k	Std	Sim	5-D	5-SD	RF	NF	Vec
F	1	96.77%	94.62%	92.47%	93.55%	93.55%	**97.85%**	93.55%
F	3	95.70%	**97.85%**	94.62%	96.77%	96.77%	96.77%	94.62%
F	5	**96.77%**	**96.77%**	93.55%	94.62%	93.55%	93.55%	88.17%
F	10	**95.70%**	94.62%	**95.70%**	92.47%	91.40%	92.47%	83.87%
J	1	81.08%	80.00%	**82.16%**	**82.16%**	81.08%	76.76%	73.51%
J	3	**84.86%**	81.62%	82.70%	83.78%	81.62%	80.00%	74.59%
J	5	85.41%	**85.95%**	83.78%	83.78%	80.00%	81.62%	74.05%
J	10	**85.41%**	84.86%	83.78%	82.16%	77.30%	84.32%	77.30%

Table 2. Experimental results for k-means clustering. Performance is measured with Rand index (R) and mutual information (MI). The best results of each experiment are shown in bold.

DS	PM	Std	Sim	5-D	5-SD	RF	NF	Vec
F	R	**0.7202**	0.7064	0.6912	0.7056	0.7186	0.7003	0.6899
F	MI	**0.1604**	0.1323	0.1371	0.1457	0.1404	0.1319	0.1020
J	R	**0.8741**	0.8692	0.8663	0.8605	0.8639	0.8660	0.8717
J	MI	0.2487	0.2400	0.2305	0.2107	0.2204	**0.2516**	0.2316

an improvement in 9 out of 12 cases. Raw frequency was better in 8 of 12 cases, while normalized frequency was an improvement in 11 of 12 cases.

For the classification experiments, the best accuracy for the F-series was 97.85%, which was achieved by both the simple representation (for $k = 3$) and the normalized frequency representation (for $k = 1$). In contrast, the best accuracy using a vector representation was 94.62% (for $k = 3$). For the J-series, the best graph-based accuracy was 85.95% (for simple, $k = 5$); the best vector-based accuracy was 77.30%.

For the clustering experiments, the best F-series results were attained by the standard representation (0.7202 for Rand index; 0.1604 for mutual information). The performance of the vector approach was 0.6899 and 0.1020 for Rand and mutual information, respectively. For the J-series, the best Rand index was obtained for standard (0.8741) while the best mutual information value was attained for normalized frequency (0.2516). In comparison, the vector-based clustering for the J-series achieved 0.8717 for Rand index and 0.2316 for mutual information.

6 Conclusions

In this paper we have provided a description of several methods of representing web document content as graphs, rather than the vector representations which are typically used. We introduced six different types of graph representations (standard, simple, n-distance, n-simple distance, raw frequency, and normalized frequency) and performed experiments with both classification using k-Nearest Neighbors and clustering using k-means on two web document collections. The results showed an improvement over the traditional vector representation in most cases for both classification and clustering. Overall, the standard representation was the best, with the simple and normalized frequency representations also performing well. We will examine the reasons for the variations in performance for different representations in future research.

In future experiments we will utilize larger data sets, to investigate scalability issues, and examine additional graph representation types. The graph representations of web documents presented here can capture term order, proximity, section location, and frequency information. We could look at more elaborate representations that capture additional information. Note that new representations would not necessitate the creation of new algorithms in order to make use of them. Incorporating expert knowledge from the field of natural language understanding (e.g. the first sentence of the paragraph is the most important; words that appear in italics or boldface are important terms) could also improve the quality of the created graphs. We will also consider extending the graph-based techniques presented here to other data mining methods. Another open topic to be explored is finding a procedure for determining the optimal size of each graph (i.e. parameter m) automatically.

Acknowledgments

This work was supported in part by the National Institute for Systems Test and Productivity at the University of South Florida under the U.S. Space and Naval Warfare Systems Command Contract No. N00039–02–C–3244 and by the Fulbright Foundation that has granted Professor Kandel the Fulbright Research Award at Tel-Aviv University, College of Engineering during the academic year 2003–2004.

References

1. Zhong, N., Liu, J., Yao, Y.: In search of the wisdom web. Computer **35** (2002) 27–32
2. Madria, S.K., Bhowmick, S.S., Ng, W.K., Lim, E.P.: Research issues in web data mining. Data Warehousing and Knowledge Discovery (1999) 303–312
3. Dumais, S., Chen, H.: Hierarchical classification of web content. In: Proceedings of SIGIR–00, 23rd ACM International Conference on Research and Development in Information Retrieval. (2000) 256–263
4. Apte, C., Damerau, F., Weiss, S.M.: Automated learning of decision rules for text categorization. ACM Transactions on Information Systems **12** (1994) 233–251
5. Zamir, O., Etzioni, O.: Web document clustering: A feasibility demonstration. In: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. (1998) 46–54

6. Salton, G.: Automatic Text Processing: the Transformation, Analysis, and Retrieval of Information by Computer. Addison-Wesley, Reading (1989)
7. Lopresti, D., Wilfong, G.: Applications of graph probing to web document analysis. In: Proceedings of the 1st International Workshop on Web Document Analysis. (2001) 51–54
8. Liang, J., Doermann, D.: Logical labeling of document images using layout graph matching with adaptive learning. In Lopresti, D., Hu, J., Kashi, R., eds.: Document Analysis Systems V. Volume 2423 of Lecture Notes in Computer Science. Springer-Verlag (2002) 224–235
9. Schenker, A., Last, M., Bunke, H., Kandel, A.: Classification of documents using graph matching. International Journal of Pattern Recognition and Artificial Intelligence **18** (to appear)
10. Schenker, A., Last, M., Bunke, H., Kandel, A.: Classification of web documents using a graph model. In: Proceedings of the 7th International Conference on Document Analysis and Recognition. (2003) 240–244
11. Schenker, A., Last, M., Bunke, H., Kandel, A.: Graph representations for web document clustering. In López, F.J.P., Campilho, A.C., de la Blanca, N.P., Sanfeliu, A., eds.: Proceedings of the 1st Iberian Conference on Pattern Recognition and Image Analysis. Volume 2652 of Lecture Notes in Computer Science., Springer-Verlag (2003) 935–942
12. Schenker, A., Last, M., Bunke, H., Kandel, A.: Clustering of web documents using a graph model. In Antonacopoulos, A., Hu, J., eds.: Web Document Analysis: Challenges and Opportunities. World Scientific Publishing Company (2003) 3–18
13. Tan, C.M., Wang, Y.F., Lee, C.D.: The use of bigrams to enhance text categorization. Information Processing and Management **38** (2002) 529–546
14. Mitchell, T.M.: Machine Learning. McGraw-Hill, Boston (1997)
15. Bunke, H., Shearer, K.: A graph distance metric based on the maximal common subgraph. Pattern Recognition Letters **19** (1998) 225–259
16. Fernández, M.L., Valiente, G.: A graph distance metric combining maximum common subgraph and minimum common supergraph. Pattern Recognition Letters **22** (2001) 753–758
17. Wallis, W.D., Shoubridge, P., Kraetz, M., Ray, D.: Graph distances using graph union. Pattern Recognition Letters **22** (2001) 701–704
18. Dickinson, P., Bunke, H., Dadej, A., Kretzl, M.: On graphs with unique node labels. In: Proceedings of the 4th International Workshop on Graph Based Representations in Pattern Recognition. Volume 2726 of Lecture Notes in Computer Science., Springer-Verlag (2003) 13–23
19. Jiang, X., Muenger, A., Bunke, H.: On median graphs: properties, algorithms, and applications. IEEE Transactions on Pattern Analysis and Machine Intelligence **23** (2001) 1144–1151
20. Zahn, C.T.: Graph-theoretical methods for detecting and describing gestalt structures. IEEE Transactions on Computers **C-20** (1971) 68–86
21. Boley, D., Gini, M., Gross, R., Han, S., Hastings, K., Karypis, G., Kumar, V., Mobasher, B., Moore, J.: Partitioning-based clustering for web document categorization. Decision Support Systems **27** (1999) 329–341
22. Turney, P.: Learning algorithms for keyphrase extraction. Information Retrieval **2** (2000) 303–336
23. Rand, W.M.: Objective criteria for the evaluation of clustering methods. Journal of the American Statistical Association **66** (1971) 846–850
24. Cover, T.M., Thomas, J.A.: Elements of Information Theory. Wiley (1991)