

Chapter 1

Introduction to Web Mining

With the recent explosive growth of the amount of content on the Internet, it has become increasingly difficult for users to find and utilize information and for content providers to classify and catalog documents. Traditional web search engines often return hundreds or thousands of results for a search, which is time consuming for users to browse. On-line libraries, search engines, and other large document repositories (*e.g.* customer support databases, product specification databases, press release archives, news story archives, *etc.*) are growing so rapidly that it is difficult and costly to categorize every document manually. In order to deal with these problems, researchers look toward automated methods of working with web documents so that they can be more easily browsed, organized, and cataloged with minimal human intervention.

In contrast to the highly structured tabular data upon which most machine learning methods are expected to operate, web and text documents are semi-structured. Web documents have well-defined structures such as letters, words, sentences, paragraphs, sections, punctuation marks, HTML tags, and so forth. We know that words make up sentences, sentences make up paragraphs, and so on, but many of the rules governing the order in which the various elements are allowed to appear are vague or ill-defined and can vary dramatically between documents. It is estimated that as much as 85% of all digital business information, most of it web-related, is stored in non-structured formats (*i.e.* non-tabular formats, such as those that are used in databases and spreadsheets) [Pet]. Developing improved methods of performing machine learning techniques on this vast amount of non-tabular, semi-structured web data is therefore highly desirable.

Clustering and classification have been useful and active areas of machine learning research that promise to help us cope with the problem of

information overload on the Internet. With clustering the goal is to separate a given group of data items (the data set) into groups called clusters such that items in the same cluster are similar to each other and dissimilar to the items in other clusters. In clustering methods no labeled examples are provided in advance for training (this is called *unsupervised learning*). Under classification we attempt to assign a data item to a predefined category based on a model that is created from pre-classified training data (*supervised learning*). In more general terms, both clustering and classification come under the area of knowledge discovery in databases or data mining. Applying data mining techniques to web page content is referred to as web content mining which is a new sub-area of web mining, partially built upon the established field of information retrieval.

When representing text and web document content for clustering and classification, a vector-space model is typically used. In this model, each possible term that can appear in a document becomes a feature dimension [Sal89]. The value assigned to each dimension of a document may indicate the number of times the corresponding term appears on it or it may be a weight that takes into account other frequency information, such as the number of documents upon which the terms appear. This model is simple and allows the use of traditional machine learning methods that deal with numerical feature vectors in a Euclidean feature space. However, it discards information such as the order in which the terms appear, where in the document the terms appear, how close the terms are to each other, and so forth. By keeping this kind of structural information we could possibly improve the performance of various machine learning algorithms. The problem is that traditional data mining methods are often restricted to working on purely numeric feature vectors due to the need to compute distances between data items or to calculate some representative of a cluster of items (*i.e.* a centroid or center of a cluster), both of which are easily accomplished in a Euclidean space. Thus either the original data needs to be converted to a vector of numeric values by discarding possibly useful structural information (which is what we are doing when using the vector model to represent documents) or we need to develop new, customized methodologies for the specific representation.

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.* [CLR97] In addition to problems where the graph itself is processed by some algorithm

(*e.g.* sorting by the depth first method or finding the minimum spanning tree) it would be extremely desirable in many applications, including those related to machine learning, to model data as graphs since these graphs can retain more information than sets or vectors of simple atomic features. Thus much research has been performed in the area of graph similarity in order to exploit the additional information allowed by graph representations by introducing mathematical frameworks for dealing with graphs. Some application domains where graph similarity techniques have been applied include face [WFKvdM97] and fingerprint [WJH01] recognition as well as anomaly detection in communication networks [DBD⁺01]. In the literature, the work comes under several different topic names including graph distance, (exact) graph matching, inexact graph matching, error-tolerant graph matching, or error-correcting graph matching. In exact graph matching we are attempting to determine if two graphs are identical. Inexact graph matching implies we are attempting not to find a perfect matching, but rather a “best” or “closest” matching. Error-tolerant and error-correcting are special cases of inexact matching where the imperfections (*e.g.* missing nodes) in one of the graphs, called the data graph, are assumed to be the result of some errors (*e.g.* from transmission or recognition). We attempt to match the data graph to the most similar model graph in our database. Graph distance is a numeric measure of dissimilarity between graphs, with larger distances implying more dissimilarity. By graph similarity, we mean we are interested in some measurement that tells us how similar graphs are regardless if there is an exact matching between them.

1.1 Overview of Web Mining Methodologies

Web mining [ZLY02] is the application of machine learning (data mining) techniques to web-based data for the purpose of learning or extracting knowledge. Web mining encompasses a wide variety techniques, including soft computing [PTM00]. Web mining methodologies can generally be classified into one of three distinct categories: web usage mining, web structure mining, and web content mining [MBNL99]. For a survey of techniques used in these areas, see Ref. [KB00]. 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. For example, the tool presented by Masseglia *et al.* [MPC99] creates association rules from web access logs, which store the identity of pages accessed by users along with

other information such as when the pages were accessed and by whom; these logs are the focus of the data mining effort, rather than the actual web pages themselves. Rules created by their method could include, for example, “70% of the users that visited page A also visited page B.” Similarly, the method of Nasraoui *et al.* [NFJK99] also examines web access logs. The method employed in that paper is to perform a hierarchical clustering in order to determine the usage patterns of different groups of users. Beeferman and Berger [BB00] described a process they developed which determines topics related to a user query using click-through logs and agglomerative clustering of bipartite graphs. The transaction-based method developed in Ref. [Mer99] creates links between pages that are frequently accessed together during the same session. Web usage mining is useful for providing personalized web services, an area of web mining research that has lately become active. It promises to help tailor web services, such as web search engines, to the preferences of each individual user. For a recent review of web personalization methods, see Ref. [EV03].

In the second category of web mining methodologies, *web structure mining*, we examine only the relationships between web documents by utilizing the information conveyed by each document’s hyperlinks. Like the web usage mining methods described above, the other content of the web pages is often ignored. In Ref. [KRRT99] Kumar *et al.* examined utilizing a graph representation of web page structure. Here nodes in the graphs are web pages and edges indicate hyperlinks between pages. By examining these “web graphs” it is possible to find documents or areas of interest through the use of certain graph-theoretical measures or procedures. Structures such as web rings, portals, or affiliated sites can be identified by matching the characteristics of these structures (*e.g.* we can identify portal pages because they have an unusually high out-degree). Graph models are also used in other web structure mining approaches. For example, in Ref. [CvdBD99] the authors’ method examines linked URLs and performs classification using a Bayesian method. The graph is processed to determine groups of pages that relate to a common topic.

In this book we are concerned only 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 knowledge discovery procedure to learn about the pages themselves and their relationships. Most typically this is done to organize a group of documents into related categories. This is especially beneficial for web search engines, since it allows users to more quickly find the informa-

tion they are looking for in comparison to the usual “endless” ranked list. There are several examples of web or text mining approaches [AHKV97] that are content-oriented and attempt to cluster documents for browsing. The Athena system of Agrawal *et al.* [AJS00] creates groupings of e-mail messages. The goal is to create folders (classes) for different topics and route e-mail messages automatically to the appropriate folders. Athena uses a clustering algorithm called C-Evolve to create topics (folders), while the classification of documents to each cluster is supervised and requires manual interaction with the user. The classification method is based on Naïve Bayes. Some notable papers that deal with clustering for web search include Ref. [BGG⁺99a], which describes 2 partitional methods, and Ref. [CH97], which is a hierarchical clustering approach. Nahm and Mooney [NM00] described a methodology where information extraction and data mining can be combined to improve upon each other; information extraction provides the data mining process with access to textual documents (text mining) and in turn data mining provides learned rules to the information extraction portion to improve its performance. An important paper that is strongly related to the current work is that of Strehl *et al.* [SGM00], which examined clustering performance of different clustering algorithms when using various similarity measures on web document collections. Clustering methods examined in the paper included k -means, graph partitioning, and self-organizing maps (SOMs). Vector-based representations were used in the experiments along with distance measures based on Euclidean distance, cosine similarity, and Jaccard similarity. One of the data sets used in this paper is publicly available and we will use it in our experiments.

1.2 Traditional Information Retrieval Techniques

Traditional information retrieval methods represent plain-text documents using a series of numeric values for 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 a non-negative integers, which represents 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 (*tf-idf*) model [Sal89], 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 machine learning techniques to documents, as there is a strong mathematical foundation for performing distance measure and centroid calculations using vectors.

1.2.1 Vector-based distance measures

Here we briefly review some of the most popular vector-related distance measures [Sal89][SGM00], which will also be used in the experiments we perform. First, we have the well-known Euclidean distance:

$$\text{dist}_{EUCL}(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}, \quad (1.1)$$

where x_i and y_i are the i th components of vectors $x = [x_1, x_2, \dots, x_n]$ and $y = [y_1, y_2, \dots, y_n]$, respectively. Euclidean distance measures the direct distance between two points in the space \mathbb{R}^n .

For applications in text and document clustering and classification, the cosine similarity measure [Sal89] is often used. We can convert this to a distance measure by the following:

$$\text{dist}_{COS}(x, y) = 1 - \frac{x \bullet y}{\|x\| \cdot \|y\|}. \quad (1.2)$$

Here \bullet indicates the dot product operation and $\|\dots\|$ indicates the magnitude (length) of a vector. If we take each document to be a point in \mathbb{R}^n formed by the values assigned to it under the vector model, each value corresponding to a dimension, then the direction of the vector formed from the origin to the point representing the document indicates the content of the document. Under the Euclidean distance, documents with large differences in size have a large distance between them, regardless of whether or not the content is similar, because the length of the vectors differs. The cosine distance is *length invariant*, meaning only the direction of the vectors is compared; the magnitude of the vectors is ignored.

Another popular distance measure for determining document similarity is the extended Jaccard similarity [Sal89], which is converted to a distance

measure as follows:

$$\text{dist}_{JAC}(x, y) = 1 - \frac{\sum_{i=1}^n x_i y_i}{\sum_{i=1}^n x_i^2 + \sum_{i=1}^n y_i^2 - \sum_{i=1}^n x_i y_i}. \quad (1.3)$$

Jaccard distance has properties of both the Euclidean and cosine distance measures. At high distance values, Jaccard behaves more like the cosine measure; at lower values, it behaves more like Euclidean. See Ref. [SGM00] for a comparison of the behavior of these distance measures.

1.2.2 *Special considerations for web documents*

Document clustering using the vector model has long been studied in the information retrieval field as a means of improving retrieval efficiency and corpus visualization [BYRN99][Sal89]. For example, in Ref. [CCA89] the application described by the authors uses the popular agglomerative hierarchical clustering method to create a cluster hierarchy for the entire document collection for the purpose of visualizing and browsing the document corpus. Another similar approach is that of Kohonen *et al.* [KKL⁺00], which uses self-organizing maps, a type of unsupervised neural network, to group documents in a collection. Like the application described in Ref. [CCA89], the Kohonen *et al.* system provides an environment where the results and groupings can be browsed. Early work on clustering documents retrieved by queries for the purpose of improving user navigation through the results is reported in Ref. [HP96].

A topic related to document clustering is that of document classification. The goal of such a task is to assign a label (or class) to a previously unseen document. This is different from document clustering, where the objective is to create groupings of a document collection. Document classification is a supervised learning task where example documents and their categories are available for learning in advance. Document classification is used for automated (rather than manual) categorization for documents. In Ref. [WAD⁺99], Weiss *et al.* studied the application of decision tree methods in order to categorize text documents. McCallum and Nigam [MN98] used a Bayesian (probabilistic) approach for document classification.

There are several reasons why information retrieval methods that deal with traditional text documents are not entirely suitable for web content mining. First, web documents contain additional markup elements (HTML tags) which are not found in plain-text documents. These elements can be a source of additional knowledge about the documents. As we saw above,

there is a branch of web mining (web structure mining) that attempts to exploit the hyperlink information found in web documents. This information is not available for plain-text documents, so we must find ways to incorporate this information into our data representations during web mining. This is a major limitation of existing web content mining methods, as they either require discarding such information to arrive at traditional plain-text vector representations or they necessitate new or altered data mining procedures which explicitly take the additional information into account. Second, the web is highly heterogeneous, especially when compared to document corpora that are related to a single topic or field of interest. For example, the term “Amazon” can refer to many things on the Internet: an on-line book store, a rain forest, a river, or a person. Thus we may be unable to use specific domain knowledge (such as specialized stop word or synonym lists) that we could otherwise utilize in a system developed for a well-defined, homogeneous document collection. Additionally, web documents have a wide variation in size, style, layout, languages, *etc.* We must deal with these variations. Third, traditional information retrieval methods may not scale well to the size or dynamic nature of the Internet. Web pages tend to change often with time (they are updated, they are moved to another location, *etc.*) and thus techniques used in traditional information retrieval systems, such as those related to generation of indices or representatives of documents, can lead to out-of-date results. The web contains hundreds of millions of pages, some of which change frequently and must be re-examined periodically. These last two points are especially problematic for web document categorization methods, since creating an adequate training set to encompass all desired categories and updating it to deal with changing conditions is extremely difficult.

In contrast to the methods mentioned above, the work presented here represents the first time web document content itself has been modeled and retained using graphs in a web mining method. Note that, as we mentioned above, graph representations have been used for web mining (*e.g.* web graphs in web structure mining). However, the difference is that those graphs represent the documents (nodes) and their relationships through hyperlinks (edges). Our graphs represent the textual content of web documents through words (nodes) and adjacency information (edges), as will be discussed in detail in Chapter 3. Only recently have a few papers appeared in the literature that deal with representing the web documents themselves using graphs. Lopresti and Wilfong compared web documents using a graph representation that primarily utilizes HTML parse informa-

tion, in addition to hyperlink and content order information [LW01][LW04]. 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 utilize the graphs themselves to determine document similarity rather than a set of extracted features. Liang and Doermann represent the physical layout of document images as graphs [LD02]. 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.* they capture the 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. Another recently reported approach [DG03][PG03] takes both the content and structure of web documents into account. Documents are initially represented by graphs according to either naturally occurring elements (*e.g.* pages, sections, paragraphs, *etc.*) or XML markup structure. However, rather than manipulating the graphs directly as data, as is done in our approach, probabilistic information is extracted to create Bayesian networks for retrieval or classification. Thus this method is more similar to a probabilistic learning model, rather than a purely graph-theoretical one.

1.3 Overview of Remaining Chapters

In this book we will be presenting important contributions that help improve the clustering and classification of web documents. This is accomplished by representing their content with a more versatile graph model, which can retain additional information that is not captured when using a vector representation. We will describe graph-theoretical extensions to existing, proven clustering and classification methods that for the first time allow them to deal with data represented by graphs rather than vectors. This approach has two main benefits. First, it allows us to keep the inherent structure of the original data by modeling web document content as a graph, rather than having to arrive at numeric feature vectors that contain only term frequency information. Second, we do not need to develop new algorithms or frameworks to deal with the graphs: we can simply apply straightforward extensions to go from classical data mining algorithms that use numerical vectors to those that can handle graphs. It is our contention

that by using graphs to keep information that is usually lost we can improve clustering and classification performance over the usual vector model for the same algorithm. We will explore this contention through a series of experiments, using the well known k -means clustering and k -nearest neighbors (k -NN) classification algorithms. A surprising realization during our experiments is that, with careful selection of the graph representation model for an application, we can achieve polynomial time complexity for the graph similarity procedure. In the general case this is an NP-Complete problem. Note that these techniques are not limited to web documents or even text documents in general: they allow any data sets that are complex enough to require representation by graphs (*e.g.* software code, computer networks, maps, images, *etc.*) to now be clustered or classified using classical, popular methods without loss of the inherent structural information.

The remainder of this book is organized as follows. A review of graph similarity techniques as well as the mathematical definitions and notation needed for the methods we propose is presented in Chapter 2. We will review the maximum common subgraph approach, which is what we use for graph similarity in our web mining algorithms. We will also give introductions to some other related methods including graph isomorphism, graph edit distance, and median of a set of graphs. The five graph-theoretical distance measures we use will also be presented here.

In Chapter 3 we will show how web documents can be modeled as graphs using six different methods: *standard*, *simple*, *n -distance*, *n -simple-distance*, *absolute frequency*, and *relative frequency*. We also give some implementation details relating to the creation of document models from their content, such as stop word removal and stemming. The complexity analysis related to these representations is also given. We will describe the three web document data sets used for our experiments in this chapter as well.

In Chapter 4 we will describe an extension to the k -means clustering algorithm that allows the utilization of graphs instead of vectors [SLBK03b] and illustrate its usefulness by applying it to the problem of clustering a collection of web documents. We will define the clustering performance indexes that will be used to measure clustering performance in our experiments. We present experiments comparing our novel approach to the traditional vector methods when using different graph distance measures and graph representations of documents. The effect of graph size on the clustering performance is also investigated. We demonstrate how multidimensional scaling can be used with graphs to allow for visualization of the graph space and clusters of documents. Further, we will measure the per-

formance of our clustering method when combined with the global k -means algorithm presented in [LVV03], which provides a deterministic method of finding “good” initial cluster center positions for k -means [SLBK03c]. Previous experimental results have shown that initialization with global k -means can lead to clustering performance which is as good or better than random initializations, and we will investigate whether this holds true for our methods and data sets as well. We also use this method to examine the question of the optimum number of clusters for the document collections. We will use the global k -means initializations for a range of different k values (numbers of clusters) and measure performance with additional cluster validity indexes.

In Chapter 5 we compare the traditional vector model representation to our new graph model in the context of the document classification task rather than clustering. We introduce a graph-based extension of the popular k -nearest neighbors (k -NN) classification algorithm [SLBK03a][SLBK04] and measure classification accuracy using the leave-one-out approach over all three web document collections. We select several values for the number of nearest neighbors, k , and look at the classification performance as a function of the size of the graphs representing each document. We also examine the effect of different graph-theoretical distance measures and graph representations on classification accuracy as in Chapter 4. Further, we compare execution times of both our graph-based approach and the traditional vector approach using cosine similarity or Jaccard similarity. Finally, we show how random node selection, a graph-theoretic analogue of the random feature subset selection method used with vector representations, can be used to build classifier ensembles that use both structural (graph-based) and statistical (vector-based) classifiers.

In Chapter 6 we describe a web search clustering system we developed within the graph-theoretic framework. This system takes a user’s query, submits it to conventional web search engines, and then organizes the resulting web documents by a hierarchical clustering procedure. Each cluster is represented by a label describing the cluster contents. Our system originally used binary vector representations of web documents and cluster labels. We later upgraded our algorithms to use graph-representations. The most notable benefit of this change is that the cluster labels under the graph-based approach preserved the correct term ordering used in the phrases describing the clusters. An additional benefit is that the term co-occurrence information captured by the edges in the graphs allows for additional cluster organization, from independent terms (which are more general) to phrases

(which are more specific). This chapter gives the reader an example of an actual, practical web mining system which utilizes graph-theoretic methods, including the details of migrating the system's algorithms from vectors to graphs, comparisons with similar systems, and examples of the system's output.

Concluding remarks and possible future extensions of the current work will be given in Chapter 7.