11

WEB MINING AND TEXT MINING

Chapter Objectives

- Explain the specifics of Web mining.
- Introduce a classification of basic Web-mining subtasks.
- Illustrate the possibilities of Web mining using Hyperlink-Induced Topic Search (HITS), LOGSOM, and Path Traversal algorithms.
- Describe query-independent ranking of Web pages.
- Formalize a text-mining framework specifying the refining and distillation phases.
- Outline latent semantic indexing.

11.1 WEB MINING

In a distributed information environment, documents or objects are usually linked together to facilitate interactive access. Examples for such information-providing environments include the World Wide Web (WWW) and online services such as America

Data Mining: Concepts, Models, Methods, and Algorithms, Second Edition. Mehmed Kantardzic.

Online, where users, when seeking information of interest, travel from one object to another via facilities such as hyperlinks and URL addresses. The Web is an ever-growing body of hypertext and multimedia documents. As of 2008, Google had discovered 1 trillion Web pages. The Internet Archive, which makes regular copies of many publicly available Web pages and media files, was three petabytes in size as of March 2009. Several billions of pages are added each day to that number. As the information offered in the Web grows daily, obtaining that information becomes more and more tedious. The main difficulty lies in the semi-structured or unstructured Web content that is not easy to regulate and where enforcing a structure or standards is difficult. A set of Web pages lacks a unifying structure and shows far more authoring styles and content variation than that seen in traditional print document collections. This level of complexity makes an "off-the-shelf" database-management and information-retrieval solution very complex and almost impossible to use. New methods and tools are necessary. Web mining may be defined as the use of data-mining techniques to automatically discover and extract information from Web documents and services. It refers to the overall process of discovery, not just to the application of standard data-mining tools. Some authors suggest decomposing Web-mining task into four subtasks:

1. *Resource Finding.* This is the process of retrieving data, which is either online or offline, from the multimedia sources on the Web, such as news articles, forums, blogs, and the text content of HTML documents obtained by removing the HTML tags.
2. *Information Selection and Preprocessing.* This is the process by which different kinds of original data retrieved in the previous subtask is transformed. These transformations could be either a kind of preprocessing such as removing stop words and stemming or a preprocessing aimed at obtaining the desired representation, such as finding phrases in the training corpus and representing the text in the first-order logic form.
3. *Generalization.* Generalization is the process of automatically discovering general patterns within individual Web sites as well as across multiple sites. Different general-purpose machine-learning techniques, data-mining techniques, and specific Web-oriented methods are used.
4. *Analysis.* This is a task in which validation and/or interpretation of the mined patterns is performed.

There are three factors affecting the way a user perceives and evaluates Web sites through the data-mining process: (1) Web-page content, (2) Web-page design, and (3) overall site design including structure. The first factor is concerned with the goods, services, or data offered by the site. The other factors are concerned with the way in which the site makes content accessible and understandable to its users. We distinguish between the design of individual pages and the overall site design, because a site is not a simply a collection of pages; it is a network of related pages. The users will not engage in exploring it unless they find its structure simple and intuitive. Clearly, understanding user-access patterns in such an environment will not only help improve the system

design (e.g., providing efficient access between highly correlated objects, better authoring design for WWW pages), it will also lead to better marketing decisions. Commercial results will be improved by putting advertisements in proper places, better customer/user classification, and understanding user requirements better through behavioral analysis.

No longer are companies interested in Web sites that simply direct traffic and process orders. Now they want to maximize their profits. They want to understand customer preferences and customize sales pitches to individual users. By evaluating a user's purchasing and browsing patterns, e-vendors want to serve up (in real time) customized menus of attractive offers e-buyers cannot resist. Gathering and aggregating customer information into e-business intelligence is an important task for any company with Web-based activities. e-Businesses expect big profits from improved decision making, and therefore e-vendors line up for data-mining solutions.

Borrowing from marketing theory, we measure the efficiency of a Web page by its contribution to the success of the site. For an online shop, it is the ratio of visitors that purchased a product after visiting this page to the total number of visitors that accessed the page. For a promotional site, the efficiency of the page can be measured as the ratio of visitors that clicked on an advertisement after visiting the page. The pages with low efficiency should be redesigned to better serve the purposes of the site. Navigation-pattern discovery should help in restructuring a site by inserting links and redesigning pages, and ultimately accommodating user needs and expectations.

To deal with problems of Web-page quality, Web-site structure, and their use, two families of Web tools emerge. The first includes tools that accompany the users in their navigation, learn from their behavior, make suggestions as they browse, and, occasionally, customize the user profile. These tools are usually connected to or built into parts of different search engines. The second family of tools analyzes the activities of users offline. Their goal is to provide insights into the semantics of a Web site's structure by discovering how this structure is actually utilized. In other words, knowledge of the navigational behavior of users is used to predict future trends. New data-mining techniques are behind these tools, where Web-log files are analyzed and information is uncovered. In the next four sections, we will illustrate Web mining with four techniques that are representative of a large spectrum of Web-mining methodologies developed recently.

11.2 WEB CONTENT, STRUCTURE, AND USAGE MINING

One possible categorization of Web mining is based on which part of the Web one mines. There are three main areas of Web mining: Web-content mining, Web-structure mining, and Web-usage mining. Each area is classified by the type of data used in the mining process. Web-content mining uses Web-page content as the data source for the mining process. This could include text, images, videos, or any other type of content on Web pages. Web-structure mining focuses on the link structure of Web pages. Web-usage mining does not use data from the Web itself but takes as input data recorded from the interaction of users using the Internet.

The most common use of Web-content mining is in the process of searching. There are many different solutions that take as input Web-page text or images with the intent of helping users find information that is of interest to them. For example, crawlers are currently used by search engines to extract Web content into the indices that allow immediate feedback from searches. The same crawlers can be altered in such a way that rather than seeking to download all reachable content on the Internet, they can be focused on a particular topic or area of interest.

To create a focused crawler, a classifier is usually trained on a number of documents selected by the user to inform the crawler as to the type of content to search for. The crawler will then identify pages of interest as it finds them and follow any links on that page. If those links lead to pages that are classified as not being of interest to the user, then the links on that page will not be used further by the crawler.

Web-content mining can also be seen directly in the search process. All major search engines currently use a list-like structure to display search results. The list is ordered by a ranking algorithm behind the scenes. An alternative view of search results that has been attempted is to provide the users with clusters of Web pages as results rather than individual Web pages. Often a hierarchical clustering that will give multiple topic levels is performed.

As an example consider the Web site Clusty.com, which provides a clustered view of search results. If one keyword were to enter [jaguar] as a search onto this Web site, one sees both a listing of topics and a list of search results side-by-side, as shown in Figure 11.1. This specific query is ambiguous, and the topics returned show that ambiguity. Some of the topics returned include: cars, Onca, Panthery (animal kingdom), and

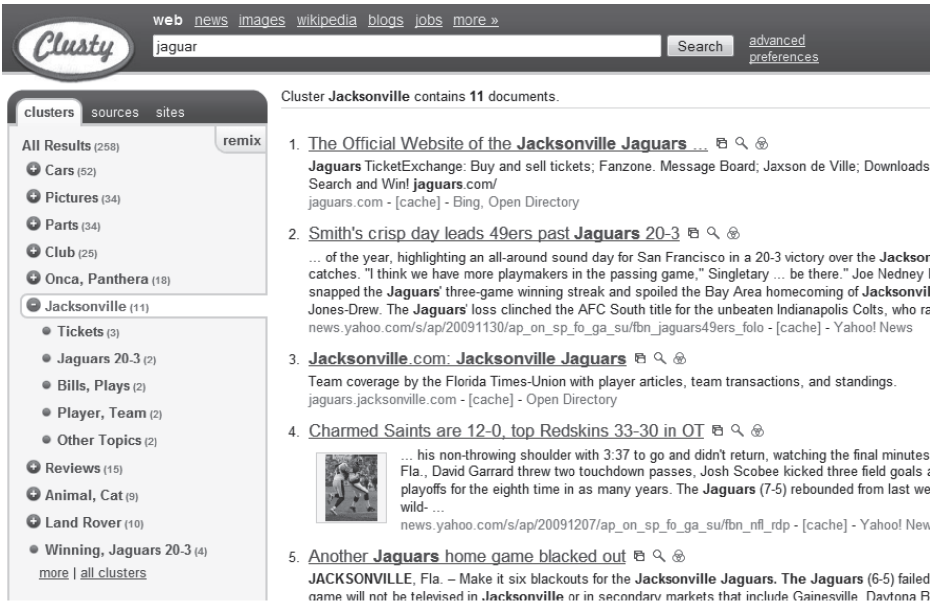


Figure 11.1. Example query from Clusty.com.

Jacksonville (American football team). Each of these topics can be expanded to show all of the documents returned for this query in a given topic.

Web-structure mining considers the relationships between Web pages. Most Web pages include one or more hyperlinks. These hyperlinks are assumed in structure mining to provide an endorsement by the linking page of the page linked. This assumption underlies PageRank and HITS, which will be explained later in this section.

Web-structure mining is mainly used in the information retrieval (IR) process. PageRank may have directly contributed to the early success of Google. Certainly the analysis of the structure of the Internet and the interlinking of pages currently contributes to the ranking of documents in most major search engines.

Web-structure mining is also used to aid in Web-content mining processes. Often, classification tasks will consider features from the content of the Web page and may consider the structure of the Web pages. One of the more common features in Web-mining tasks taken from structure mining is the use of anchor text. Anchor text refers to the text displayed to users on an HTML hyperlink. Oftentimes the anchor text provides summary keywords not found on the original Web page. The anchor text is often as brief as search-engine queries. Additionally, if links are endorsements of Web pages, then the anchor text offers keyword-specific endorsements.

Web-usage mining refers to the mining of information about the interaction of users with Web sites. This information may come from server logs, logs recorded by the client's browser, registration form information, and so on. Many usage questions exist, such as the following: How does the link structure of the Web site differ from how users may prefer to traverse the page? Where are the inefficiencies in the e-commerce process of a Web site? What segments exist in our customer base?

There are some key terms in Web-usage mining that require defining. A "visitor" to a Web site may refer to a person or program that retrieves a Web page from a server. A "session" refers to all page views that took place during a single visit to a Web site. Sessions are often defined by comparing page views and determining the maximum allowable time between page views before a new session is defined. Thirty minutes is a standard setting.

Web-usage mining data often requires a number of preprocessing steps before meaningful data mining can be performed. For example, server logs often include a number of computer visitors that could be search-engine crawlers, or any other computer program that may visit Web sites. Sometimes these "robots" identify themselves to the server passing a parameter called "user agent" to the server that uniquely identifies them as robots. Some Web page requests do not make it to the Web server for recording, but instead a request may be filled by a cache used to reduce latency.

Servers record information on a granularity level that is often not useful for mining. For a single Web-page view, a server may record the browsers' request for the HTML page, a number of requests for images included on that page, the Cascading Style Sheets (CSS) of a page, and perhaps some JavaScript libraries used by that Web page. Often there will need to be a process to combine all of these requests into a single record. Some logging solutions sidestep this issue by using JavaScript embedded into the Web page to make a single request per page view to a logging server. However, this approach

has the distinct disadvantage of not recording data for users that have disabled JavaScript in their browser.

Web-usage mining takes advantage of many of the data-mining approaches available. Classification may be used to identify characteristics unique to users that make large purchases. Clustering may be used to segment the Web-user population. For example, one may identify three types of behavior occurring on a university class Web site. These three behavior patterns could be described as users cramming for a test, users working on projects, and users consistently downloading lecture notes from home for study. Association mining may identify two or more pages often viewed together during the same session, but that are not directly linked on a Web site. Sequence analysis may offer opportunities to predict user navigation patterns and therefore allow for within site, targeted advertisements. More on Web-usage mining will be shown through the LOGSOM algorithm and through the section on "Mining path traversal patterns."

11.3 HITS AND LOGSOM ALGORITHMS

To date, index-based search engines for the Web have been the primary tool with which users search for information. Experienced Web surfers can make effective use of such engines for tasks that can be solved by searching with tightly constrained keywords and phrases. These search engines are, however, unsuited for a wide range of less precise tasks. How does one select a subset of documents with the most value from the millions that a search engine has prepared for us? To distill a large Web-search topic to a size that makes sense to a human user, we need a means of identifying the topic's most authoritative Web pages. The notion of authority adds a crucial dimension to the concept of relevance: We wish to locate not only a set of relevant pages, but also those that are of the highest quality.

It is important that the Web consists not only of pages, but also hyperlinks that connect one page to another. This hyperlink structure contains an enormous amount of information that can help to automatically infer notions of authority. Specifically, the creation of a hyperlink by the author of a Web page represents an implicit endorsement of the page being pointed to. By mining the collective judgment contained in the set of such endorsements, we can gain a richer understanding of the relevance and quality of the Web's contents. It is necessary for this process to uncover two important types of pages: *authorities*, which provide the best source of information about a given topic and *hubs*, which provide a collection of links to authorities.

Hub pages appear in a variety of forms, ranging from professionally assembled resource lists on commercial sites to lists of recommended links on individual home pages. These pages need not themselves be prominent, and working with hyperlink information in hubs can cause much difficulty. Although many links represent some kind of endorsement, some of the links are created for reasons that have nothing to do with conferring authority. Typical examples are navigation and paid advertisement hyperlinks. A hub's distinguishing feature is that they are potent conferrers of authority on a focused topic. We can define a *good hub* if it is a page that points to many good

authorities. At the same time, a good authority page is a page pointed to by many good hubs. This mutually reinforcing relationship between hubs and authorities serves as the central idea applied in the *HITS algorithm* that searches for good hubs and authorities. The two main steps of the HITS algorithm are

1. *the sampling component*, which constructs a focused collection of Web pages likely to be rich in relevant information, and
2. *the weight-propagation component*, which determines the estimates of hubs and authorities by an iterative procedure and obtains the subset of the most relevant and authoritative Web pages.

In the sampling phase, we view the Web as a directed graph of pages. The HITS algorithm starts by constructing the subgraph in which we will search for hubs and authorities. Our goal is a subgraph rich in relevant, authoritative pages. To construct such a subgraph, we first use query terms to collect a root set of pages from an index-based search engine. Since many of these pages are relevant to the search topic, we expect that at least some of them are authorities or that they have links to most of the prominent authorities. We therefore expand the root set into a base set by including all the pages that the root-set pages link to, up to a designated cutoff size. This base set V typically contains from 1000 to 5000 pages with corresponding links, and it is a final result of the first phase of HITS.

In the weight-propagation phase, we extract good hubs and authorities from the base set V by giving a concrete numeric interpretation to all of them. We associate a nonnegative authority weight a_p and a nonnegative hub weight h_p with each page $p \in V$. We are interested only in the relative values of these weights; therefore, normalization is applied so that their total sum remains bounded. Since we do not impose any prior estimates, we set all a and h values to a uniform constant initially. The final weights are unaffected by this initialization.

We now update the authority and hub weights as follows. If a page is pointed to by many good hubs, we would like to increase its authority weight. Thus, we update the value of a_p for the page p to be the sum of h_q over all pages q that link to p:

$$a_p = \sum h_q, \forall q \text{ such that } q \rightarrow p$$

where the notation $q \rightarrow p$ indicates that page q links to page p. In a strictly dual fashion, if a page points to many good authorities, we increase its hub weight

$$h_p = \sum a_q, \forall q \text{ such that } p \rightarrow q$$

There is a more compact way to write these updates. Let us number the pages $\{1,2, \ldots, n\}$, and define their adjacency matrix A to be $n \times n$ matrix whose $(i, j)^{th}$ element is equal to 1 if page i links to page j, and 0 otherwise. All pages at the beginning of the computation are both hubs and authorities, and, therefore, we can represent them as vectors

$$a = \{a_1, a_2, \ldots, a_n\} \text{ and}$$

$$h = \{h_1, h_2, \ldots, h_n\}$$

Our update rules for authorities and hubs can be written as

$$a = A^T h$$

$$h = A\, a$$

or, substituting one into another relation,

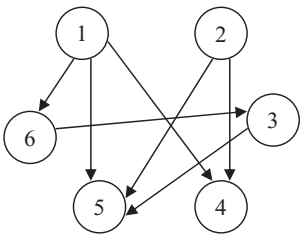
$$a = A^T h = A^T A\, a = (A^T A)\, a$$

$$h = A\, a = A\, A^T h = (A\, A^T)\, h$$

These are relations for iterative computation of vectors a and h. Linear algebra tells us that this sequence of iterations, when normalized, converges to the principal eigenvector of A^TA. This says that the hub and authority weights we compute are truly an intrinsic feature of the linked pages collected, not an artifact of our choice of initial weights. Intuitively, the pages with large weights represent a very dense pattern of linkage, from pages of large hub weights to pages of large authority weights. Finally, HITS produces a short list consisting of the pages with the largest hub weights and the pages with the largest authority weights for the given search topic. Several extensions and improvements of the HITS algorithm are available in the literature. Here we will illustrate the basic steps of the algorithm using a simple example.

Suppose that a search engine has selected six relevant documents based on our query, and we want to select the most important authority and hub in the available set. The selected documents are linked into a directed subgraph, and the structure is given in Figure 11.2a, while corresponding adjacency matrix A and initial weight vectors a and h are given in Figure 11.2b.

The first iteration of the HITS algorithm will give the changes in the a and h vectors:



(a)

$$A = \begin{bmatrix} 0&0&0&1&1&1 \\ 0&0&0&1&1&0 \\ 0&0&0&0&1&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&1&0&0&0 \end{bmatrix}$$

$a = \{0.1, 0.1, 0.1, 0.1, 0.1, 0.1\}$
$h = \{0.1, 0.1, 0.1, 0.1, 0.1, 0.1\}$

(b)

Figure 11.2. Initialization of the HITS algorithm. (a) Subgraph of the linked pages; (b) adjacency matrix A and weight vectors for the given graph.

$$a = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0.1 \\ 0.1 \\ 0.1 \\ 0.1 \\ 0.1 \\ 0.1 \end{bmatrix}$$

$$= [0 \quad 0 \quad 0.1 \quad 0.5 \quad 0.6 \quad 0.3]$$

$$h = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0.1 \\ 0.1 \\ 0.1 \\ 0.1 \\ 0.1 \\ 0.1 \end{bmatrix}$$

$$= [0.6 \quad 0.5 \quad 0.3 \quad 0 \quad 0 \quad 0.1]$$

Even this single iteration of the HITS algorithm shows that, in the given set of documents, document-5 has the most authority and document-1 is the best hub. Additional iterations will correct the weight factors for both vectors, but the obtained ranking of authorities and hubs will stay unchanged for this example.

The continuous growth in the size and use of the Internet creates difficulties in the search for information. Resource discovery is frustrating and inefficient when simple keyword searches can convey hundreds of thousands of documents as results. Many of them are irrelevant pages, some of them may have been moved, and some abandoned. While the first Web-mining algorithm HITS is primarily based on static information describing the Web-site structure, the second one, LOGSOM, uses dynamic information about a user's behavior. LOGSOM is a sophisticated method that organizes the layout of the information in a user-interpretable graphic form. The LOGSOM system uses self-organizing maps (SOM) to organize Web pages into a two-dimensional (2-D) table, according to users' navigation patterns. The system organizes Web pages according to the interest of Web users by keeping track of their navigation paths.

The SOM technique is used as the most appropriate technique for the problem of Web-page organization because of its strength not only in grouping data points into clusters, but also in graphically representing the relationship among clusters. The system starts with a Web-log file indicating the date, time, and address of the requested Web pages as well as the IP address of the user's machine. The data are grouped into meaningful transactions or sessions, where a transaction is defined by a set of user-requested Web pages. We assume that there is a finite set of unique URLs:

$$U = \{url_1, url_2, \ldots, url_n\}$$

and a finite set of m user transactions:

$$T = \{t_1, t_2, \ldots, t_m\}$$

TABLE 11.1. Transactions Described by a Set of URLs

	URL_1	URL_2	. . .	URL_n
t_1	0	1		1
t_2	1	1		0
. . .				
t_m	0	0		0

TABLE 11.2. Representing URLs as Vectors of Transaction Group Activity

	Transaction Groups			
	1	2	. . .	k
URL_1	15	0		2
URL_2	2	1		10
. . .				
URL_n	0	1		2

Transactions are represented as a vector with binary values u_i:

$$t = \{u_1, u_2, \ldots, u_n\}$$

where

$$u_i = \begin{cases} 1 & \text{if } url_i \in t \\ 0 & \text{otherwise} \end{cases}$$

Preprocessed log files can be represented as a binary matrix. One example is given in Table 11.1.

Since the dimensions of a table ($n \times m$) for real-world applications would be very large, especially as input data to SOM, a reduction is necessary. By using the K-means clustering algorithm, it is possible to cluster transactions into prespecified number k ($k << m$) of transaction groups. An example of a transformed table with a new, reduced data set is represented in Table 11.2, where the elements in the rows represent the total number of times a group accessed a particular URL (the form of the table and values are only one illustration, and they are not directly connected with the values in Table 11.1).

The new, reduced table is the input for SOM processing. Details about the application of SOM as a clustering technique and the settings of their parameters are given in the previous chapter. We will explain only the final results and their interpretation in terms of Web-page analysis. Each URL will be mapped onto a SOM based on its similarity with other URLs in terms of user usage or, more precisely, according to users' navigation patterns (transaction group "weights" in Table 11.2). Suppose that the SOM is a 2-D map with $p \times p$ nodes, where $p \times p \geq n$, then a typical result of SOM processing is given in Table 11.3. The dimensions and values in the table are not the results of any

TABLE 11.3. A Typical SOM Generated by the Description of URLs

	1	2	3	. . .	p
1	2		1		15
2	3	1	10	. . .	
. . .					
p		54		. . .	11

computation with values in Tables 11.1 and 11.2, but a typical illustration of the SOM's final presentation.

The SOM organizes Web pages into similar classes based on users' navigation patterns. The blank nodes in the table show that there are no corresponding URLs, while the numbered nodes indicate the number of URLs contained within each node (or within each class). The distance on the map indicates the similarity of the Web pages measured by the user-navigation patterns. For example, the number 54 in the last row shows that 54 Web pages are grouped in the same class because they have been accessed by similar types of people, as indicated by their transaction patterns. Similarity here is measured not by similarity of content but by similarity of usage. Therefore, the organization of the Web documents in this graphical representation is based solely on the users' navigation behavior.

What are the possible applications of the LOGSOM methodology? The ability to identify which Web pages are being accessed by a company's potential customers gives the company information to make improved decisions. If one Web page within a node successfully refers clients to the desired information or desired page, the other pages in the same node are likely to be successful as well. Instead of subjectively deciding where to place an Internet advertisement, the company can now decide objectively, supported directly by the user-navigation patterns.

11.4 MINING PATH–TRAVERSAL PATTERNS

Before improving a company's Web site, we need a way of evaluating its current usage. Ideally, we would like to evaluate a site based on the data automatically recorded on it. Each site is electronically administered by a Web server, which logs all activities that take place in it in a file called a Web-server log. All traces left by the Web users are stored in this log. Therefore, from these log files we can extract information that indirectly reflects the site's quality by applying data-mining techniques. We can mine data to optimize the performance of a Web server, to discover which products are being purchased together, or to identify whether the site is being used as expected. The concrete specification of the problem guides us through different data-mining techniques applied to the same Web-server log.

While the LOGSOM methodology is concentrated on similarity of Web pages, other techniques emphasize the similarity of a user's paths through the Web. Capturing

user-access patterns in a Web environment is referred to as *mining path–traversal patterns*. It represents an additional class of data-mining techniques, which is showing great promise. Note that because users travel along information paths to search for the desired information, some objects or documents are visited because of their location rather than their content. This feature of the traversal pattern unavoidably increases the difficulty of extracting meaningful information from a sequence of traversal data, and explains the reason why current Web-usage analyses are mainly able to provide statistical information for traveling points, but not for traveling paths. However, as these information-providing services become increasingly popular, there is a growing demand for capturing user-traveling behavior to improve the quality of such services.

We first focus on the theory behind the navigational patterns of users in the Web. It is necessary to formalize known facts about navigation: that not all pages across a path are of equal importance, and that users tend to revisit pages previously accessed. To achieve a data-mining task, we define a navigation pattern in the Web as a generalized notion of a sequence, the materialization of which is the directed-acyclic graph. A sequence is an ordered list of items, in our case Web pages, ordered by time of access. The log file L is a multiset of recorded sequences. It is not a simple set, because a sequence may appear more than once.

When we want to observe sequence s as a concatenation of the consecutive subsequences x and y, we use the notation:

$$s = x\ y$$

The function *length*(s) returns the number of elements in the sequence s. The function *prefix*(s, i) returns the subsequence composed of the first i elements of s. If $s' = prefix(s,i)$, we say that s' is a prefix of s and is denoted as $s' \leq s$. Analysis of log files shows that Web users tend to move backwards and revisit pages with a high frequency. Therefore, a log file may contain duplicates. Such revisits may be part of a guided tour or may indicate disorientation. In the first case, their existence is precious as information and should be retained. To model cycles in a sequence, we label each element of the sequence with its occurrence number within the sequence, thus distinguishing between the first, second, third, and so on occurrence of the same page.

Moreover, some sequences may have common prefixes. If we merge all common prefixes together, we transform parts of the log file into a tree structure, each node of which is annotated with the number of sequences having the same prefix up to and including this node. The tree contains the same information as the initial log file. Hence, when we look for frequent sequences, we can scan the tree instead of the original log multiset. On the tree, a prefix shared among k sequences appears and gets tested only once.

Sequence mining can be explained as follows: Given a collection of sequences ordered in time, where each sequence contains a set of Web pages, the goal is to discover sequences of maximal length that appear more frequently than a given percentage threshold over the whole collection. A frequent sequence is maximal if all sequences containing it have a lower frequency. This definition of the sequence-mining problem implies that the items constituting a frequent sequence need not necessarily occur

adjacent to each other. They just appear in the same order. This property is desirable when we study the behavior of Web users because we want to record their intents, not their errors and disorientations.

Many of these sequences, even those with the highest frequencies, could be of a trivial nature. In general, only the designer of the site can say what is trivial and what is not. The designer has to read all patterns discovered by the mining process and discard unimportant ones. It would be much more efficient to automatically test data-mining results against the expectations of the designer. However, we can hardly expect a site designer to write down all combinations of Web pages that are considered typical; expectations are formed in the human mind in much more abstract terms. Extraction of informative and useful maximal sequences continues to be a challenge for researchers.

Although there are several techniques proposed in the literature, we will explain one of the proposed solutions for mining traversal patterns that consists of two steps:

(a) In a first step, an algorithm is developed to convert the original sequence of log data into a set of *traversal subsequences*. Each traversal subsequence represents a maximum forward reference from the starting point of a user access. It should be noted that this step of conversion would filter out the effect of backward references, which are mainly made for ease of traveling. The new reduced set of user-defined forward paths enables us to concentrate on mining meaningful user-access sequences.

(b) The second step consists of a separate algorithm for determining the frequent-traversal patterns, termed *large reference sequences*. A large reference sequence is a sequence that appears a sufficient number of times in the log database. In the final phase, the algorithm forms the *maximal references* obtained from large reference sequences. A maximal large sequence is a large reference sequence that is not contained in any other maximal reference sequence.

For example, suppose the traversal log of a given user contains the following path (to keep it simple, Web pages are represented by letters):

$$\text{Path} = \{\text{A B C D C B E G H G W A O U O V}\}$$

The path is transformed into the tree structure shown in Figure 11.3. The set of maximum forward references (MFR) found in step (a) after elimination of backward references is

$$\text{MFR} = \{\text{ABCD, ABEGH, ABEGW, AOU, AOV}\}.$$

When MFR have been obtained for all users, the problem of finding frequent-traversal patterns is mapped into one of finding frequently occurring consecutive subsequences among all MFR. In our example, if the threshold value is 0.4 (or 40%), large-reference sequences (LRS) with lengths 2, 3, and 4 are

$$\text{LRS} = \{\text{AB, BE, EG, AO, ABE, BEG, ABEG}\}$$

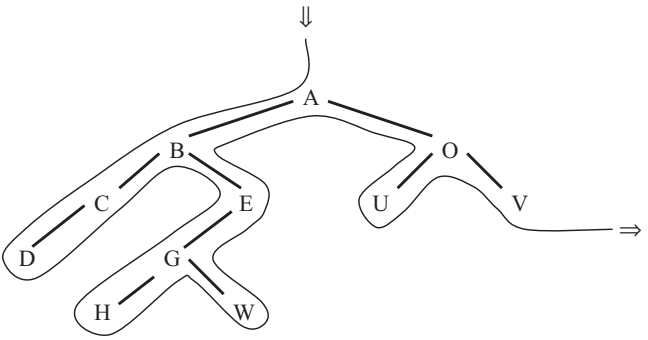


Figure 11.3. An example of traversal patterns.

Finally, with LRS determined, *maximal reference sequences* (MRS) can be obtained through the process of selection. The resulting set for our example is

$$\text{MRS} = \{\text{ABEG, AO}\}.$$

In general, these sequences, obtained from large log files, correspond to a frequently accessed pattern in an information-providing service.

The problem of finding LRS is very similar to that of finding frequent itemsets (occurring in a sufficient number of transactions) in association-rule mining. However, they are different from each other in that a reference sequence in the mining-traversal patterns has to be references in a given order, whereas a large itemset in mining association rules is just a combination of items in a transaction. The corresponding algorithms are different because they perform operations on different data structures: lists in the first case, sets in the second. As the popularity of Internet applications explodes, it is expected that one of the most important data-mining issues for years to come will be the problem of effectively discovering knowledge on the Web.

11.5 PAGERANK ALGORITHM

PageRank was originally published by Sergey Brin and Larry Page, the co-creators of Google. It likely contributed to the early success of Google. PageRank provides a global ranking of nodes in a graph. For search engines it provides a query-independent, authority ranking of all Web pages. PageRank has similar goals of finding authoritative Web pages to that of the HITS algorithm. The main assumption behind the PageRank algorithm is that every link from page *a* to page *b* is a vote by page *a* for page *b*. Not all votes are equal. Votes are weighted by the PageRank score of the originating node.

PageRank is based on the random surfer model. If a surfer were to randomly select a starting Web page, and at each time step the surfer were to randomly select a link on the current Web page, then PageRank could be seen as the probability that this random surfer is on any given page. Some Web pages do not contain any hyperlinks. In this model it is assumed that the random surfer selects a random Web page when exiting

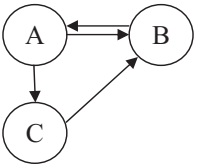


Figure 11.4. First example used to demonstrate PageRank.

pages with no hyperlinks. Additionally, there is some chance that the random surfer will stop following links and restart the process.

Computationally the PageRank (*Pr*) of page u can be computed as follows:

$$\Pr(u) = \frac{1-d}{N} + d\left(\sum_{v \in In(u)} \frac{\Pr(v)}{|Out(v)|}\right)$$

Here d is a dampening factor $0 \le d \le 1$ usually set to 0.85. N refers to the total number of nodes in the graph. The function $In(u)$ returns the set of nodes with edges pointing into node u. $|Out(v)|$ returns the number of nodes with edges pointing from v to that node. For example, if the Web-page connections are those given in Figure 11.4, and the current node under consideration were node B, then the following values would hold through all iterations: N = 3, In(B) = {A,C}, |Out(A)| = |{B,C}| = 2, and |Out(C)| = |{B}| = 1. The values for Pr(A) and Pr(C) would vary depending on the calculations from the previous iterations. The result is a recursive definition of PageRank. To calculate the PageRank of a given node, one must calculate the PageRank of all nodes with edges pointing into that given node.

Often PageRank is calculated using an iterative approach where all nodes are given an initial value for Pr of 1/N. Then during a single iteration we calculate what the PageRank of each node would be according to the current values of all nodes linking to that node. This process is repeated until the change between iterations is below some predetermined threshold or the maximum number of iterations is achieved. Let us consider an example graph with three nodes as follows:

Initially the Pr values are as follows:

$$\Pr(A) = 1/N = 0.333$$

$$\Pr(B) = 0.333$$

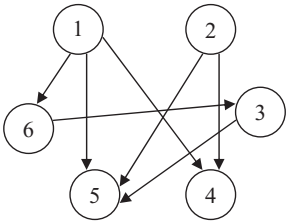
$$\Pr(C) = 0.333$$

The first iteration is as follows:

$$\Pr(A) = (1-0.85)/3 + 0.85(\Pr(B)/1) = 0.15/3 + 0.85(0.333) = 0.333$$

$$\begin{aligned}\Pr(B) &= (1-0.85)/3 + 0.85(\Pr(A)/2 + \Pr(C)/1) \\ &= 0.15/3 + 0.85(0.333/2 + 0.333) = 0.475\end{aligned}$$

$$\Pr(C) = (1-0.85)/3 + 0.85(\Pr(A)/2) = 0.15/3 + 0.85(0.333/2) = 0.192$$



(a)

Node	Page Rank
1	0.095
2	0.095
3	0.198
4	0.162
5	0.330
6	0.121

(b)

Figure 11.5. An example graph and scoring for PageRank also used with HITS. (a) Subgraph of linked pages; (b) calculated PageRank for given graph.

The second iteration then shows the passing of these values through the graph:

$$\Pr(A) = (1-0.85)/3 + 0.85(\Pr(B)/1) = 0.15/3 + 0.85(0.475) = 0.454$$

$$\begin{aligned}\Pr(B) &= (1-0.85)/3 + 0.85(\Pr(A)/2 + \Pr(C)/1) \\ &= 0.15/3 + 0.85(0.454/2 + 0.192) = 0.406\end{aligned}$$

$$\Pr(C) = (1-0.85)/3 + 0.85(\Pr(A)/2) = 0.15/3 + 0.85(0.454/2) = 0.243$$

If we carry this same procedure out to 100 iterations, we achieve the following results:

$$\Pr(A) = 0.388$$

$$\Pr(B) = 0.397$$

$$\Pr(C) = 0.215$$

Additional iterations produce the same results. From this we can see a stable ordering emerge. B has the largest PageRank value having two in-links, more than any other. However, page A is not far behind, since page B has only a single link to page A without dividing its PageRank value among a number of outbound links.

Next we consider the example used for the HITS algorithm, which is applied on the graph in Figure 11.5a. The PageRank of this graph with a dampening factor of 0.85 is given in Figure 11.5b after running 100 iterations. A reader may, for practice, check the results after the first iteration, or implement the PageRank algorithm and check final results in Figure 11.5b.

From the values given in Figure 11.5b we can see that node 5 has the highest PageRank by far and also has the highest in-degree or edges pointing in to it. Surprising perhaps is that node 3 has the next highest score, having a score higher than node 4, which has more in-edges. The reason is that node 6 has only one out-edge pointing to node 3, while the edges pointing to node 4 are each one of multiple out-edges. Lastly, as expected, the lowest ranked edges are those with no in-edges, nodes 1 and 2.

One of the main contributions of Google's founders is implementation and experimental evaluation of the PageRank algorithm. They included a database of web sites

with 161 million links, and the algorithm converge in 45 iterations. Repeated experiments with 322 million links converged in 52 iterations. These experiments were evidence that PageRank converges in *log(n)* time where *n* is number of links, and it is applicable for a growing Web. Of course, the initial version of the PageRank algorithm, explained in a simplified form in this text, had numerous modifications to evolve into the current commercial version implemented in the Google search engine.

11.6 TEXT MINING

Enormous amounts of knowledge reside today in text documents that are stored either within organizations or are freely available. Text databases are rapidly growing because of the increasing amounts of information available in electronic form, such as electronic publications, digital libraries, e-mail, and the World Wide Web. Data stored in most text databases are semi-structured, and special data-mining techniques, called text mining, have been developed for discovering new information from large collections of textual data.

In general, there are two key technologies that make online text mining possible. One is *Internet search* capabilities and the other is the *text analysis* methodology. *Internet search* has been around for a few years. With the explosion of Web sites in the past decade, numerous search engines designed to help users find content appeared practically overnight. Yahoo, Alta Vista, and Excite are three of the earliest, while Google and Bing are the most popular in recent years. Search engines operate by indexing the content in a particular Web site and allowing users to search these indices. With the new generation of Internet-search tools, users can gain relevant information by processing a smaller amount of links, pages, and indices.

Text analysis, as a field, has been around longer than Internet search. It has been a part of the efforts to make computers understand natural languages, and it is commonly thought of as a problem for artificial intelligence. Text analysis can be used anywhere where there is a large amount of text that needs to be analyzed. Although automatic processing of documents using different techniques does not allow the depth of analysis that a human can bring to the task, it can be used to extract key points in the text, categorize documents, and generate summaries in a situation when a large number of documents makes manual analysis impossible.

To understand the details of text documents you can either search for keywords, or you can try to categorize the semantic content of the document itself. When identifying keywords in text documents, you are looking at defining specific details or elements within documents that can be used to show connections or relationships with other documents. In the IR domain, documents have been traditionally represented in the vector space model. Documents are tokenized using simple syntactic rules (such as white-space delimiters in English), and tokens are transformed to canonical form (e.g., "reading" to "read," "is," "was," and "are" to "be"). Each canonical token represents an axis in a Euclidean space. Documents are vectors in this n-dimensional space. If a token *t* called term occurs *n* times in document *d*, then the *t*th coordinate of *d* is simply *n.* One may choose to normalize the length of the document to 1, using the L_1, L_2, or L_∞ norms:

$$\| d_1 \| = \sum_t n(d,t), \qquad \| d_2 \| = \sqrt{\sum_t n(d,t)^2}, \qquad \| d_\infty \| = \max_t n(d,t)$$

where n(*d*,*t*) is the number of occurrences of a term t in a document d. These representations do not capture the fact that some terms, also called keywords (like "algorithm"), are more important than others (like "the" and "is") in determining document content. If t occurs in n_t out of N documents, n_t/N gives a sense of rarity, and hence, the importance of the term. The inverse document frequency, IDF = 1 + log (n_t/N), is used to stretch the axes of the vector space differentially. Thus the *t*th coordinate of document d may be represented with the value $(n(d,t)/\|d_1\|) \times IDF(t)$ in the weighted vector space model. In spite of being extremely crude and not capturing any aspect of language or semantics, this model often performs well for its intended purpose. Also, in spite of minor variations, all these models of text regard documents as multisets of terms, without paying attention to ordering between terms. Therefore, they are collectively called bag-of-words models. Very often, the outputs from these keyword approaches can be expressed as relational data sets that may then be analyzed using one of the standard data-mining techniques.

Hypertext documents, usually represented as basic components on the Web, are a special type of text-based document that have hyperlinks in addition to text. They are modeled with varying levels of details, depending on the application. In the simplest model, hypertext can be regarded as directed graph (D, L) where D is the set of nodes representing documents or Web pages, and L is the set of links. Crude models may not need to include the text models at the node level, when the emphasis is on document links. More refined models will characterize some sort of joint distribution between the term distributions of a node with those in a certain neighborhood of the document in the graph.

Content-based analysis and partition of documents is a more complicated problem. Some progress has been made along these lines, and new text-mining techniques have been defined, but no standards or common theoretical background has been established in the domain. Generally, you can think of text categorization as comparing a document to other documents or to some predefined set of terms or definitions. The results of these comparisons can be presented visually within a semantic landscape in which similar documents are placed together in the semantic space and dissimilar documents are placed further apart. For example, indirect evidence often lets us build semantic connections between documents that may not even share the same terms. For example, "car" and "auto" terms co-occurring in a set of documents may lead us to believe that these terms are related. This may help us to relate documents with these terms as similar. Depending on the particular algorithm used to generate the landscape, the resulting topographic map can depict the strengths of similarities among documents in terms of Euclidean distance. This idea is analogous to the type of approach used to construct Kohonen feature maps. Given the semantic landscape, you may then extrapolate concepts represented by documents.

The automatic analysis of text information can be used for several different general purposes:

1. to provide an overview of the contents of a large document collection and organize them in the most efficient way;
2. to identify hidden structures between documents or groups of documents;
3. to increase the efficiency and effectiveness of a search process to find similar or related information; and
4. to detect duplicate information or documents in an archive.

Text mining is an emerging set of functionalities that are primarily built on text-analysis technology. Text is the most common vehicle for the formal exchange of information. The motivation for trying to automatically extract, organize, and use information from it is compelling, even if success is only partial. While traditional commercial text-retrieval systems are based on inverted text indices composed of statistics such as word occurrence per document, text mining must provide values beyond the retrieval of text indices such as keywords. Text mining is about looking for semantic patterns in text, and it may be defined as the process of analyzing text to extract interesting, nontrivial information that is useful for particular purposes.

As the most natural form of storing information is text, text mining is believed to have a commercial potential even higher than that of traditional data mining with structured data. In fact, recent studies indicate that 80% of a company's information is contained in text documents. Text mining, however, is also a much more complex task than traditional data mining as it involves dealing with unstructured text data that are inherently ambiguous. Text mining is a multidisciplinary field involving IR, text analysis, information extraction, natural language processing, clustering, categorization, visualization, machine learning, and other methodologies already included in the data-mining "menu"; even some additional specific techniques developed lately and applied on semi-structured data can be included in this field. Market research, business-intelligence gathering, e-mail management, claim analysis, e-procurement, and automated help desk are only a few of the possible applications where text mining can be deployed successfully. The text-mining process, which is graphically represented in Figure 11.6, consists of two phases:

- *text refining*, which transforms free-form text documents into a chosen intermediate form (IF), and
- *knowledge distillation*, which deduces patterns or knowledge from an IF.

An IF can be semi-structured, such as a conceptual-graph representation, or structured, such as a relational-data representation. IFs with varying degrees of complexity are suitable for different mining purposes. They can be classified as *document-based,* wherein each entity represents a document, or *concept-based*, wherein each entity represents an object or concept of interests in a specific domain. Mining a document-based IF deduces patterns and relationships across documents. Document clustering, visualization, and categorization are examples of mining from document-based IFs.

For a fine-grained, domain-specific, knowledge-discovery task, it is necessary to perform a semantic analysis and derive a sufficiently rich representation to capture the

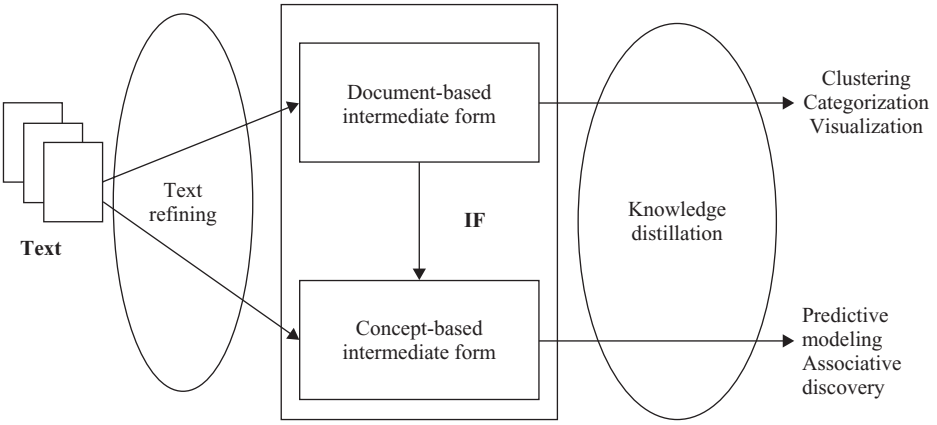


Figure 11.6. A text-mining framework.

relationship between objects or concepts described in the document. Mining a concept-based IF derives patterns and relationships across objects and concepts. These semantic-analysis methods are computationally expensive, and it is a challenge to make them more efficient and scalable for very large text corpora. Text-mining operations such as predictive modeling and association discovery fall in this category. A document-based IF can be transformed into a concept-based IF by realigning or extracting the relevant information according to the objects of interests in a specific domain. It follows that a document-based IF is usually domain-independent and a concept-based is a domain-dependent representation.

Text-refining and knowledge-distillation functions as well as the IF adopted are the basis for classifying different text-mining tools and their corresponding techniques. One group of techniques, and some recently available commercial products, focuses on document organization, visualization, and navigation. Another group focuses on text-analysis functions, IR, categorization, and summarization.

An important and large subclass of these text-mining tools and techniques is based on document visualization. The general approach here is to organize documents based on their similarities and present the groups or clusters of the documents as 2-D or 3-D graphics. IBM's Intelligent Miner and SAS Enterprise Miner are probably the most comprehensive text-mining products today. They offer a set of text-analysis tools that include tools for feature-extraction, clustering, summarization, and categorization; they also incorporate a text search engine. More examples of text-mining tools are given in Appendix A.

Domain knowledge, not used and analyzed by any currently available text-mining tool, could play an important role in the text-mining process. Specifically, domain knowledge can be used as early as in the text-refining stage to improve parsing efficiency and derive a more compact IF. Domain knowledge could also play a part in knowledge distillation to improve learning efficiency. All these ideas are still in their infancy, and we expect that the next generation of text-mining techniques and tools will improve the quality of information and knowledge discovery from text.

11.7 LATENT SEMANTIC ANALYSIS (LSA)

LSA is a method that was originally developed to improve the accuracy and effectiveness of IR techniques by focusing on semantic meaning of words across a series of usage contexts, as opposed to using simple string-matching operations. LSA is a way of partitioning free text using a statistical model of word usage that is similar to eigenvector decomposition and factor analysis. Rather than focusing on superficial features such as word frequency, this approach provides a quantitative measure of semantic similarities among documents based on a word's context.

Two major shortcomings to the use of term counts are synonyms and polysemes. Synonyms refer to different words that have the same or similar meanings but are entirely different words. In the case of the vector approach, no match would be found between a query using the term "altruistic" and a document using the word "benevolent" though the meanings are quite similar. On the other hand, polysemes are words that have multiple meanings. The term "bank" could mean a financial system, to rely upon, or a type of basketball shot. All of these lead to very different types of documents, which can be problematic for document comparisons.

LSA attempts to solve these problems, not with extensive dictionaries and natural language processing engines, but by using mathematical patterns within the data itself to uncover these relationships. We do this by reducing the number of dimensions used to represent a document using a mathematical matrix operation called singular value decomposition (SVD).

Let us take a look at an example data set. This very simple data set consists of five documents. We will show the dimension reduction steps of LSA on the first four documents ($d_{1..4}$), which will make up our training data. Then we will make distance comparisons to a fifth document (d_S) in our test set, using the nearest neighbor classification approach. The initial document set is:

- d_1: A bank will protect your money.
- d_2: A guard will protect a bank.
- d_3: Your bank shot is money.
- d_4: A bank shot is lucky.
- d_5: bank guard.

From the text data we derive a vector representation of our documents using only term counts. This vector representation could be thought of as a matrix with rows representing terms and columns representing documents. This representation may be seen in Figure 11.7.

The first step of LSA is to decompose the matrix representing our original data set of four documents, matrix A, using SVD as follows: $A = USV^T$. The calculation of SVD is beyond the scope of this text, but there are several computing packages that will perform this operation for you (such as R or MATLAB packages). The matrices resulting from this decomposition can be seen in Figure 11.8. The $\mathbf{U}$ and $\mathbf{V^T}$ matrices provide a vector of weights for terms and documents, respectively. Consider $\mathbf{V^T}$, this matrix

Terms		d1	d2	d3	d4		d5
a		1	2	0	1		0
bank		1	1	1	1		1
guard		0	1	0	0		1
is		0	0	1	1		0
lucky	**A** =	0	0	0	1	**d5** =	0
money		1	0	1	0		0
protect		1	1	0	0		0
shot		0	0	1	1		0
will		1	1	0	0		0
your		1	0	1	0		0

Figure 11.7. Initial term counts.

$$\mathbf{U} = \begin{bmatrix} -0.575 & 0.359 & 0.333 & 0.072 \\ -0.504 & -0.187 & 0.032 & -0.044 \\ -0.162 & 0.245 & 0.137 & -0.698 \\ -0.197 & -0.473 & 0.237 & -0.188 \\ -0.105 & -0.172 & 0.401 & 0.626 \\ -0.236 & -0.260 & -0.506 & 0.029 \\ -0.307 & 0.286 & -0.205 & 0.144 \\ -0.197 & -0.473 & 0.237 & -0.188 \\ -0.307 & 0.286 & -0.205 & 0.144 \\ -0.236 & -0.260 & -0.506 & 0.029 \end{bmatrix}, \mathbf{S} = \begin{bmatrix} 3.869 & 0.000 & 0.000 & 0.000 \\ 0.000 & 2.344 & 0.000 & 0.000 \\ 0.000 & 0.000 & 1.758 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.667 \end{bmatrix}, \mathbf{V^T} = \begin{bmatrix} -0.560 & -0.628 & -0.354 & -0.408 \\ 0.095 & 0.575 & -0.705 & -0.404 \\ -0.602 & 0.241 & -0.288 & 0.705 \\ 0.562 & -0.465 & -0.542 & 0.417 \end{bmatrix}$$

Figure 11.8. Singular value decomposition of initial data.

gives a new 4-D representation to each document where each document is described with a corresponding column. Each of the new dimensions is derived from the original 10 word count dimensions. For example, document 1 is now represented as follows: $d_1 = -0.56x_1 + 0.095x_2 - 0.602x_3 + 0.562x_4$, where each x_n represents one of the new, derived dimensions. The **S** matrix is a diagonal matrix of eigenvalues for each principal component direction.

With this initial step, we have already reduced the number of dimensions representing each document from 10 to four. We now show how one would go about further reducing the number of dimensions. When the data set contains many documents, the previous step is not enough to reduce the number of dimensions meaningfully. To perform further reduction, we first observe that matrix **S** provides us with a diagonal matrix of eigenvalues in descending order as follows: $\lambda_1, \ldots, \lambda_4 = \{3.869, 2.344, 1.758, 0.667\}$. We will be able to capture most of the variability of the data by retaining only the first k eigenvalues rather than all n terms (in our matrix S, n = 4). If for example k = 2, then we retain $(\lambda_1^2 + \lambda_2^2)/\sum_{i=1}^{4} \lambda_i^2 = 0.853$ or 85% of the variability when we move from the four new dimensions per document to only two. The rank 2 approximations can be seen in Figure 11.9. This rank 2 approximation is achieved by selecting the first k columns from **U**, the upper left k × k matrix from **S,** and the top k rows from $\mathbf{V^T}$ as shown in Figure 11.9.

The rank k approximation to $\mathbf{V^T}$ gives us our dimensionally reduced data set where each document is now described with only two dimensions. $\mathbf{V^T}$ can be thought of as a

$$U \approx U_k = \begin{bmatrix} -0.575 & 0.359 \\ -0.504 & -0.187 \\ -0.162 & 0.245 \\ -0.197 & -0.473 \\ -0.105 & -0.172 \\ -0.236 & -0.260 \\ -0.307 & 0.286 \\ -0.197 & -0.473 \\ -0.307 & 0.286 \\ -0.236 & -0.260 \end{bmatrix}, S \approx S_k = \begin{bmatrix} 3.869 & 0.000 \\ 0.000 & 2.344 \end{bmatrix}, V^T \approx V_k^T = \begin{bmatrix} -0.560 & -0.628 & -0.354 & -0.408 \\ 0.095 & 0.575 & -0.705 & -0.404 \end{bmatrix}$$

Figure 11.9. Rank 2 approximation of the singular value decomposition.

$$V'^T = A'^T U_k S_k^{-1}$$

$$V'^T = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} -0.575 & 0.359 \\ -0.504 & -0.187 \\ -0.162 & 0.245 \\ -0.197 & -0.473 \\ -0.105 & -0.172 \\ -0.236 & -0.260 \\ -0.307 & 0.286 \\ -0.197 & -0.473 \\ -0.307 & 0.286 \\ -0.236 & -0.260 \end{bmatrix} \begin{bmatrix} 0.258 & 0.000 \\ 0.000 & 0.427 \end{bmatrix} = \begin{bmatrix} -0.172 & 0.025 \end{bmatrix}$$

Figure 11.10. SVD calculations to produce a 2-D approximation for d_5.

transformation of the original document matrix and can be used for any of a number of text-mining tasks, such as classification and clustering. Improved results may be obtained using the newly derived dimensions, comparing against the data-mining tasks using the original word counts.

In most text-mining cases, all training documents (in our case 4) would have been included in matrix A and transformed together. Document five (d_5) was intentionally left out of these calculations to demonstrate a technique called "folding-in," which allows for a small number of documents to be transformed into the new reduced space and compared with a training document database. Matrices from our previous SVD calculations are used for this transformation. This is done using the following modified formula: $V'^T = A'^T U_k S_k^{-1}$. This equation is a rearrangement of terms from the initial SVD equation replacing the original data set, matrix A, with the term counts for document five (d_5) as matrix A'. The resulting multiplication can be seen in Figure 11.10. The result is the document d_5 represented in the reduced 2-D space, $d_5 = [-0.172, 0.025]$.

To visualize the transformed data, we have plotted our example documents, $d_{1..5}$, in Figure 11.11. We now perform the nearest neighbor classification algorithm. If the task is to disambiguate the term "bank," and the possible classes are "financial institution" and "basketball shot," then a review of the original text reveals that documents d_1, d_2, and d_5 are of the class "financial institution" and documents d_3 and d_4 are of the class "basketball shot." To classify test document d_5, we compare d_5 with each other

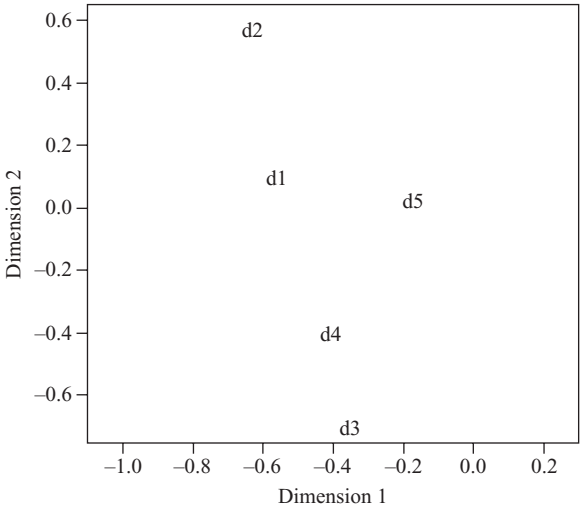


Figure 11.11. 2-D plot of documents and query using LSA.

TABLE 11.4. Use of Euclidean Distance to Find Nearest Neighbor to d_5 in Both 2-d and 10-d (the Smallest Distance Ranks First)

Comparison	*10-d (original)*		*2-d (LSA)*	
	Distance	Rank	Distance	Rank
d_1–d_5	2.449	3–4	0.394	1
d_2–d_5	2.449	3–4	0.715	3
d_3–d_5	2.236	1–2	0.752	4
d_4–d_5	2.236	1–2	0.489	2

document. The closest neighbor will determine the class of d_5. Ideally, d_5 should be nearest to d_1 and d_2, and farthest from d_3 and d_4, as pointed earlier. Table 11.4 shows these calculations using the Euclidean distance metric. The assessment for which document is closest is made based on both the original 10 dimensions and the reduced set of two dimensions. Table 11.5 shows the same comparison using cosine similarity to compare documents. Cosine similarity is often used in text-mining tasks for document comparisons.

Euclidean distance as shown in Table 11.4 when using 10 dimensions ranks d_3 and d_4 above d_1 and d_2. By the formulation of Euclidean distance, when two documents both share a term or both do not share a term, the result is a distance of 0 for that dimension. After applying the LSA transformation, the Euclidean distance ranks d_1 above d_3 and d_4. However, document d_2 only is ranked above d_3 and not d_4.

Cosine similarity calculates the cosine of the angle between two vectors representing points in n-dimensional space. The result is a similarity score between 0 and 1,

TABLE 11.5. Use of Cosine Similarity to Find Most Similar Document to d_5 in Both 2-d and 10-d (the Largest Similarity Ranks First)

Comparison	*10-d (original)*		*2-d (LSA)*	
	Similarity	Rank	Similarity	Rank
d1–d5	0.289	4	0.999	1
d2–d5	0.500	1	0.826	2
d3–d5	0.316	2–3	0.317	4
d4–d5	0.316	2–3	0.603	3

where a 1 shows highest similarity, and 0 shows no similarity. For document vector comparisons, no additional strength of similarity is added by two vectors containing a 0 for a specific term. This is a very beneficial characteristic for textual applications. When we consider Table 11.5 we see that without the transformation, in 10-D space, d_2 is ranked above d_3 and d_4 for containing both terms in d_5. However, d_1 is ranked below d_3 and d_4 for having more terms than these other sentences. After the LSA transformation, d_1 and d_2 are ranked above d_3 and d_4, providing the expected ranking. In this simple example the best results occurred when we first transformed the data using LSA and then used cosine similarity to measure the similarity between initial training documents in a database and a new document for comparison. Using the nearest neighbor classifier, d_5 would be classified correctly as a "financial institution" document using cosine similarity for both the 10-d and 2-d cases or using Euclidean distance for the 2-d case. Euclidean distance in the 10-d case would have classified d_5 incorrectly. If we used k-nearest neighbor with $k = 3$, then Euclidean distance in the 10-d case would have also incorrectly classified d_5. Clearly, the LSA transformation affects the results of document comparisons, even in this very simple example. Results are better because LSA enable better representation of a document's semantics.

11.8 REVIEW QUESTIONS AND PROBLEMS

1. Give specific examples of where Web-content mining, Web-structure mining, and Web-usage mining would be valuable. Discuss the benefits.
2. Given a table of linked Web pages:

Page	Linked to page
A	B, D, E, F
B	C, D, E
C	B, E, F
D	A, F, E
E	B, C, F
F	A, B

(a) Find authorities using two iterations of the HITS algorithm.
(b) Find hubs using two iterations of the HITS algorithm.
(c) Find the PageRank scores for each page after one iteration using 0.1 as the dampening factor.
(d) Explain the HITS and PageRank authority rankings obtained in (a) and (c).

3. For the traversal log: {X, Y, Z, W, Y, A, B, C, D, Y, C, D, E, F, D, E, X, Y, A, B, M, N},

(a) find MFR;
(b) find LRS if the threshold value is 0.3 (or 30%);
(c) Find MRS.

4. Given the following text documents and assumed decomposition:

Document	Text
A	Web-content mining
B	Web-structure mining
C	Web-usage mining
D	Text mining

$$USV^T = \begin{matrix} -0.60 & 0.43 & 0.00 & 0.00 & 2.75 & 0.00 & 0.00 & 0.00 & -0.55 & -0.55 & -0.55 & -0.30 \\ -0.20 & 0.14 & 0.00 & 0.82 & 0.00 & 1.21 & 0.00 & 0.00 & 0.17 & 0.17 & 0.17 & -0.95 \\ -0.71 & -0.36 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 & -0.71 & 0.71 & 0.00 \\ -0.20 & 0.14 & -0.71 & -0.41 & 0.00 & 0.00 & 0.00 & 1.00 & 0.82 & -0.41 & -0.41 & 0.00 \\ -0.20 & 0.14 & 0.71 & -0.41 & & & & & & & & \\ -0.11 & -0.79 & 0.00 & 0.00 & & & & & & & & \end{matrix}$$

(a) create matrix A by using term counts from the original documents;
(b) obtain rank 1, 2, and 3 approximations to the document representations;
(c) calculate the variability preserved by rank 1, 2, and 3 approximations;
(d) Manually cluster documents A, B, C, and D into two clusters.

5. Given a table of linked Web pages and a dampening factor of 0.15:

Page	Linked to page
A	F
B	F
C	F
D	F
E	A, F
F	E

(a) find the PageRank scores for each page after one iteration;
(b) find the PageRank scores after 100 iterations, recording the absolute difference between scores per iteration (be sure to use some programming or scripting language to obtain these scores);

(c) explain the scores and rankings computed previously in parts (a) and (b). How quickly would you say that the scores converged? Explain.

6. Why is the text-refining task very important in a text-mining process? What are the results of text refining?

7. Implement the HITS algorithm and discover authorities and hubs if the input is the table of linked pages.

8. Implement the PageRank algorithm and discover central nodes in a table of linked pages.

9. Develop a software tool for discovering maximal reference sequences in a Web-log file.

10. Search the Web to find the basic characteristics of publicly available or commercial software tools for association-rule discovery. Document the results of your search.

11. Apply LSA to 20 Web pages of your choosing and compare the clusters obtained using the original term counts as attributes against the attributes derived using LSA. Comment on the successes and shortcomings of this approach.

12. What are the two main steps in mining *traversal patterns* using log data?

13. The XYZ Corporation maintains a set of five Web pages: {A, B, C, D, and E}. The following sessions (listed in timestamp order) have been created:

$$S_1 = [A, B, C], S_2 = [A, C], S_3 = [B, C, E], \text{and } S_4 = [A, C, D, C, E].$$

Suppose that support threshold is 30%. Find all *large sequences* (after building the tree).

14. Suppose a Web graph is undirected, that is, page i points to page j if and only page j points to page i. Are the following statements true or false? Justify your answers briefly.

(a) The hubbiness and authority vectors are identical, that is, for each page, its hubbiness is equal to its authority.

(b) The matrix M that we use to compute PageRank is symmetric, that is, M[i; j] = M[j; i] for all i and j.

11.9 REFERENCES FOR FURTHER STUDY

Akerkar, R., P. Lingras, *Building an Intelligent Web: Theory and Practice*, Jones and Bartlett Publishers, Sudbury, MA, 2008.

This provides a number of techniques used in Web mining. Code is provided along with illustrative examples showing how to perform Web-content mining, Web-structure mining and Web-usage mining.

Chang, G., M. J. Haeley, J. A. M. McHugh, J. T. L. Wang, *Mining the World Wide Web: An Information Search Approach*, Kluwer Academic Publishers, Boston, MA, 2001.

This book is an effort to bridge the gap between information search and data mining on the Web. The first part of the book focuses on IR on the Web. The ability to find relevant documents on the Web is essential to the process of Web mining. The cleaner the set of Web documents and data are, the better the knowledge that can be extracted from them. In the second part of the book, basic concepts and techniques on text mining, Web mining, and Web crawling are introduced. A case study, in the last part of the book, focuses on a search engine prototype called EnviroDaemon.

Garcia, E., *SVD and LSI Tutorial 4: Latent Semantic Indexing (LSI) How-to Calculations*, Mi Islita, 2006, http://www.miislita.com/information-retrieval-tutorial/svd-lsi-tutorial-4-lsi-how-to-calculations.html.

This Web tutorial provides students with a greater understanding of latent semantic indexing. It provides a detailed tutorial aimed at students. All calculations are pictured giving the student an opportunity to walk through the entire process.

Han, J., M. Kamber, *Data Mining: Concepts and Techniques*, 2nd edition, San Francisco, Morgan Kaufmann, 2006.

This book gives a sound understanding of data-mining principles. The primary orientation of the book is for database practitioners and professionals with emphasis on OLAP and data warehousing. In-depth analysis of association rules and clustering algorithms is the additional strength of the book. All algorithms are presented in easily understood pseudo-code, and they are suitable for use in real-world, large-scale data-mining projects, including advanced applications such as Web mining and text mining.

Mulvenna, M. D., et al., ed., Personalization on the Net Using Web Mining, *CACM*, Vol. 43, No. 8, 2000.

This is a collection of articles that explains state-of-the-art Web-mining techniques for developing personalization systems on the Internet. New methods are described for analyses of Web-log data in a user-centric manner, influencing Web-page content, Web-page design, and overall Web-site design.

Zhang, Q., R. S. Segall, Review of Data, Text and Web Mining Software, *Kybernetes*, Vol. 39, No. 4, 2010, pp. 625–655.

The paper reviews and compares selected software for data mining, text mining, and Web mining that are not available as free open-source software. The software for data mining are SAS® Enterprise Miner™, Megaputer PolyAnalyst® 5.0, NeuralWare Predict®, and BioDiscovery GeneSight®. The software for text mining are CompareSuite, SAS® Text Miner, TextAnalyst, VisualText, Megaputer PolyAnalyst® 5.0, and WordStat. The software for Web mining are Megaputer PolyAnalyst®, SPSS Clementine®, ClickTracks, and QL2. The paper discusses and compares the existing features, characteristics, and algorithms of selected software for data mining, text mining, and Web mining, respectively.