10

Polynomial Time Complexity Graph Distance Computation for Web Content Mining

Adam Schenker, Horst Bunke, Mark Last, and Abraham Kandel

Summary. Utilizing graphs with unique node labels reduces the complexity of the maximum common subgraph problem, which is generally NP-complete, to that of a polynomial time problem. Calculating the maximum common subgraph is useful for creating a graph distance measure, since we observe that graphs become more similar (and thus have less distance) as their maximum common subgraphs become larger and vice versa. With a computationally practical method of determining distances between graphs, we are no longer limited to using simpler vector representations for machine learning applications. We can perform well-known algorithms, such as k-means clustering and k-nearest neighbors classification, directly on data represented by graphs, losing none of the inherent structural information. We demonstrate the benefits of the additional information retained in a graph-based data model for web content mining applications. We introduce several graph representations for capturing web document information and present some examples of our experimental results, which compare favorably with traditional vector methods.

10.1 Introduction

In this chapter we consider applying data mining algorithms, such as the k-nearest neighbors classification algorithm and k-means clustering algorithm, to web document content; this is known as *web content mining*. Content-based classification of web documents is useful because it allows users to more easily navigate and browse collections of documents [1, 2]. 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 that 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 that attempts to organize data items into similar groups, while classification is a supervised learning technique that aims to assign a specific label to each data item. In web content mining, clustering is performed in order to arrange web documents into related groups, such as by the topic of the documents. This has benefits when the classes are not known a priori, such as in web search engines [3], since it allows systems to display results grouped by clusters (topics), 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. Similarly, for clustering methods, cluster centers, or other representatives used for the clusters, are required to change over time to reflect the Internet's constant and rapid influence on language and emerging new concepts. This process occurs, for example, because the topic associated with a cluster representative takes on new meanings (e.g., "Java"), or because new concepts are created that previously had no clusters related to them (e.g., "blogs"). With the arrival of new training examples, we can create new clusters or update existing ones using methods of incremental clustering (see [4]).

Traditionally, data mining methods have represented document content 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 nonnegative integers, which represent the number of times a term appears on a document (i.e., term frequency). Nonnegative 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 [5], 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.

To overcome this limitation, 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. Graphs are important and effective mathematical constructs for modeling relationships and structural information. Graphs (and their more restrictive form, trees) are used in many different problems, including sorting, compression, traffic/flow analysis, resource allocation, etc. [6] Utilizing graphs allows us to keep the inherent structural information of the original web document without having to discard information as we do with a vector model representation. However, until recently, we have not had available to us mathematical techniques for determining distance between graphs as we have had with vectors. Thus data mining techniques such as clustering and classification could not be applied to graphs without creating new mathematical frameworks for dealing with the graphs.

In this chapter we show how the determination of the maximum common subgraph between a pair of graphs can lead to a numerical distance measure between the graphs [7]. A problem with computing the maximum common subgraph is that this is an NP-complete problem in the general case. However, it has been shown that when the nodes in the graphs have unique labels associated with them, the time complexity of finding the maximum common subgraph becomes polynomial [8]. We introduce several methods of representing web document content by graphs with unique node labels. We then proceed to describe how these graphs may be clustered or classified by straightforward extensions to well-known machine learning algorithms such as k-means or k-nearest neighbors when utilizing graph distance measures. We also show some examples of some experimental results obtained from using our graph-based methods.

The remainder of the chapter is organized as follows. In section 10.2 we discuss the complexity issues related to using graphs in machine learning. We show how the maximum common subgraph of a pair of graphs can be used to derive a distance measure between the graphs, and how this computation of the maximum common subgraph can be performed in polynomial time when the graphs have unique node labels. We describe our graph representations of web document content, which make use of the unique node label property, in section 10.3. Versions of the k-means clustering and k-nearest neighbors classification algorithms that utilize graphs and graph distance measures are presented in Section 10.4. Section 10.5 presents some examples of results obtained when using these algorithms to perform web content mining on graph-based data. Conclusions are presented in section 10.6.

10.2 Graph Complexity

10.2.1 Basic Definitions

In this subsection we present some basic definitions related to graph theory. Practically speaking, *graphs* are used to model some system of entities such that the entities are represented by *nodes* in the graph and the relationships present between the entities are reflected in the *edges* connecting the nodes. Formally, we define a graph as follows:

Definition 1. *A* graph *G is a 4-tuple: $G = (V, E, \alpha, \beta)$, where V is a set of* nodes *(also called* vertices*), $E \subseteq V \times V$ is a set of* edges *connecting the nodes, $\alpha : V \rightarrow \Sigma_V$ is a function labeling the nodes, and $\beta : V \times V \rightarrow \Sigma_E$ is a function labeling the edges (Σ_V and Σ_E being the sets of labels that can appear on the nodes and edges, respectively). For brevity, we may abbreviate G as $G = (V, E)$ by omitting the labeling functions.*

A graph that is contained within another graph is called a *subgraph*. Conversely, a graph that contains another graph is also called a *supergraph*. Formally, subgraphs and supergraphs are defined as follows:

Definition 2. *A graph $G_1 = (V_1, E_1, \alpha_1, \beta_1)$ is a* subgraph *of a graph $G_2 = (V_2, E_2, \alpha_2, \beta_2)$, denoted $G_1 \subseteq G_2$, if $V_1 \subseteq V_2$, $E_1 \subseteq E_2 \cap (V_1 \times V_1)$, $\alpha_1(x) = \alpha_2(x)\ \forall x \in V_1$, and $\beta_1((x, y)) = \beta_2((x, y))\ \forall (x, y) \in E_1$. Conversely, graph G_2 is also called a* supergraph *of G_1.*

When we say that two graphs are *isomorphic*, we mean that the graphs contain the same number of nodes and there is a direct 1-to-1 correspondence between the nodes in the two graphs such that the edges between nodes and all labels are preserved.

Definition 3. *Formally, a graph $G_1 = (V_1, E_1, \alpha_1, \beta_1)$ and a graph $G_2 = (V_2, E_2, \alpha_2, \beta_2)$ are said to be* isomorphic*, denoted $G_1 \cong G_2$, if there exists a bijective function $f : V_1 \rightarrow V_2$ such that the following conditions are met:*

1. $\forall x \in V_1 :\ \alpha_1(x) = \alpha_2(f(x))$
2. $\forall (x, y) \in E_1 :\ (f(x), f(y)) \in E_2$ *and* $\beta_1((x, y)) = \beta_2((f(x), f(y)))$
3. $\forall (f(x), f(y)) \in E_2 :\ (x, y) \in E_1$ *and* $\beta_2((f(x), f(y))) = \beta_1((x, y))$

Such a function f is also called a graph isomorphism *between G_1 and G_2.*

There is also the notion of *subgraph isomorphism*, meaning that a graph is isomorphic to a part of (i.e., a subgraph of) another graph:

Definition 4. *Given a graph isomorphism f between graphs G_1 and G_2 as defined above and another graph G_3, if $G_2 \subseteq G_3$, then f is a* subgraph isomorphism *between G_1 and G_3.*

Graph isomorphism was one of the earliest approaches to *graph matching*, the procedure of determining if two graphs are identical to each other. It is not known whether graph isomorphism is an NP-complete problem; however, subgraph isomorphism is NP-complete [9]. Clearly, as the number of nodes in the graphs increase, the number of possible matchings to be checked increases combinatorally. A general procedure for determining subgraph isomorphism is given in Ullman [10]. The naive algorithm for graph isomorphism is to maintain a matrix that indicates which nodes in each graph are compatible; it can require all possible permutations of matchings to determine if there is an isomorphism. The procedure in Ullman [10] improves the complexity by pruning the search space.

Graph isomorphism tells us only that there exists an exact match between two graphs (i.e., that they are identical). It does not give us any indication of similarity between graphs, only whether they are isomorphic or not. Subgraph isomorphism tells us if one graph appears as part of another graph. More relaxed approaches to graph matching, *inexact graph matching* and *graph distance*, have been proposed [11, 12]. Inexact graph matching attempts not to find if two graphs are identical, but rather attempts to find a mapping between the nodes of two graphs that achieves maximum similarity (a "best" matching). Graph distance approaches provide a numerical value that approximates the dissimilarity (distance) between two graphs.

Such new methods have become very important for pattern recognition and machine learning, as they allow us to deal with more robust graph-based data in a manner similar to those used for simpler vector models. Specifically, they permit algorithms to better tolerate noise and imperfect data in the graphs. For example, a missing node or edge caused by noise is not acceptable under graph isomorphism, but may still achieve good results using an inexact matching approach.

10.2.2 Maximum Common Subgraph

A popular method for determining graph distance is the *graph edit distance* approach. *Edit distance* is a method that is used to measure the difference between symbolic data structures such as trees [13] and strings [14]. It is also known as the *Levenshtein distance*, from early work in error-correcting/detecting codes that allowed insertion and deletion of symbols [15]. The concept is straightforward. Various operations are defined on the structures, such as deletion, insertion, and renaming of elements. A cost function is associated with each operation, and the minimum cost needed to transform one structure into the other using the operations is the distance between them. Edit distance has also been applied to graphs, as graph edit distance [16, 17]. The operations in graph edit distance are insertion, deletion, and relabeling of nodes and edges. The distance between two graphs is thus the minimum cost needed to edit one graph into the other by adding, deleting, and renaming nodes and edges.

It has been shown that there is a direct relationship between graph edit distance and the maximum common subgraph between two graphs [7]. Specifically, the two are equivalent under certain restrictions on the cost functions. The *maximum common subgraph* of two graphs is the largest graph the two graphs have in common, and is defined as follows:

Definition 5. *A graph g is a* maximum common subgraph (mcs) *of graphs G_1 and G_2, denoted $mcs(G_1, G_2)$, if: (1) $g \subseteq G_1$ (2) $g \subseteq G_2$ and (3) there is no other subgraph g' ($g' \subseteq G_1$, $g' \subseteq G_2$) such that $|g'| > |g|$.*

In definition 5 above, $|g|$ is usually taken to mean $|V|$, i.e., the number of nodes in the graph; it is used to indicate the "size" of a graph. However, in this chapter we use a different definition of graph size that also takes into account the contribution of the edges in the graphs (see equation (10.5)). Otherwise, with the traditional definition, a sparsely connected graph with many nodes is considered larger than a graph with a few nodes but many edges.

Similar to the maximum common subgraph, there is the complementary idea of minimum common supergraph:

Definition 6. *A graph g is a* minimum common supergraph (MCS) *of graphs G_1 and G_2, denoted $MCS(G_1, G_2)$, if: (1) $G_1 \subseteq g$ (2) $G_2 \subseteq g$ and (3) there is no other supergraph g' ($G_1 \subseteq g'$, $G_2 \subseteq g'$) such that $|g'| < |g|$.*

One method for determining the maximum common subgraph is given in Levi [18]; this approach is to create a compatibility graph for the two given graphs, and then find the largest clique within it. Another approach involves backtracking search [19].

Following the observation that the size of the maximum common subgraph is related to the similarity between two graphs, a graph distance measure based on the maximum common subgraph has been introduced [20]:

$$d_{MCS}(G_1, G_2) = 1 - \frac{|mcs(G_1, G_2)|}{\max(|G_1|, |G_2|)}, \tag{10.1}$$

where $\max(x, y)$ is the usual maximum of two numbers x and y, and $|\ldots|$ indicates the size of a graph (see above). The concept behind this distance measure is that as the size of the maximum common subgraph of a pair of graphs becomes larger, the more similar the two graphs are (i.e., they have more in common). The larger the maximum common subgraph, the smaller $d_{MCS}(G_1, G_2)$ becomes, indicating more similarity and less distance. If the two graphs are in fact identical, their maximum common subgraph is the same as the graphs themselves, and thus the size of all three graphs is equal: $|G_1| = |G_2| = |mcs(G_1, G_2)|$. This leads to the distance, $d_{MCS}(G_1, G_2)$, becoming 0. Conversely, if no maximum common subgraph exists, then $|mcs(G_1, G_2)| = 0$ and $d_{MCS}(G_1, G_2) = 1$. This distance measure has been shown to be a metric [20], and produces a value in $[0, 1]$. This distance measure has four important properties. First, it is restricted to producing a number in the interval $[0, 1]$. Second, the distance is 0 only when the two graphs are identical. Third, the distance between two graphs is symmetric. Fourth, it obeys the triangle inequality, which ensures that the distance measure behaves in an intuitive way. For example, if we have two dissimilar objects (i.e., there is a large distance between them) the triangle inequality implies that a third object that is similar (i.e., has a small distance) to one of those objects must be dissimilar to the other. The advantage of this approach over the graph edit distance method is that it does not require the determination of any cost coefficients or other parameters. However, the metric as it is defined in (10.1) may not be appropriate for all applications; for example, the size of the smaller graph in d_{MCS} makes no contribution to the value of the distance measure, which may be useful to consider in some instances. Thus other distance measures based on the size of the maximum common subgraph or minimum common supergraph have been proposed.

A second distance measure that has been proposed by Wallis et al. [21], based on the idea of graph union, is

$$d_{WGU}(G_1, G_2) = 1 - \frac{|mcs(G_1, G_2)|}{|G_1| + |G_2| - |mcs(G_1, G_2)|}. \tag{10.2}$$

By "graph union" we mean that the denominator represents the size of the union of the two graphs in the set theoretic sense; specifically adding the size of each graph ($|G_1| + |G_2|$)

and then subtracting the size of their intersection ($|mcs(G_1, G_2)|$) leads to the size of the union (the reader may easily verify this using a Venn diagram). This distance measure behaves similarly to d_{MCS}. The motivation for using graph union in the denominator is to allow for changes in the smaller graph to exert some influence over the distance measure, which does not happen with d_{MCS}, as mentioned above. This measure was also demonstrated to be a metric, and creates distance values in $[0, 1]$.

Fernández and Valiente [22] have proposed a distance measure based on both the maximum common subgraph and the minimum common supergraph:

$$d_{MMCS}(G_1, G_2) = |MCS(G_1, G_2)| - |mcs(G_1, G_2)|, \tag{10.3}$$

where $MCS(G_1, G_2)$ is the minimum common supergraph of graphs G_1 and G_2. The concept that drives this distance measure is that the maximum common subgraph provides a "lower bound" on the similarity of two graphs, while the minimum common supergraph is an "upper bound." If two graphs are identical, then both their maximum common subgraph and minimum common supergraph are the same as the original graphs and $|G_1| = |G_2| = |MCS(G_1, G_2)| = |mcs(G_1, G_2)|$, which leads to $d_{MMCS}(G_1, G_2) = 0$. As the graphs become more dissimilar, the size of the maximum common subgraph decreases, while the size of the minimum common supergraph increases. This in turn leads to increasing values of $d_{MMCS}(G_1, G_2)$. For two graphs with no maximum common subgraph, the distance will become $|MCS(G_1, G_2)| = |G_1| + |G_2|$. d_{MMCS} has also been shown to be a metric, but it does not produce values normalized to the interval $[0, 1]$, unlike d_{MCS} or d_{WGU}. Note that if it holds that $|MCS(G_1, G_2)| = |G_1| + |G_2| - |mcs(G_1, G_2)|\ \forall G_1, G_2$, we can compute $d_{MMCS}(G_1, G_2)$ as $|G_1| + |G_2| - 2|mcs(G_1, G_2)|$. This is much less computationally intensive than computing the minimum common supergraph.

10.2.3 Graphs with Unique Node Labels

As mentioned above, the subgraph isomorphism problem is NP-complete. As finding the maximum common subgraph requires determining subgraph isomorphism, it is also an NP-complete problem [8]. However, recently it has become known that for certain classes of graphs the maximum common subgraph, and thus the graph distance, can be determined in polynomial time. Specifically, graphs whose node labels are unique can have their maximum common subgraphs computed in $O(n^2)$ time, where n is the number of nodes in the graph [8]. Formally, a graph has unique node labels according to the following definition:

Definition 7. *A graph $G = (V, E, \alpha, \beta)$ has unique node labels if for $\forall v_1, v_2 \in V, \alpha(v_1) \neq \alpha(v_2)$ unless $v_1 = v_2$.*

Note that the elements of set V, i.e., the nodes, are always uniquely defined. However, in the general case, i.e., in a graph without any restrictions, different nodes may carry the same label. For example, the field of chemistry uses graphs to represent molecules; nodes correspond to atoms and edges to bonds formed between atoms. A water molecule (H_2O) would have a graph with three nodes: one for oxygen (labeled "O") and two for hydrogen (both labeled "H").

The above result follows from the fact that determining the nodes of the maximum common subgraphs between two graphs, $G_1 = (V_1, E_1, \alpha_1, \beta_1)$ and $G_2 = (V_2, E_2, \alpha_2, \beta_2)$, each with unique node labels, reduces to the problem of finding the intersection of two sets, namely $GL_1 = \{\alpha_1(v) | \forall v \in V_1\}$ and $GL_2 = \{\alpha_2(v) | \forall v \in V_2\}$. Similarly the minimum common supergraph can be computed by taking the union of these two sets. The actual procedure can be performed as follows:

1. Determine the set of labels that each of the two original graphs have in common, GL_{mcs}, by computing the intersection of sets GL_1 and GL_2 (see above), i.e., $GL_{mcs} = GL_1 \cap GL_2$.
2. For each label $L \in GL_{mcs}$, create a new node N in V_{mcs} labeled such that $\alpha_{mcs}(N) = L$.
3. Determine the edges of the maximum common subgraph E_{mcs} by examining all pairs of nodes in V_{mcs} and add edges to E_{mcs} that connect pairs of nodes in both of the original graphs and that have matching edge labels; the added edge in the maximum common subgraph will have the same label.

We see that the complexity of this method is $O(|V_1| \cdot |V_2|)$ for step 1, since we need only compare each node label from one graph to each node label of the other and determine whether there is a match or not. Thus the maximum number of comparisons is $|V_1| \cdot |V_2|$, and since each node has a unique label we need to consider each combination only once. For step 2, the complexity is $O(|V_{mcs}|)$. The complexity is $O(|V_{mcs}|^2)$ for step 3, since we have $|V_{mcs}|$ nodes and we look at all combinations of pairs of nodes to determine if an edge should be added between them or not:

$$\binom{|V_{mcs}|}{2} = \frac{|V_{mcs}|!}{(|V_{mcs}|-2)! \cdot 2!} = \frac{|V_{mcs}| \cdot (|V_{mcs}|-1)}{2} < |V_{mcs}|^2. \tag{10.4}$$

Thus the overall complexity is $O(|V_1| \cdot |V_2| + |V_{mcs}| + |V_{mcs}|^2) \leq O(|V|^2 + |V_{mcs}|^2) = O(|V|^2)$ if we substitute $V = \max(|V_1|, |V_2|)$. Note that the case of the minimum common supergraph is the same, except we change the intersection in step 1 to a union.

Given this result, we introduce graph representations of data that utilize unique node labels to take advantage of the improved time complexity for determining the maximum common subgraph, which, in turn, allows for graph distance to be calculated in polynomial time. Our application domain is web content mining, and our graph representations of web documents are given in the next section.

10.3 Graph Representations for Web Document Content

In this section we describe methods for representing web document content using graphs with unique node labels instead of the vector representations that are traditionally used. 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. Also, 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 hyperlinks 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 in 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 [5]. For the final step in our graph creation process, we perform a simple stemming method and *conflate* words (an information-retrieval term for merging multiple word forms so they are represented by a single entity) to the most frequently occurring form by relabeling nodes and updating edges as needed.

An example of this type of graph representation is given in Figure 10.1. 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." A brief point of clarification is necessary concerning the *link* section. We do not examine the URLs of the hyperlinks to create the graphs; instead we are examining the text that labels the hyperlink itself and appears on the web document for the user to click. Note that there is no restriction on the form of the graph, and that cycles are allowed. If pairs of terms appear adjacent in more than one section, we add an edge for each occurrence, labeled appropriately.

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 [23]. The methods presented here, with the exception of the frequency representations described below, do not require or use the computation of term probability relationships.

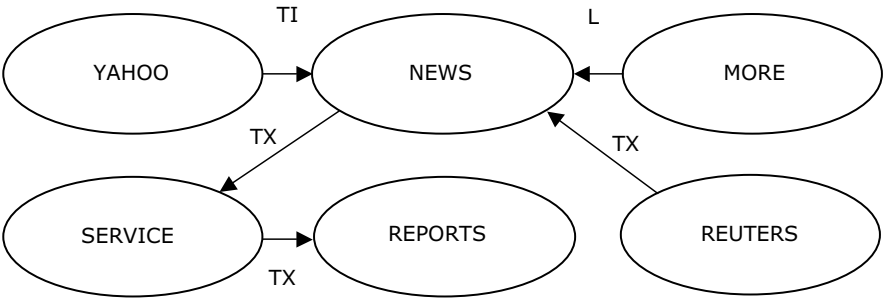

Fig. 10.1. Example of a *standard* graph representation of a document.

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 Figure 10.2.

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

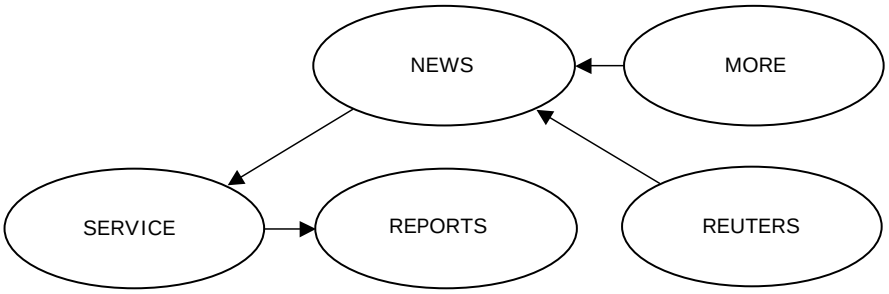

Fig. 10.2. Example of a *simple* graph representation of a document.

1, an edge from term AAA to term CCC labeled 2, and so on. The complete graph for this example is shown in Figure 10.3.

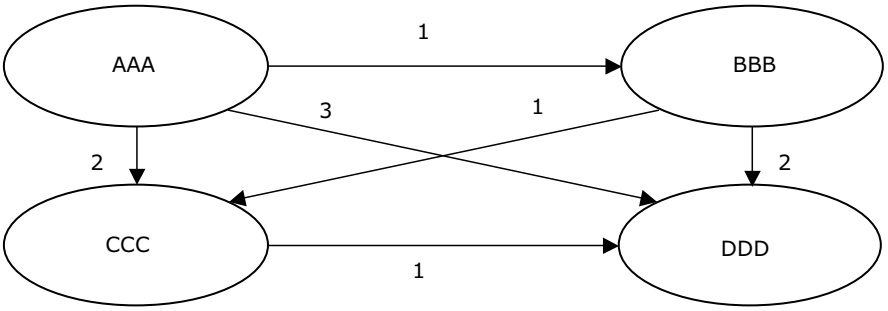

Fig. 10.3. Example of an n-*distance* graph representation of a document.

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).

Previously we stated that the "size" of a graph, $|G|$, is usually defined as the number of nodes in the graph. However, for our particular representations of web documents it is detrimental to ignore the contribution of the edges, which indicate the number of phrases (term adjacencies) identified in the document content. Further, it is possible to have more than one edge between two nodes for certain representations. Thus we will use the following definition of graph size for all representations except the frequency representations (the size of a graph under the frequency representations will be described below). Formally, the size of a graph $G = (V, E, \alpha, \beta)$, denoted $|G|$, is defined as

$$|G| = |V| + |E|. \tag{10.5}$$

Thus we will take the size to be the sum of the number of vertices and edges in the graph for the standard, simple, n-distance, and n-simple distance representations.

However, under the raw frequency and normalized frequency representations the graph size is defined as the total of the node frequencies added to the total of the edge frequencies. We need this modification to reflect the frequency information in the graph size. As an example, consider two raw frequency graphs each with a node "A"; however, term "A" appears two times in one document and 300 in the other. This difference in frequency information is not captured under equation (10.5). Further, when we compute the maximum common subgraph for these representations, we take the minimum frequency element (either node or edge) as the value for the maximum common subgraph. To continue the above example, node "A" in the maximum common subgraph would have a frequency of 2, which is $\min(2, 300)$.

10.4 Graph-Based Web Mining Algorithms

Now that we have graph representations of web documents with unique node labels, we can compute the distance between two web documents in polynomial time. This allows us to retain the graph representations for use in various machine learning methods. The main benefit of this approach is that the additional structural information captured in the graphs is maintained, unlike other methods where we need to discard the structural information to arrive at a vector representation.

In this section we describe two classical machine learning algorithms, k-means and k-nearest neighbors, and show how they can be extended in a straightforward manner to utilize graphs and graph distance.

10.4.1 k-Means Clustering with Graphs

The k-means clustering algorithm is a simple and straightforward method for clustering data [24]. The basic algorithm is given in Figure 10.4. Traditionally, each item to be clustered is represented as a vector in the Euclidean space $\Re^m$, and a vector distance measure such as Jaccard is used [5]:

$$dist_{JAC}(\mathbf{x}, \mathbf{y}) = 1 - \frac{\sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n} x_i^2 + \sum_{i=1}^{n} y_i^2 - \sum_{i=1}^{n} x_i y_i}, \tag{10.6}$$

where x_i and y_i are the ith components of vectors $\mathbf{x}$ and $\mathbf{y}$, respectively.

For our graph-based approach, instead of vectors we will represent web document content using graphs, as discussed in section 10.3. To compute distances, we simply use one of the

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. 10.4. The basic k-means clustering algorithm.

methods described in section 10.2.2. 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* [25]. Formally, the median of a set of graphs S is a graph $g \in S$ ($S = \{G_1, G_2, \ldots, G_n\}$) such that g has the lowest average distance to all graphs in S:

$$g = \arg\min_{\forall s \in S} \left(\frac{1}{|S|} \sum_{i=1}^{|S|} dist(s, G_i) \right). \tag{10.7}$$

The median of a set of graphs is the graph from the set that has the minimum average distance to all the other graphs in the set. Here the distance is computed with the graph-theoretic distance measures mentioned in section 10.2.2. The procedure is fairly straightforward, though the equation may seem complex at first. We start by selecting some specific graph, let us call it s, and then compute the distances between s and all other graphs in a pair-wise fashion. These distances are summed and then divided by the total number of graphs to calculate an average distance between s and all the other graphs. This number is saved and associated with graph s; we repeat the above process with all the graphs, taking each one in turn to be "s." The median graph is then selected by finding the graph that has the minimum distance.

We wish to clarify here 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 the weights on the edges connecting the nodes indicate the distance between the objects the nodes represent. The goal is to partition this graph, breaking it up into several connected components that 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 [26]. This is very different from the technique we described in this section, since it is the data (in this case, the web documents) themselves that are represented by graphs, not the overall clustering problem.

10.4.2 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 [24] begins with a set of training examples; in the

traditional k-NN approach these are numerical feature vectors. Each of these training examples is associated with a label that indicates to what class the example belongs. Given a new, previously unseen 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 examples closest (i.e., with least distance) to the input instance. Here k is a user-provided parameter and distance is computed with a vector distance measure, such as equation (10.6).

Once we have found the k nearest training examples using some distance measure, we estimate the class by the majority among the k training examples. 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 among the nearest neighbors, we can either choose one class randomly or break the tie with some other method, such as selecting the tied class that has the minimum distance neighbor. For the experiments in this chapter we will use the latter method, which in our experiments has shown a slight improvement over random tie breaking.

To extend the k-NN method to work with graph representations of web documents instead of vector representations, we need only utilize one of the graph distance measures presented in section 10.2.2 in place of the traditional vector distance measures. Then we may use graphs in place of vectors with no further changes to the algorithm.

10.5 Experimental Results

10.5.1 Data Sets

To evaluate the performance of the graph-based k-means and k-NN 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* [27]. The data sets are available under these names at `ftp://ftp.cs.umn.edu/dept/users/boley/PDDPdata/`. 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 provide only a preprocessed 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 task in mind other 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 subcategories of four major category areas: *manufacturing*, *labor*, *business and finance*, and *electronic communication and networking*. Because there are multiple subcategory 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) that 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 elsewhere [28, 29, 30].

For the vector representation experiments, which are presented as a baseline for comparison purposes, there were already several precreated 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 schemes (i.e., inverse document frequency, see [5]) 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.

10.5.2 Experimental Details

For our experiments we use a maximum graph size of 30 nodes per graph, which corresponds to setting $m = 30$ (see section 10.3). This parameter value was selected based on previous experimental results, and has been shown to work adequately for both data sets (further results with other graph sizes are omitted for brevity). We select a single value for m to be used by all graphs for experimental consistency. However, the value of m could be different for each graph, which would allow for more flexibility than vector-space models, since they require a fixed number of dimensions for every document. The graph model can allow for a different representation size for each document, which would require some method of selecting a "good" value of m for each document. This is part of the more general *keyphrase extraction* problem [31], which does not have a trivial solution; describing methods for dealing with it is beyond the scope of this chapter. Note that it is also possible to reduce the size of the graphs by examination of graph-theoretic features, such as focusing on large connected components, nodes with high edge degrees, or components with certain topologies.

The d_{MCS} distance measure (10.1) was used to compute graph distance for both algorithms. For the "distance" related graph representations, n-distance and n-simple distance, we used $n = 5$ (i.e., 5-distance and 5-simple distance). The vector representation results reported for comparison reflect using a distance measure based on Jaccard similarity, equation (10.6). We used Jaccard distance because this was consistently the best performing vector distance measure in our experimental results. Euclidean distance is generally not used for information retrieval tasks and performs poorly because it lacks a length-invariance property. With Euclidean distance, large variations in overall document size cause large distances between their representative vectors, even though the two documents may be about identical topics; the document content is ideally described by vector direction, not length. (For further discussion of this topic, see [5, 32].)

Clustering performance is measured using two performance indices that indicate the similarity of obtained clusters to the "ground truth" clusters. The first performance index is the *Rand index* [33], 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 for measuring clustering performance is *mutual information* [34], 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. Classification accuracy was assessed by the leave-one-out method, where we use all but one of the instances in the data set as training examples and attempt to classify the remaining input instance. The procedure

is carried out using each instance in the data set as the input instance once, and the overall accuracy is reported.

10.5.3 Examples of Results

The performance of clustering the F and J data sets, as measured by the Rand index when compared with ground truth, after applying k-means clustering, is given in Figure 10.5. Similarly, the performance as measured by mutual information is given in Figure 10.6. The figures compare the performance obtained when using the different graph representations presented in section 10.3. 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 the Jaccard similarity [5], which is the best performing vector distance measure we have worked with. The white bars correspond to the F-series data set, whereas the black bars are the J-series. On our system, a 2.6 GHz Pentium 4 with 1 gigabyte of memory, the average time to create clusters for the F-series using the graph-based method and the standard representation was 22.7 seconds, whereas it took 59.5 seconds on average for the J-series.

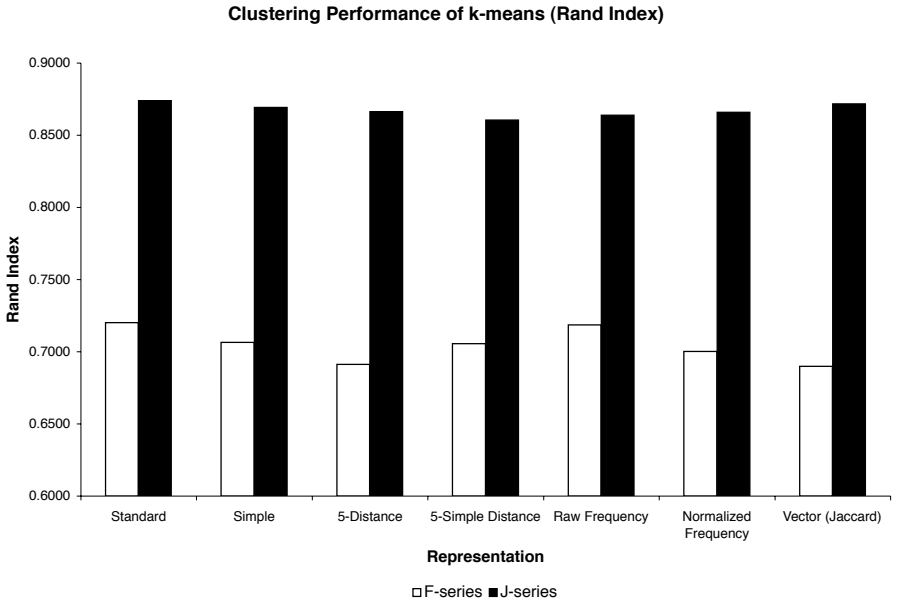

Fig. 10.5. Performance of k-means clustering on F and J data sets as measured by the Rand index.

The results for the k-nearest neighbors classification experiments are given in Figures 10.7 and 10.8 for the F and J data sets, respectively. Similar to the clustering results, the various representations are compared. The different bars in each group correspond to different values of k (the number of nearest neighbors). The white bars correspond to $k = 1$, the gray bars are for $k = 3$, the striped bars indicate $k = 5$, and the black bars are $k = 10$. The graph-based k-NN 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.

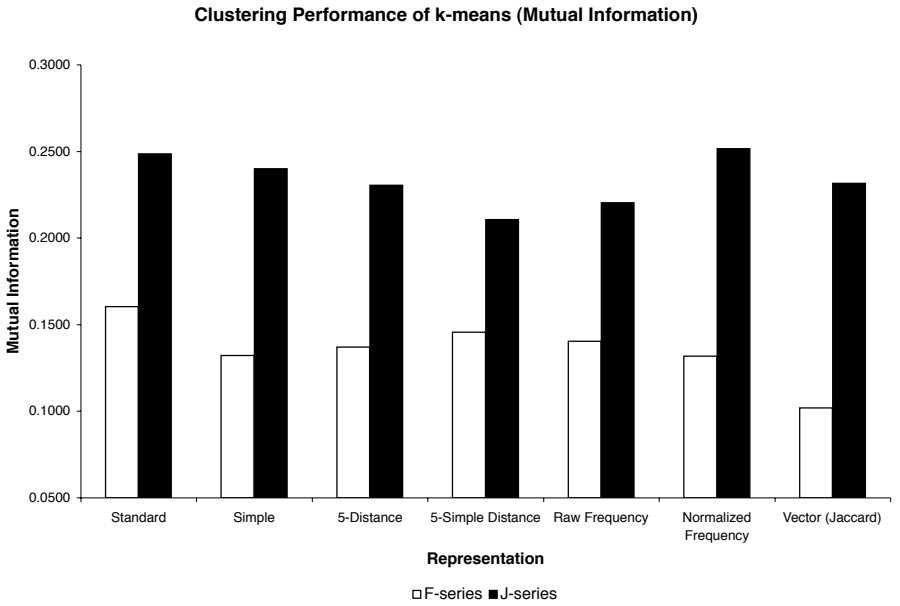

Fig. 10.6. Performance of k-means clustering on F and J data sets as measured by mutual information.

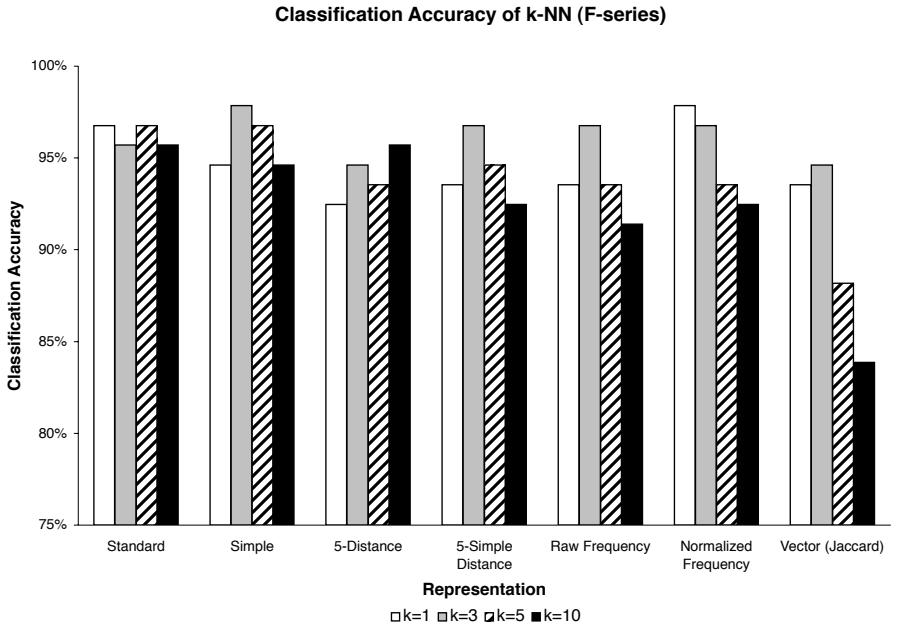

Fig. 10.7. Performance of k-nearest neighbors classification for the F-series data set with accuracy measured using leave-one-out.

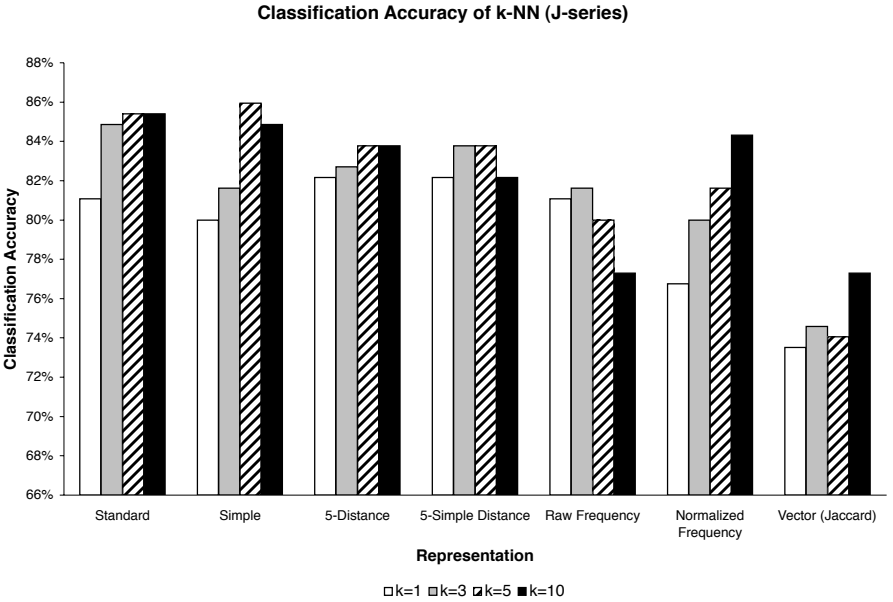

Fig. 10.8. Performance of k-nearest neighbors classification for the J-series data set with accuracy measured using leave-one-out.

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 eight out of 12 experiments. The 5-simple distance representation was an improvement in nine out of 12 cases. Raw frequency was better in eight of 12 cases, while normalized frequency was an improvement in 11 of 12 cases.

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.

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%.

Additional experimental results comparing the performance of different graph distance measures (section 10.2.2) can be found in [28, 35]. Evaluations of other clustering algorithms when utilizing graphs are reported in [30, 36]. Creation of classifier ensembles using random node selection for graphs is described in [37].

10.6 Conclusion

We have demonstrated how using graphs with unique node labels reduces the complexity of the maximum common subgraph problem to polynomial time, and how utilizing the maximum common subgraph allows us to calculate a graph distance measure. Such graph distance measures are useful for allowing clustering and classification algorithms to work with graph representations of data, which contain additional structural information when compared to their vector counterparts. We introduced several methods of representing web document content using graphs with unique node labels. We also presented graph-based versions of the k-means and k-nearest neighbors algorithms, and showed some examples of experimental results when applying these methods to web document collections. The results show our graph-based approach can outperform traditional vector models for both clustering and classification.

Acknowledgments

This work was supported in part by the National Institute for Systems Test and Productivity at the University of South Florida under U.S. Space and Naval Warfare Systems Command Contract No. N00039–02–C–3244.

References

[1] C. Apte, F. Damerau, S.M. Weiss. Automated learning of decision rules for text categorization. *ACM Transactions on Information Systems*, 12, 233–251, 1994.

[2] S. Dumais, H. Chen. Hierarchical classification of web content. In *Proceedings of SIGIR-00, 23rd ACM International Conference on Research and Development in Information Retrieval*, pages 256–263, 2000.

[3] O. Zamir, O. Etzioni. Web document clustering: a feasibility demonstration. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 46–54, 1998.

[4] A.K. Jain, M.N. Murty, P.J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3), 264–323, 1999.

[5] G. Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Reading, MA: Addison-Wesley, 1989.

[6] T.H. Cormen, C.E. Leiserson, R.L. Rivest. *Introduction to Algorithms*. Cambridge, MA: MIT Press, 1997.

[7] H. Bunke. On a relation between graph edit distance and maximum common subgraph. *Pattern Recognition Letters*, 18, 689–694, 1997.

[8] P.J. Dickinson, H. Bunke, A. Dadej, M. Kraetzl. Matching graphs with unique node labels. *Pattern Analysis and Applications*, 7(3), 243–254, 2004.

[9] M.R. Garey, D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York: W. H. Freeman, 1979.

[10] J.R. Ullman. An algorithm for subgraph isomorphism. *Journal of the Association for Computing Machinery*, 23, 31–42, 1976.

[11] J.T.L. Wang, K. Zhang, G.-W. Chirn. Algorithms for approximate graph matching. *Information Sciences*, 82, 45–74, 1995.

[12] B.T. Messmer, H. Bunke. A new algorithm for error-tolerant subgraph isomorphism detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(5), 493–504, 1998.

[13] K.-C. Tai. The tree-to-tree correction problem. *Journal of the Association for Computing Machinery*, 26(3), 422–433, 1979.
[14] R.A. Wagner, M.J. Fischer. The string-to-string correction problem. *Journal of the Association for Computing Machinery*, 21, 168–173, 1974.
[15] V. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics-Doklady*, 10, 707–710, 1966.
[16] A. Sanfeliu, K.S. Fu. A distance measure between attributed relational graphs for pattern recognition. *IEEE Transactions on Systems, Man, and Cybernetics*, 13, 353–363, 1983.
[17] H. Bunke, G. Allermann. Inexact graph matching for structural pattern recognition. *Pattern Recognition Letters*, 1(4), 245–253, 1983.
[18] G. Levi. A note on the derivation of maximal common subgraphs of two directed or undirected graphs. *Calcolo*, 9, 341–354, 1972.
[19] J.J. McGregor. Backtrack search algorithms and the maximal common subgraph problem. *Software Practice and Experience*, 12, 23–34, 1982.
[20] H. Bunke, K. Shearer. A graph distance metric based on the maximal common subgraph. *Pattern Recognition Letters*, 19, 255–259, 1998.
[21] W.D. Wallis, P. Shoubridge, M. Kraetz, D. Ray. Graph distances using graph union. *Pattern Recognition Letters*, 22, 701–704, 2001.
[22] M.-L. Fernández, G. Valiente. A graph distance metric combining maximum common subgraph and minimum common supergraph. *Pattern Recognition Letters*, 22, 753–758, 2001.
[23] C.-M. Tan, Y.-F. Wang, C.-D. Lee. The use of bigrams to enhance text categorization. *Information Processing and Management*, 38, 529–546, 2002.
[24] T.M. Mitchell. *Machine Learning*. New York: McGraw-Hill, 1997.
[25] X. Jiang, A. Muenger, H. Bunke. On median graphs: properties, algorithms, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(10), 1144–1151, 2001.
[26] C.T. Zahn. Graph-theoretical methods for detecting and describing gestalt structures. *IEEE Transactions on Computers*, C-20, 68–86, 1971.
[27] D. Boley, M. Gini, R. Gross, et al. Partitioning-based clustering for web document categorization. *Decision Support Systems*, 27, 329–341, 1999.
[28] A. Schenker, M. Last, H. Bunke, A. Kandel. Classification of documents using graph matching. *International Journal of Pattern Recognition and Artificial Intelligence*, 18(3), 475–496, 2004.
[29] A. Schenker, M. Last, H. Bunke, A. Kandel. Classification of web documents using a graph model. In *Proceedings of the 7th International Conference on Document Analysis and Recognition*, pages 240–244, 2003.
[30] A. Schenker, M. Last, H. Bunke, A. Kandel. A comparison of two novel algorithms for clustering web documents. In *Proceedings of the 2nd International Workshop on Web Document Analysis*, pages 71–74, 2003.
[31] P.D. Turney. Learning algorithms for keyphrase extraction. *Information Retrieval*, 2(4), 303–336, 2000.
[32] A. Strehl, J. Ghosh, R. Mooney. Impact of similarity measures on web-page clustering. In *AAAI-2000: Workshop of Artificial Intelligence for Web Search*, pages 58–64, 2000.
[33] W.M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66, 846–850, 1971.
[34] T.M. Cover, J.A. Thomas. *Elements of Information Theory*. New York: Wiley, 1991.
[35] A. Schenker, M. Last, H. Bunke, A. Kandel. Comparison of distance measures for graph-based clustering of documents. In E. Hancock, M. Vento, eds. *Proceedings of the 4th*

IAPR-TC15 International Workshop on Graph-based Representations in Pattern Recognition, volume 2726 of *Lecture Notes in Computer Science*, pages 202–213. New York: Springer-Verlag, 2003.

[36] A. Schenker, M. Last, H. Bunke, A. Kandel. Comparison of algorithms for web document clustering using graph representations of data. In A. Fred, T. Caelli, R.P.W. Duin, A. Campilho, D. de Ridder, eds. *Proceedings of the Joint IAPR Workshop on Syntactical and Structural Pattern Recognition*, volume 3138 of *Lecture Notes in Computer Science*, pages 190–197. New York: Springer-Verlag, 2004.

[37] A. Schenker, H. Bunke, M. Last, A. Kandel. Building graph-based classifier ensembles by random node selection. In F. Roli, J. Kittler, T. Windeatt, eds. *Proceedings of the 5th International Workshop on Multiple Classifier Systems*, volume 3077 of *Lecture Notes in Computer Science*, pages 214–222. New York: Springer-Verlag, 2004.