Data Mining and the Web: Past, Present and Future

Minos N. Garofalakis
Bell Laboratories
minos@bell-labs.com

Rajeev Rastogi
Bell Laboratories
rastogi@bell-labs.com

S. Seshadri
Bell Laboratories
seshadri@bell-labs.com

Kyuseok Shim
Bell Laboratories
shim@bell-labs.com

1 Introduction

The World Wide Web is rapidly emerging as an important medium for transacting commerce as well as for the dissemination of information related to a wide range of topics (e.g., business, government, recreation). According to most predictions, the majority of human information will be available on the Web in ten years. These huge amounts of data raise a grand challenge, namely, how to turn the Web into a more useful information utility.

Crawlers, search engines and Web directories like Yahoo! constitute the state-of-the-art tools for information retrieval on the Web today. Crawlers for the major search engines retrieve Web pages on which full-text indexes are constructed. A user query is simply a list of keywords (with some additional operators), and the query response is a list of pages ranked based on their similarity to the query.

Today's search tools, however, are plagued by the following four problems: (1) the *abundance problem*, that is, the phenomenon of hundreds of irrelevant documents being returned in response to a search query, (2) *limited coverage* of the Web, (3) a limited query interface that is based on syntactic keyword-oriented search, and (4) limited customization to individual users. These problems, in turn, can be attributed to the following characteristics of the Web. First and foremost, the Web is a huge, diverse and dynamic collection of interlinked hypertext documents. There are about 300 million pages on the Web today with about 1 million being added daily. Furthermore, it is widely believed that 99% of the information on the Web is of no interest to 99% of the people. Second, except for *hyperlinks*, the Web is largely unstructured. Finally, most information on the Web is in the form of HTML documents for which analysis and extraction of content is very difficult. Furthermore, the contents of many internet sources are hidden behind search interfaces and, thus, cannot be indexed – HTML documents are dynamically generated by these sources, in response to queries, using data stored in commercial DBMSs.

The question therefore is: how can we overcome these and other challenges that impede the Web resource discovery process? Fortunately, new and sophisticated techniques that have been developed in the area of *data mining* (also known as *knowledge discovery*), can aid in the extraction of useful information from the web. Data mining algorithms have been shown to scale well for large data sets and have been successfully applied to several areas like medical diagnosis, weather prediction, credit approval, customer segmentation, marketing and fraud detection.

In this paper, we begin by reviewing popular data mining techniques like association rules, classification, clustering and outlier detection. We provide a brief description of each technique as well as efficient algorithms for implementing the technique. We then discuss algorithms for discovering Web, hypertext and hyperlink structure, that have been proposed by researchers in recent years. The key difference between these algorithms and earlier data mining algorithms is that the latter take hyperlink information into account. Finally, we conclude by listing research issues that still remain to be addressed in the area of Web Mining.

2 Data Mining Techniques

In this section, we briefly describe key data mining algorithms that have been developed for large databases. A number of these algorithms are also applicable in the Web context and can be used to find related Web pages, as well as to cluster and categorize them.

2.1 Association Rules

Association rules, introduced in [1], provide a useful mechanism for discovering correlations among items belonging to customer transactions in a market basket database. An association rule has the form $X \rightarrow Y$, where X and Y are sets of items or *itemsets*. Let the *support* of an itemset X be the fraction of database transactions that contain X. The *support* of a rule of the form $X \rightarrow Y$ is then the same as the support of $X \cup Y$, while its *confidence* is the ratio of the supports of $X \cup Y$ and X. The association rules problem is that of computing all association rules that satisfy user-specified minimum support and minimum confidence constraints.

The *Apriori* algorithm [2] is the most popular algorithm for computing association rules. It first computes frequent itemsets (that is, itemsets that satisfy the minimum support constraint), and these are then used in a subsequent phase to compute rules that satisfy the minimum confidence constraint. Frequent itemsets are computed in passes - in the k^{th} pass, frequent itemsets of size k, F_k, are computed. F_k is used to generate C_{k+1}, the candidate itemsets for whom support is counted in the $k+1^{th}$pass. The key idea is that *every subset of a frequent itemset must also be frequent*, and so an itemset can be pruned from C_{k+1} if any of its k-subsets does not belong to F_k.

Several optimizations to the Apriori algorithm have been proposed in the literature [19, 22, 25]. In [19], the authors present an optimization in which when support for candidate itemsets in C_k is counted, the $k+1$ itemsets are hashed into buckets and a count of the number of itemsets that are hashed into each bucket is maintained. Later, when the candidate set C_{k+1} is generated in the next pass, if the hash bucket corresponding to a $k+1$-itemset does not have minimum support, then that itemset is pruned.

In [22], the authors propose a scheme in which the database is partitioned and frequent itemsets are computed for each partition. The rationale is that for an itemset to be frequent over the entire database, it must be frequent in at least one of the partitions. Consequently, once frequent itemsets for every partition have been computed, in a final pass, support for only these frequent itemsets needs to be computed using the entire database.

A sampling-based scheme for computing frequent itemsets is proposed in [25]. The idea is to first compute frequent itemsets using a random sample of the database. For the frequent itemsets computed F, the negative border is defined to be the itemsets that are not contained in F, but all of whose subsets are in F. A pass is made over the entire database to count support for itemsets in F and it's negative border. If any of the itemsets in the negative border turns out to be frequent, then a subsequent pass is made over the database and support for itemsets that could potentially be frequent is computed.

Recently, in [17], the authors proposed algorithms for mining frequent itemsets in the presence of *anti-monotone* and *succinct* constraints. A constraint C is said to be anti-monotone if for every itemset that satisfies C, every one of its subsets also satisfies C. Examples of anti-monotone constraints are $min(I) > v$ or $max(I) < v$, where I is an itemset. The authors note that during each pass of the Apriori algorithm, itemsets that do not satisfy anti-monotone constraints can be safely pruned. The authors also propose algorithms for counting support in the presence of *succinct* constraints, which are constraints that can be expressed as $I_1 \cup I_2 \cup \cdots$ for itemsets I_j.

In [6], the authors consider the problem of mining frequent sequences from a database of sequences in the presence of *regular expression* constraints. Since regular expression constraints are not anti-monotone, the authors show that there exists a trade-off between the pruning due to the regular expression constraint and support-based pruning. Specifically, the authors show that as the regular expression constraint is used to prune more candidates, fewer candidates get pruned due to the minimum support constraint.

Algorithms for mining frequent itemsets and sequences are useful for discovering Web structure and interesting access patterns. For instance, by teating each Web page as a transaction and the URLs it refers to as items, frequent itemsets result in groups of related URLs that are frequently referenced together. Similarly, frequent sequences in user Web log data yield information about user access patterns (that is, the sequence of URLs frequently traversed) that is of immense value to advertizers, Web site designers etc.

2.2 Classification

Classification is an important problem in data mining. It has been studied extensively by the machine learning community as a possible solution to the *knowledge acquisition* or *knowledge extraction* problem. The input to a classifier is a *training set* of records, each of which is tagged with a class label. A set of attribute values defines each record. The goal is to induce a model or description for each class in terms of the attributes. The model is then used to classify future records whose classes are unknown. Classifiers are useful in the Web context to build taxonomies and topic hierarchies on Web pages, and subsequently perform context-based searches for Web pages relating to a specific topic.

Decision trees classifiers are popular since they are easily interpreted by humans and are efficient to build. Most decision tree algorithms have two phases: a *building* phase followed by a *pruning* phase. In the building phase, the training data set is recursively partitioned until all the records in a partition have the same class. For every partition, a new node is added to the decision tree and it is labeled with the attribute and value (referred to as the *split value*) that was used to partition the set of tuples. In the pruning phase, nodes are iteratively pruned to prevent "overfitting" and to obtain a tree with higher accuracy. An important class of pruning algorithms are those based on the Minimum Decision Length (MDL) principle [15]. A subtree S is pruned if the cost of directly encoding the records in S is no more than the cost of encoding S plus the cost of the records in each leaf of S.

One of the first truly scaleable algorithms for classification was SPRINT [23]. In order to classify large datasets, SPRINT maintains separate attribute lists for each attribute and sorts the attribute lists for numeric attributes. The Rainforest algorithm proposed in [8] employs the notion of AVC-sets that maintain a count of tuples for distinct attribute value, class label pairs. These AVC-sets are used to determine how to partition sets of tuples without having to maintain separate attribute lists. The PUBLIC algorithm proposed in [20] integrates the building and pruning phases of decision tree construction in order to avoid expending effort on building portions of the tree that are any way going to be pruned. Finally, in BOAT [7], the authors use samples to construct b bootstrap decision trees which are used to compute confidence intervals for various split values for numeric attributes. A single pass is then performed over the database to determine the exact split value.

2.3 Clustering and Outlier Detection

Clustering, in data mining, is a useful technique for discovering interesting data distributions and patterns in the underlying data. The problem of clustering can be defined as follows: given n data points in a d-dimensional metric space, partition the data points into k clusters such that the data points within a cluster are more similar to each other than data points in different clusters.

Existing clustering algorithms can be broadly classified into *partitional* and *hierarchical* [11]. Partitional clustering algorithms attempt to determine k partitions that optimize a certain criterion function. In contrast, a hierarchical clustering is a sequence of partitions in which each partition is nested into the next partition in the sequence. An *agglomerative* algorithm for hierarchical clustering starts with the disjoint set of clusters, which places each input data point in an individual cluster. Pairs of items or clusters are then successively merged until the number of clusters reduces to k. At each step, the pair of clusters merged are the ones between which the distance is the minimum.

In [18], the authors propose a partitional clustering method for large databases which is based on randomized search. Each cluster is represented by its *medoid*, the most centrally located point in the cluster, and the objective is to find the k best medoids that optimize the criterion function. The authors reduce this problem to that of graph search by representing each set of k medoids as a node in the graph, two nodes being adjacent if they have $k-1$ medoids in common. Initially, an arbitrary node is set to be the current node and a fixed number of iterations are performed. In each iteration, a random neighbor of the current node is set to be the current node if it results in better clustering.

In [26], the authors present a clustering method named BIRCH whose I/O complexity is a little more than one scan of the data. BIRCH first pre-clusters the data into the maximum possible and finest possible subclusters that can fit in main-memory. For the pre-clustering phase, BIRCH employs a CF-tree to store cluster summaries, which is a balanced tree structure similar to an R-tree [21]. For each point, the CF-tree is traversed to find the closest cluster. If the cluster is within epsilon distance, the point is absorbed into the cluster. Otherwise, the point starts a new cluster.

A density-based algorithm called DBSCAN was proposed in [5]. DBSCAN requires the user to specify two parameters that are used to define the minimum density for clustering – the radius *Eps* of the neighborhood of a point and the minimum number of points *MinPts* in the neighborhood. Clusters are then found by starting from an arbitrary point and if its neighborhood satisfies the minimum density, including the points in its neighborhood into the cluster. The process is then repeated for the newly added points.

The CURE algorithm [10] chooses a random sample of the data, partitions it and partially clusters each partition. The partial clusters are then used to generate the final clusters in a subsequent pass. CURE employs a hierarchical clustering algorithm, but represents each cluster by a set of representative points distributed around the centroid. In [9], the authors propose ROCK, a clustering algorithm for categorical attributes. A pair of points whose similarity exceeds a certain threshold are referred to as neighbors. Robust clustering is then ensured by using the number of common neighbors between a pair of points as a measure of the distance between the points.

In [13], the authors introduce the notion of distance-based outliers. For a fraction p and a distance d, a point o is considered an outlier if p points lie at a greater distance than d from the point. The authors propose nested-loop and cell-based algorithms for computing distance based outliers.

3 Web Mining Techniques

In this section, we describe recent algorithms that exploit hyperlink information for discovering Web structure and Web communities. We also discuss recent work on finding the structure of hypertext documents and storing them.

3.1 Hubs and Authorities

In [12], to discover the underlying Web structure, the authors develop a notion of *hyperlinked communities* on the WWW through an analysis of the link topology. The key observation is that such communities typically contain two distinct, but inter-related, types of pages: *authorities* and *hubs*. Authorities are highly-referenced pages on the topic, while hubs are pages that "point" to many of the authorities, serving to cluster them together. Hubs and authorities thus exhibit a strong *mutually reinforcing relationship*: a good hub points to many good authorities; a good authority is pointed to by many good hubs.

Discovering the community for a specific topic/query thus involves computing the hubs and authorities for the topic, which is achieved as follows.

1. First a set S of seed pages on the topic are collected. Typically, these are the pages returned by a search engine on the query. This is then expanded to a larger base set by adding in any pages that point to, or are pointed to by, any page in S.
2. With each page p, a hub weight $h(p)$ and an authority weight $a(p)$, all initialized to 1, are associated. Let $p \rightarrow q$ denote "page p has a hyperlink to page q". The h's and a's for the pages are iteratively updated as follows: $a(p) = \sum_{q \rightarrow p} h(q)$, $h(p) = \sum_{p \rightarrow q} a(q)$.
3. After a fixed number of iterations, the 10 pages with the highest a values along with the 10 pages with the highest h values are considered to be the core of the community.

3.2 Building Web Knowledge Bases

The problem of creation of knowledge bases is addressed in [14] by enumerating and organizing all web occurrences of chosen subgraphs. Examples of subgraphs considered include cliques, webrings (which are star-shaped graphs with bidirectional links on the spokes) and bipartitie cores (which consists of two sets of nodes L and R, such that every node in L links to every node in R). The authors claim that a knowledge base of such structures constitutes a good starting point for deeper analyses and mining than raw web data, as well as for searching and navigation.

The authors propose an algorithm in the elimination/generation paradigm for efficiently enumerating the subgraphs of interest. For instance, consider the problem of enumerating cliques of size 4. In the elimination step, all nodes whose in-degree or out-degree is less than 4 are pruned. In the generation step, a list of nodes with in-degree or out-degree 3, along with their in- or out-neighborhoods is constructed. Following this, it is verified that each node in the neighborhood of a node points to all other nodes in its neighborhood.

In [3], the authors present algorithms for categorizing hypertext using hyperlinks. The authors show that a naive approach based on using a text classifier to determine the class for a web document using text from its neighbors performs badly. Instead, the authors propose a scheme that uses a bayesian classifier and exploits the knowledge of the classes of all the immediate neighbors of the document to be classified.

3.3 Mining the Structure of Web Documents

Web pages are instances of semistructured data, and thus mining their structure is critical to extracting information from them. The problem of extracting/mining schema from semistructured data is considered in [16]. The specific problem addressed is the following: given a data set I, find a typing τ and a data set J of typing τ such that the size of τ is smaller than a certain threshold and the distance between I and J is minimized. Thus, the goal is to find a τ that is small enough and such that I presents as few defects as possible with respect to τ. The authors first cast the typing problem in terms of a monadic datalog program, and then apply the greatest fixpoint semantics to the program to discover a perfect typing. The perfect types are then clustered (after defining a distance measure between pairs of types) using a greedy algorithm to compute k types such that the sum of the distances of each type to the closest (among the k types) is minimized. The greedy algorithm gives an $O(\log n)$ approximation of the optimal solution.

Recently, there has been a great deal of interest in XML which requires documents to store tags along with the data to convey semantic information. In [4, 24], schemes for storing XML documents in relational databases are proposed. The rationale is that by storing XML documents in relational databases 1) space costs associated with storing tags can be reduced, and 2) features of an RDBMS can be exploited. The approach in [4] attempts to mine the relational schema for a set of semistructured documents using a mining algorithm that computes frequent tree patterns in the data. The semistructured data model is then mapped to the relational model using a declarative query language, STORED. Parts of the semistructured data that do not fit the schema are stored in an "overflow" graph. Given the complete STORED mapping, queries and updates over the semistructured source are then rewritten into queries and updates on the relational source. In [24], the authors present several algorithms for constructing relational schemas from DTDs. They also develop algorithms that convert XML documents to relational tuples, translate semistructured queries over XML documents to SQL queries over tables and convert the results to XML.

4 Web Mining Research Issues

The approaches described in the previous section represent initial attempts at mining hyperlink and hypertext structure. However, to improve information retrieval and the quality of searches on the Web, a number of research issues still need to be addressed, which we list below.

Mining Web Structure. The work from [12, 14, 3], presented in the previous section, represent some of the pioneering efforts in mining Web structure. These approaches only take into account hyperlink information and pay little or no attention to the content of Web pages. It would be interesting to devise approaches that take into account both hyperlink information as well as Web page content.

Topic hierarchies like the one provided by Yahoo! give a hierarchical classification of documents. Searches in the context of a specific topic help to eliminate clutter by focusing the search to only documents pertaining to the topic. Unfortunately, since these hierarchies are manually generated, they cover only a small portion of the Web. The challenge here is to automate the classification and clustering of millions of dynamically changing Web documents with diverse authorship.

Improving Customization. Customization involves learning about an individual user's preferences/interests based on access patterns or alternately, based on explicit directives from the user. Thus, customization aids in providing users with pages, sites and advertizements that are of interest to them. It may also be possible for sites to automatically optimize their design and organization based on observed user patterns.

Extracting Information from Hypertext Documents. Information extraction from web resources is complicated since HTML annotations provide very little semantic information. Furthermore, a number of sources hide information behind search interfaces. Thus, the majority of today's information-extraction systems rely on "hand coded" wrappers to access a fixed set of Web resources. Obviously, such a manual approach cannot scale, and new techniques for automating the extraction process from unfamiliar documents need to be devised. It is widely believed that HTML will be replaced by XML, forcing documents to become self-describing through the specification of tag sets (referred to as the Document Type Definitions, or DTDs). Thus, the contents of each XML document can be extracted by consulting the DTDs to which the document conforms. Web sites will also be able to describe their query capabilities through XML – thus enabling structured queries like "find the cheapest airline ticket from New York to Chicago" or "list all jobs with salary $>$ 50K in the Boston area". Thus, with XML, it may be possible to transform the entire Web into one unified database.

More research, however, is still needed in the areas of storage of XML documents, and integration and querying of XML documents originating from different sources. Other interesting research problems include extracting DTD information given a set of XML/semistructured documents, and merging DTD information from several XML sources.

References

[1] Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. Mining association rules between sets of items in large databases. In *Proc. of the ACM SIGMOD Conference on Management of Data*, pages 207–216, Washington, D.C., May 1993.

[2] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules. In *Proc. of the VLDB Conference*, Santiago, Chile, September 1994.

[3] S. Chakrabarti, B. Dom, and P. Indyk. Enhanced hypertext categorization using hyperlinks. In *Proc. of the ACM SIGMOD Conference on Management of Data*, June 1998.

[4] A. Deutsch, M. Fernandez, and D. Suciu. Storing semistructured data with stored. In *Proc. of the ACM SIGMOD Conference on Management of Data*, June 1999.

[5] Martin Ester, Hans-Peter Kriegel, Jorg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial database with noise. In *Int'l Conference on Knowledge Discovery in Databases and Data Mining (KDD-96)*, Montreal, Canada, August 1996.

[6] M. Garofalakis, R. Rastogi, and K. Shim. Spirit:sequential pattern mining with regular expression constraints. In *Proc. of the VLDB Conference*, Edinburgh, Scotland, September 1999.

[7] J. Gehrke, V. Ganti, R. Ramakrishnan, and W. Loh. Boat–optimistic decision tree construction. In *Proc. of the ACM SIGMOD Conference on Management of Data*, June 1999.

[8] Johannes Gehrke, Raghu Ramakrishnan, and Venkatesh Ganti. Rainforest - a framework for fast decision tree construction of large datasets. In *Proc. of the Int'l Conf. on Very Large Data Bases*, New York, 1998.

[9] S. Guha, R. Rastogi, and K. Shim. Rock: A robust clustering algorithm for categorical attributes. In *Int'l Conference on Data Engineering*, Sydney, Australia, 1999.

[10] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. "CURE: An Efficient Clustering Algorithm for Large Databases". In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, Seattle, Washington, June 1998.

[11] Anil K. Jain and Richard C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, Englewood Cliffs, New Jersey, 1988.

[12] J. Kleinberg. Authoritative sources in s hyperlinked environment. In *Proc. of ACM-SIAM Symposium on Discrete Algorithms*, 1998.

[13] E. Knorr and R. Ng. Algorithms for mining distance-based outliers in large datasets. In *Proc. of the VLDB Conference*, pages 392–403, New York, USA, September 1994.

[14] R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Extracting large-scale knowledge bases from the web. In *Proc. of the Int'l Conf. on Very Large Data Bases*, Edinburgh, Scotland, 1999.

[15] Manish Mehta, Jorma Rissanen, and Rakesh Agrawal. MDL-based decision tree pruning. In *Int'l Conference on Knowledge Discovery in Databases and Data Mining (KDD-95)*, Montreal, Canada, August 1995.

[16] S. Nestorov, S. Abiteboul, and R. Motwani. Extracting schema from semistructured data. In *Proc. of the ACM SIGMOD Conference on Management of Data*, June 1998.

[17] R. T. Ng, L. V. S. Lakshmanan, J. Han, and A. Pang. Exploratory mining and pruning optimizations of constrained association queries. In *Proc. of the ACM SIGMOD Conference on Management of Data*, June 1998.

[18] Raymond T. Ng and Jiawei Han. Efficient and effective clustering methods for spatial data mining. In *Proc. of the VLDB Conference*, Santiago, Chile, September 1994.

[19] Jong Soo Park, Ming-Syan Chen, and Philip S. Yu. An effective hash based algorithm for mining association rules. In *Proc. of the ACM-SIGMOD Conference on Management of Data*, San Jose, California, May 1995.

[20] Rajeev Rastogi and Kyuseok Shim. Public: A decision tree classifier that integrates building and pruning. In *Proc. of the Int'l Conf. on Very Large Data Bases*, New York, 1998.

[21] H. Samet. *The Design and Analysis of Spatial Data Structures*. Addison-Wesley, 1989.

[22] A. Savasere, E. Omiecinski, and S. Navathe. An efficient algorithm for mining association rules in large databases. In *Proc. of the VLDB Conference*, Zurich, Switzerland, September 1995.

[23] John Shafer, Rakesh Agrawal, and Manish Mehta. SPRINT: A scalable parallel classifier for data mining. In *Proc. of the VLDB Conference*, Bombay, India, September 1996.

[24] J. Shanmugasundaram, G. He, K. Tufte, C. Zhang, D. DeWitt, and J. Naughton. Relational databases for querying xml documents: Limitations and opportunities. In *Proc. of the Int'l Conf. on Very Large Data Bases*, Edinburgh, Scotland, 1999.

[25] Hannu Toivonen. Sampling large databases for association rules. In *Proc. of the VLDB Conference*, Bombay, India, September 1996.

[26] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. Birch: An efficient data clustering method for very large databases. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 103–114, Montreal, Canada, June 1996.