

A RESEARCH SUPPORT SYSTEM FRAMEWORK FOR WEB DATA MINING

Jin Xu Yingping Huang Gregory Madey

Dept. of Computer Science
University of Notre Dame
Notre Dame, IN 46556

Email: {jxu1, yhuang3, gmadey}@cse.nd.edu

ABSTRACT

Design and implementation of a research support system for web data mining has become a challenge for researchers wishing to utilize useful information on the web. This paper proposes a framework for web data mining support systems. These systems are designed for identifying, extracting, filtering and analyzing data from web resources. They combine web retrieval and data mining techniques together to provide an efficient infrastructure to support web data mining for research.

Keywords: Data mining, web retrieval, clustering, classification, association rules, research support system

1. INTRODUCTION

The evolution of the World Wide Web has brought us enormous and ever growing amounts of data and information. With the abundant data provided by the web, it has become an important resource for research. However, traditional data extraction and mining techniques can not be applied directly to the web due to its semi-structured or even unstructured nature. Web pages are Hypertext documents, which contain both text and hyperlinks to other documents. Furthermore, web data are heterogeneous and dynamic. Thus, design and implementation of a web data mining research support system has become a challenge for researchers in order to utilize useful information from the web.

Usually, web mining is categorized as web content mining and web usage mining. Web content mining studies the search and retrieval of information on the web, while web usage mining discovers and analyzes user access pattern [1]. A knowledge discovery tool, WebLogMiner, is discussed in [8] which uses OLAP and data mining techniques for mining web server log files. In [5], a web mining framework which integrated both usage and content attributes of a site is described. Specific techniques based on clustering and association rules are proposed. Yao [3, 6, 7] presents a framework and information retrieval techniques to support individual scientists doing research. This paper proposes a framework for designing web data mining research support

systems. These systems are designed for identifying, extracting, filtering and analyzing data from web resources. They combine web retrieval and data mining techniques together to provide an efficient infrastructure to support web data mining for research. This framework is composed of several stages. Features of each stage are explored and implementation techniques are presented. A case study on mining data from a large software development site is provided as an example of how to use this framework. Our work provides a general solution which researchers can follow to utilize web resources in their research.

The rest of this paper is organized as follows: Section 2 presents a framework for web mining research support systems and a brief overview of its components. Section 3 describes design and implementation of web data retrieval. Section 4 focuses on processing and analyzing web data. Section 5 presents a case study based on this web mining research support system. Conclusions and future work are given in Section 6.

2. FRAMEWORK OVERVIEW

In order to explore web data, we construct a research support system framework for web data mining, as shown in Fig 1, consisting of four phases: source identification, content selection, information retrieval and data mining.

In the first phase, proper web sites should be chosen according to research needs. This includes identifying availability, relevance and importance of web sites. Key words searching by using search engine can be used to find appropriate web sites.

After finding all web sites identified by the first phase, the second phase is to select appropriate contents on those web sites, such as documentation, newsgroups, forums, mailing lists, etc. Usually, a web site contains many web pages, including relevant and irrelevant information. This phase is important because it decides which web information should be extracted. The selection of web pages is based on research purpose and a researcher's experience.

In the information retrieval phase, a crawler is designed

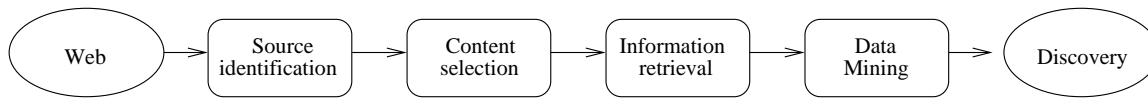


Fig. 1. Research support system framework for web data mining

to automatically extract information selected during the selection phase. Specific tools and techniques are employed to effectively retrieve useful knowledge/information from web sources. Additional effort may be required for dynamic content retrieval and specific data sources such as newsgroup, forum, etc.

The final phase is to conduct data mining on extracted web data. It includes preparing data for analysis. An extracted web page may contain missing data, extraneous data, wrong format and unnecessary characters. Furthermore, some data should be processed in order to protect privacy. Advanced data mining techniques are employed here to help analyzing data.

3. INFORMATION RETRIEVAL

Information retrieval is used to provide access to data on the web. This implies a web mining research support system should be able to search for and retrieve specific contents on the web efficiently and effectively. There are two major categories of searching tools on the Web: directories (Yahoo, Netscape, etc.) and search engines (Lycos, Google, etc.). It is hard to use directories with the increase of web sites. Search engines cannot meet every search requirement

In our system, a web crawler based on advanced tools and techniques is developed to help find useful information from web resources. Web crawlers are also called spiders, robots, worms, etc. A web crawler is a program which automatically traverses web sites, downloads documents and follows links to other pages [4]. It keeps a copy of all visited pages for later uses. Many web search engines use web crawlers to create entries for indexing. They can also be used in other possible applications such as page validation, structural analysis and visualization, update notification, mirroring and personal web assistants/agents etc. [2]. Search engines are not adequate for web mining for a research project. It is necessary to design a web crawler which includes methods to find and gather the research related information from the web. Although different research projects have different web information which leads to different web crawlers, those crawlers still have some common designs, as shown in Figure 2. They can be implemented by Java, Perl, Python, etc.

A web crawler should start with a URL, identify other links on the HTML page, visit those links, extract information from web pages and store information into databases. Thus, a web crawler consists of a URL access method, a web page parser with some extractors, and databases. The

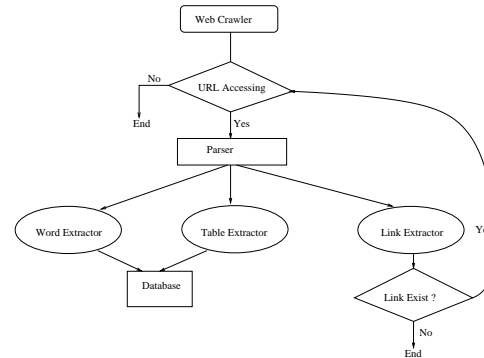


Fig. 2. The web crawler

access function of a web crawler should prevent repeatedly accessing the same web address and should identify dead links. The parser recognizes start tags, end tags, text and comments. The databases provide storage for extracted web information.

The key component of a web crawler is the parser, which includes a word extractor, a table extractor and a link extractor. The word extractor is used to extract word information. It should provide string checking functions. Tables are used commonly in web pages to align information. A table extractor identifies the location of the data in a table. A link extractor retrieves links contained in a web page. There are two types of links – absolute links and relative links. An absolute link gives the full address of a web page, while a relative link needs to be converted to a full address by adding a prefix.

4. DATA MINING TECHNOLOGY

To facilitate web mining, the following data mining algorithms can be applied to find patterns and trends in the data collected from the web: clustering, classification, association rules. In the following sections, we will introduce each of the algorithms and explain their applications.

4.1. Association Rules

Association rules mining tries to find interesting association or correlation relationship among a large set of data items. A typical example of association rules mining is the market basket analysis. An association rule is something like "80% of people who buy beer also buy fried chicken".

Association rules mining can also be applied to predict web access patterns for personalization. For example, we

may discover that 80% of people who access page A and page B also access page C. Page C might not have a direct link from either page A or page B. The information discovered might be used to create a link to page C from page A or page B. One example of this application is amazon.com. We often see something like "customers who buy this book also buy book A". The association rules mining can be applied to web data to explore the behavior of web users and find patterns of their behaviors.

4.2. Classification

The goal of classification is to predict which of several classes a case (or an observation) belongs to. Each case consists of n attributes, one of which is the target attribute, all others are predictor attributes. Each of the target attribute's value is a class to be predicted based on the $n - 1$ predictor attributes.

Classification is a two-step process. First, a classification model is built based on training data set. Second, the model is applied to new data for classification. In the middle of the two steps, some other steps might be taken, such as lift computation. Lift computation is a way of verifying whether a classification model is valuable. A value larger than 1 is normally good.

Classification models can be applied on the web to make business decisions. Applications include classifying email messages as junk mails, detecting credit card fraud, network intrusion detection, etc.

4.3. Clustering

Clustering is used to find natural groupings of data. These natural groupings are clusters. A cluster is a collection of data that are similar to one another. A good clustering algorithm produces clusters such that inter-cluster similarity is low and intra-cluster similarity is high. Clustering can be used to group customers with similar behavior and to make business decisions in industry.

5. CASE STUDY: OPEN SOURCE SOFTWARE (OSS) DEVELOPMENT

The OSS community has developed a substantial amount of the infrastructure of the Internet, and has several outstanding technical achievements, including Apache, Perl, Linux, etc. These programs were written, developed, and debugged largely by part time contributors, who in most cases were not paid for their work, and without the benefit of any traditional project management techniques. A research study of how the OSS community functions may help IT planners make more informed decisions and develop more effective strategies for using OSS software.

The OSS development community is a global virtual community. Thus, we have the advantage in that their digital interactions are archived and can be data mined. With approximately 70,000 projects, 90,000 developers, and 700,000 registered users, SourceForge.net, sponsored by VA Software is the largest OSS development and collaboration site. This site provides highly detailed information about the projects and the developers, including project characteristics, most active projects, and "top ranked" developers.

5.1. Data Collection

After informing SourceForge of our plans and receiving permission, we gathered data monthly at SourceForge. SourceForge provides project management tools, bug tracking, mail list services, discussion forums, version control software for hosted projects. Data is collected at the community, project, and developer level, characterizing the entire OSS phenomenon, across multiple numbers of projects, investigating behaviors and mechanisms at work at the project and developer levels.

The primary data required for this research are two tables – project statistics and developers. The project statistics table, shown in Figure 1, consists of records with 9 fields: project ID, lifespan, rank, page views, downloads, bugs, support, patches and CVS. The developers table has 2 fields: project ID and developer ID. Because projects can have many developers and developers can be on many projects, neither field is unique primary key. Thus the composite key composed of both attributes serves as a primary key. Each project in SourceForge has a unique ID when registering with SourceForge.

A web crawler, implemented by Perl and CPAN (Comprehensive Perl Archive - the repository of Perl module/libraries) modules, traversed the SourceForge web server to collect the necessary data. All project home pages in SourceForge have a similar top-level design. Many of these pages are dynamically generated from a database. The web crawler uses LWP, the libwww-Perl library, to fetch each project's homepage. CPAN has a generic HTML parser to recognize start tags, end tags, text and comments, etc. Because both statistical and member information are stored in tables, the web crawler uses an existing Perl Module called *HTML::TableExtract* and string comparisons provided by Perl to extract information. Link extractors are used if there are more than one page of members.

5.2. Data Mining

There are several data mining algorithms that can be applied to the web data. Among them are Naive Bayes, Classification and Regression Tree (CART), for classification, A Priori for association rules mining, K-means and Orthogonal-Cluster for clustering. Let's first focus on the classification

Table 1. Project statistics

project ID	lifespan	rank	page views	downloads	bugs	support	patches	all trackers	tasks	cvs
1	1355 days	31	12,163,712	71,478	4,160	46,811	277	52,732	44	0
2	1355 days	226	4,399,238	662,961	0	53	0	62	0	14,979
3	1355 days	301	1,656,020	1,064,236	364	35	15	421	0	12,263
7	1355 days	3322	50,257	20,091	0	0	0	12	0	0
8	1355 days	2849	6,541,480	0	17	1	1	26	0	13,896

algorithms: Naive Bayes and CART.

5.2.1. Classification

The Naive Bayes algorithms makes prediction using Bayes' Theorem. Naive Bayes assumes that each attribute is independent from others. In this case study, that is not the case. For example, the "downloads" feature is closely related to the "cvs" feature, the "rank" feature is closely related to other features, since it is calculated from other features.

CART builds classification and regression trees for predicting continuous dependent variables (regression) and categorical predictor variables (classification). We are only interested in the classification type of problems since the software we are using only handles this type of problems. Oracle's implementation of CART is called Adaptive Bayes Network (ABN). ABN predicts binary and multi-class targets. Thus discretizing the target attribute is desirable.

In this case study, we try to predict downloads from other features. As stated previously, the "downloads" feature is binned into ten equal buckets. We predict the downloads resides which buckets based on the values of other features. As expected, the Naive Bayes algorithms is not suitable for predicting "downloads", since it is related to other features, such as "cvs". The accuracy of Naive Bayes is less than 10%. While Naive Bayes performs badly on predicting "downloads", the ABN algorithms can predict "downloads" quite accurately which is about 63%. At first sight, the accuracy 63% is not attractive, but it is a good prediction since we could only get 10% of correct predictions without classification. The lift computation confirms that the resulting classification model is quite good, as shown in Figure 3. This figure shows that we found all the records whose "downloads" feature is 1 in just the first 20% of records. The rules built by the ABN classification model show that "downloads" is closely related to "cvs".

We conclude that the ABN algorithms is suitable for predicting "downloads" in our case study. The following table compares the two algorithms, namely, ABN and Naive Bayes. From the table, we see that ABN takes much longer time to build a classification model, but the resulting model is much more accurate.

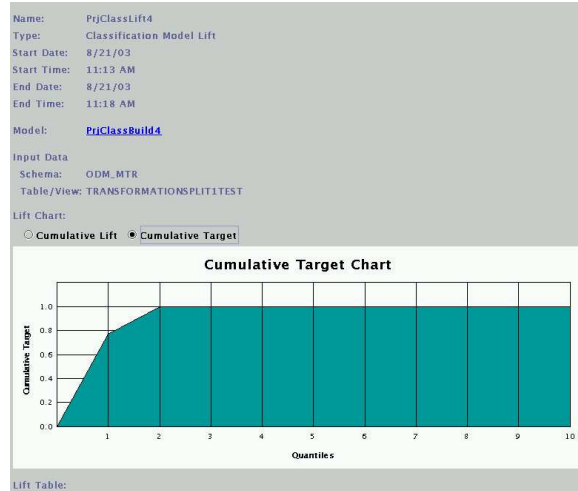


Fig. 3. The lift chart

Table 2. Comparison of ABN and Naive Bayes

Name	Build Time	Accuracy
ABN	0:19:56	63%
Naive Bayes	0:0:30	9%

5.2.2. Association Rules

The association rules mining problem can be decomposed into two subproblems:

- Find all combinations of items, called frequent itemsets, whose support is greater than the minimum support.
- Use the frequent itemsets to generate the association rules. For example, if AB and A are frequent itemsets, then the rule $A \rightarrow B$ holds if the ratio of support(AB) to support(A) is greater than the minimum confidence.

One of the most famous association rules mining algorithms is called A Priori. Oracle implements this algorithm using SQL. We use the algorithm to try to find correlations between features of projects. The algorithm takes two inputs, namely, the minimum support and the minimum confidence. We choose 0.01 for minimum support and 0.5 for

Cluster ID	Cases	Split Rule
1	50000	SUPPORT in (4)
3	23828	LIFESPAN in (4)
6	8534	SUPPORT in (6)
18	3590	n/a
19	4944	n/a
7	15294	PAGE_VIEWS in (7)
8	10449	PAGE_VIEWS in (4)
14	5412	n/a
15	5037	n/a
9	4845	n/a
2	26172	ALL_TRKS in (4)
4	10993	SUPPORT equal (1)
12	3035	n/a
13	7958	n/a
5	15179	BUGS in (6)

Fig. 4. The clusters

IF (condition)	Then (cluster)	Confidence	Support
ALL_TRKS in (10, 3, 4, 5, 8, 9) and BUGS in (1, 2, 3, 4, 5, 6, 7, 8, 9) and SUPPORT in (10, 5, 6, 7, 8, 9) and PAGE_VIEWS in (10, 8, 9) and RANK in (2, 3, 4, 5, 6, 7, 9) and SUPPORT in (10, 5, 6, 7, 8) and TASKS in (1, 10, 2, 3, 8, 9)	CLUSTER equal (9)	1.0	1.0
ALL_TRKS in (10, 5, 6, 8, 9) and BUGS in (1, 2, 3, 4, 5, 6, 7, 8, 9) and SUPPORT in (10, 5, 6, 7, 8, 9) and PAGE_VIEWS in (10, 8, 9) and RANK in (2, 3, 4, 5, 6, 7, 9) and SUPPORT in (10, 5, 6, 7, 8) and TASKS in (1, 10, 2, 3, 8, 9)	CLUSTER equal (11)	1.0	1.0
ALL_TRKS in (1, 2, 3) and BUGS in (2, 3, 4, 5, 6, 7, 8, 9) and SUPPORT in (10, 5, 6, 7, 8, 9) and PAGE_VIEWS in (10, 8, 9) and RANK in (2, 3, 4, 5, 6, 7, 9) and SUPPORT in (10, 5, 6, 7, 8) and TASKS in (1, 10, 2, 3, 8, 9)	CLUSTER equal (12)	1.0	1.0
ALL_TRKS in (1, 2, 3, 4) and BUGS in (1, 2, 3, 4, 5, 6, 7, 8, 9) and SUPPORT in (10, 5, 6, 7, 8, 9) and PAGE_VIEWS in (10, 8, 9) and RANK in (2, 3, 4, 5, 6, 7, 9) and SUPPORT in (10, 5, 6, 7, 8) and TASKS in (1, 10, 2, 3, 8, 9)	CLUSTER equal (13)	1.0	1.0
ALL_TRKS in (3, 4, 5, 6, 8, 9) and BUGS in (1, 2, 3, 4, 5, 6, 7, 8, 9) and SUPPORT in (10, 5, 6, 7, 8, 9) and PAGE_VIEWS in (10, 8, 9) and RANK in (2, 3, 4, 5, 6, 7, 9) and SUPPORT in (10, 5, 6, 7, 8) and TASKS in (1, 10, 2, 3, 8, 9)	CLUSTER equal (14)	1.0	1.0
ALL_TRKS in (3, 4, 5, 8, 9) and BUGS in (1, 2, 3, 4, 5, 6, 7, 8, 9) and SUPPORT in (10, 5, 6, 7, 8, 9) and PAGE_VIEWS in (10, 8, 9) and RANK in (2, 3, 4, 5, 6, 7, 9) and SUPPORT in (10, 5, 6, 7, 8) and TASKS in (1, 10, 2, 3, 8, 9)	CLUSTER equal (15)	1.0	1.0

Fig. 5. The rules that define clusters

minimum confidence. we find that the feature “all trks”, “cvs” and “downloads” are “associated”. More rules can be seen from Figure 5. Not all of the rules discovered are of interest.

5.2.3. Clustering

We are interested in putting the projects with similar features together to form clusters. Two algorithms can be used to accomplish this: k-means and o-cluster. The k-means algorithm is a distance-based clustering algorithm, which partitions the data into predefined number of clusters. The o-cluster algorithm is a hierarchical and grid-based algorithm. The resulting clusters define dense areas in the attribute space. The dimension of the attribute space is the number of attributes involved in the clustering algorithm.

We apply the two clustering algorithms to projects in this case study. Figure 4 and Figure 5 shows the resulting clusters and the rules that define the clusters.

6. CONCLUSION AND FUTURE WORK

This paper discusses a framework for web mining research support system and describes its procedures. It then discusses implementing techniques on web data extraction and analysis. A sourceforge web mining case is presented as an example of how to apply this framework.

This work is an exploratory study of web data retrieval and data mining on web data. We try to evaluate the data extraction process and data mining software which can be used to discover knowledge in the web data. The actual interesting discoveries are still in progress. We are expected to discover interesting patterns from the data.

This research was partially supported by the US National Science Foundation, CISE/IIS-Digital Science and Technology, under Grant No. 0222829.

7. REFERENCES

- [1] R. Cooley, B. Mobasher, and J. Srivastava. Web mining: Information and pattern discovery on the world wide web. In *International Conference on Tools with Artificial Intelligence*, pages 558–567, Newport Beach, 1997.
- [2] Francis Crimmins. Web crawler review. <http://dev.funnelback.com/crawler-review.html>, 2001.
- [3] Yao J.T. and Yao Y.Y. Web-based information retrieval support systems: building research tools for scientists in the new information age. In *Proceedings of the IEEE/WIC International Conference on Web Intelligence*, Halifax, Canada, 2003.
- [4] M. Koster. The web robots pages. <http://info.webcrawler.com/mak/projects/robots/robots.html>, 1999.
- [5] Bamshad Mobasher, Honghua Dai, Tao Luo, Yuqing Sun, and Jiang Zhu. Integrating web usage and content mining for more effective personalization. In *EC-Web*, pages 165–176, 2000.
- [6] Yao Y.Y. Information retrieval support systems. In *FUZZ-IEEE'02 in The 2002 IEEE World Congress on Computational Intelligence*, Honolulu, Hawaii, USA, 2002.
- [7] Yao Y.Y. A framework for web-based research support systems. In *Computer Software and Application Conference, (COMPOSAC 2003)*, Dallas, Texas, 2003.
- [8] Osmar R. Zaïane, Man Xin, and Jiawei Han. Discovering web access patterns and trends by applying OLAP and data mining technology on web logs. In *Advances in Digital Libraries*, pages 19–29, 1998.