Applied Computational Intelligence and Soft Computing
Volume 2012 (2012), Article ID 593147, 7 pages
http://dx.doi.org/10.1155/2012/593147

Research Article

An Application of Improved Gap-BIDE Algorithm for Discovering Access Patterns

Xiuming Yu,[1] Meijing Li,[1] Taewook Kim,[1] Seon-phil Jeong,[2] and Keun Ho Ryu[1,2,3]

[1]Database and Bioinformatics Laboratory, Chungbuk National University, Cheongju 361-763, Republic of Korea
[2]Division of Science and Technology, BNU-HKBU United International College, Zhuhai 519-085, China
[3]Multimedia Systems Laboratory, School of Computer Science and Engineering, The University of Aizu, Aizu-Wakamatsu, Fukushima 965-8580, Japan

Received 9 March 2012; Revised 14 May 2012; Accepted 20 May 2012

Academic Editor: Qiangfu Zhao

Abstract

Discovering access patterns from web log data is a typical sequential pattern mining application, and a lot of access pattern mining algorithms have been proposed. In this paper, we propose an improved approach of Gap-BIDE algorithm to extract user access patterns from web log data. Compared with the previous Gap-BIDE algorithm, a process of getting a large event set is proposed in the provided algorithm; the proposed approach can find out the frequent events by discarding the infrequent events which do not occur continuously in an accessing time before generating candidate patterns. In the experiment, we compare the previous access pattern mining algorithm with the proposed one, which shows that our approach is very efficient in discovering access patterns in large database.

1. Introduction

The web has become an important channel for conducting business transactions and e-commerce. Also, it provides a convenient means for us to communicate with each other worldwide. With the rapid development of web technology, the web has become an important and preferred platform for distributing and acquiring information. The data collected automatically by the web and application web servers represent the navigational behavior of web users, and such data is called web log data.

Web mining is a technology to discover and extract useful information from web log data. Because of the tremendous growth of information sources, increasing interest of various research communities, and the recent interest in e-commerce, the area of web mining has become vast and more interesting. It deals with data related to the web, such as data hidden in web contents, data presented on web pages, and data stored on web servers. Based on the kinds of data, there are three categories of web mining: web content mining, web structure mining, and web usage mining [1]. The Web usage data includes the data from web server access logs, proxy server logs, and browser logs. It is also known as web access patterns. Web usage mining tries to discover the access patterns from web log files. Web access tracking can be defined as web page history [2]; the mining task is a process of extracting interesting patterns in web access logs. There are so many techniques of mining web usage data including statistical analysis [3], association rules [4], sequential patterns [5–7], classification [8–10], and clustering [11–13]. Access pattern mining is a popular approach of sequential pattern mining, which extracts frequent subsequences from a sequence database [14]. Further, discovering access patterns is an important challenge in the field of web mining. And the popular applications of access patterns mining are obtaining useful information of web users' behavior.

A lot of studies have been proposed on access pattern mining for finding valuable knowledge from web log data, such as AprioriAll algorithm [15, 16] and GSP (generalized sequential pattern) algorithm [17]. All of above algorithms mine sequential patterns using a paradigm of candidate generate-and-test maintain a candidate set of already mined patterns in the mining process. When the data set is huge, it will generates a lot of candidate patterns. In other words, GSP algorithm needs much memory while the data set is large. The BIDE algorithms [18] mine frequent patterns without keeping the candidate pattern sets, therefore it needs less space during the mining task. And above algorithms focus on finding out the patterns which are adjacent and that may miss some hidden relationships among noncontinuous patterns. So the constraint of gap should be considered. In the paper [19], the author proposed an improved BIDE algorithm (Gap-BIDE) for mining closed sequential patterns with gap constraint and considers the patterns that are not only adjacent but also noncontiguous; Gap-BIDE algorithm had been applied to web mining in [20]. And in the previous work [21], we have improved the Gap-BIDE algorithm by discarding infrequent events before generating frequent candidate events and applying the improved algorithm to access pattern mining and discussed the efficient of parameter of the values of gap. In this paper, we perform the improved algorithm and compare the efficiency with previous access pattern mining algorithms, such as GSP algorithm.

The rest of this paper is organized as follows. Section 2 presents the precedent of our algorithm compared with the original algorithm. Section 3 focuses on discovering access patterns, namely, preprocessing, pattern discovery, and result analysis, and it focuses on the efficiency of the proposed approach in terms of access pattern mining. In Section 4, we present an extensive performance study. Finally, we conclude this study in

Section 5.

2. Algorithm of Improved Gap-BIDE

2.1. Gap-BIDE Algorithm

Gap-BIDE algorithm is presented in paper [19], and it inherits the same design philosophy as BIDE algorithm. It shares the same merit, that is, it does not need to maintain a candidate pattern set, which saves space consumption, and it can find some hidden relationships among the patterns that contend for the gap constraint.

The algorithm first finds the set of all frequent patterns, and it then mines the gap-constrained closed sequential patterns with pattern P as the prefix. In this process, it first scans the backward spaces of prefix pattern P, uses the gap-constrained backscan pruning method to prune search space, scans the forward spaces of prefix P, and uses the gap-constrained pattern closure checking scheme to check whether or not pattern P is closed; finally, it scans each forward space of all appearances of pattern P and finds the set of all locally frequent items, L, uses each item in L to extend P, and mines the gap-constrained closed sequential patterns for the new prefix by calling subroutine again.

In the algorithm, forward space is defined as that given an appearance of pattern $P[M, N]$ with triple (sid, beginPos, and endPos). The forward space of appearance is part of the sequence of range [endPos + M, endPos + N] ∩ [endPos, l), where l is the length of sequence sid. Here, the definition of forward space (FS) is induced for getting frequent subsequence patterns. We can get the sequence support of every subsequence by scanning the forward spaces of the appearances of a prefix pattern. The sequences whose supports are greater than or equal to the minimal support threshold Minsup will be the frequent subsequences patterns of a prefix pattern.

The definition of backward space (BS) is important, and it is defined as that given an appearance of pattern $P[M, N]$ with triple (sid, beginPos, and end-Pos). The backward space of appearance is part of the sequence sid that is of the range [beginPos $-N$, beginPos $-M$] ∩ [0, beginPos).

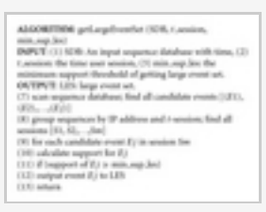
Performance of proposed approach shows that Gap-BIDE is both runtime and space efficient in mining frequent, closed sequences with gap constraints.

2.2. Improved Gap-BIDE Algorithm

Although Gap-BIDE algorithm is advanced in the algorithms of sequential pattern mining, there are still a lot of fool's errands are done during the mining task, such as generating some candidate patterns for infrequent events in the original data set. To avoid the unnecessary memory use, an improved algorithm is proposed. Our algorithm is designed based on the Gap-BIDE algorithm; the main idea is to discard infrequent events before generating frequent candidate events; we call this process as getting a large event set.

Algorithm 1 is the main algorithm. The Algorithm 2 is a subroutine of Algorithm 1; it proposes the process of getting a large event set. A large event set (LES) is an event set that contains the events that satisfy a user specified minimum support threshold. The events in LES represent the transactions or objects with large proportion in the entire data set. In this paper, a web log file denotes the data set, and one web page is defined as an event; thus, LES denotes the set of web pages that are accessed by web users with enough frequency in a period of time. In this mining process, the generation sequence through LES can reduce the number of test data to improve the efficiency and accuracy of the mining task. After obtaining large event set, sequence data with only large events are generated. Then the algorithm scans the generated database, finds the set of all frequent items with length (length-1), and calls Algorithm 3 iteratively. Algorithm 3 patternGrowth (P) is the other subroutine of Algorithm 1; it proposes the process to mine the gap-constrained closed sequential patterns with pattern P as the prefix.

Algorithm 1: Improved Gap-BIDE algorithm.



Algorithm 2: Get large event set.

Algorithm 3: Generate closed sequential patterns.

An important definition for generating LES is the user session. The user session is an activity that a user with a unique IP address spends on a web page during a specified period of time. It can be used to identify a continuous access to user statistics visits by this measure. The specified period of time is determined via a cookie, also known as web cookie and HTTP cookie, which can be set by the server with or without an expiration date, modified by web designer and is set to a default value of 600 seconds. Within the expiration date, the access of web user is effective.

3. Discovery of Access Patterns

In this section, the process of mining task is discussed.

3.1. Data Preprocessing

Web log files reside on the web servers that record the activities of clients who access the web server via a web browser. Traditionally, there have been many types of web log files including error logs, access logs, and referrer logs. In this paper, data in the web access log is defined as the raw

data. The web access log records all requests that are processed by the web server. Data in the log file contains some missing value data and irrelevant attributes; it cannot be directly used for the mining task. In this section, we describe the process of data cleaning and attribute selection to remove unwanted data.

(1) *Data cleaning*: removing irrelevant data.
    (a) *Remove the records with URLs of jpg, png, gif, js, css, and so on, which are automatically generated when a web page is requested.*
    (b) *Remove the data with wrong statue numbers that start with the numbers 4 or 5*. These wrong records are caused by the error of requests or server. For example, the HTTP client error: 400 Bad Request and 404 Not Found and HTTP server error: 500 Internal Server Error and 505 HTTP Version Not Supported.
    (c) Discard missing value data that are caused by breaking a web page while loading.
(2) *Attribute selection*: removing the irrelevant attributes. There are many attributes in one record of web log file. In this paper, we need the attributes of IP Address, Time, and URL; thus, the rest of attributes of method, status, size, and so on, need to be discarded.
(3) *Transformed URLs into code numbers.*

It is difficult to distinguish the requested URLs of web log data in thousands of records. There are typically dozens of kinds of web pages in thousands of records. So, the URLs can be transformed into code numbers for simplicity. For example, a web log data that comes from the server of website http://www.vtsns.edu.rs/, and there are 31 different kinds of web pages that have been accessed. We transform their URLs into code numbers, such as galerija.php → 1, nenastavno_osoblje.php → 15, and rezultati_ispita.php → 21.

We choose a set of data from a web log file as an example data. After data preprocessing, we get the clean data shown in Table 1.

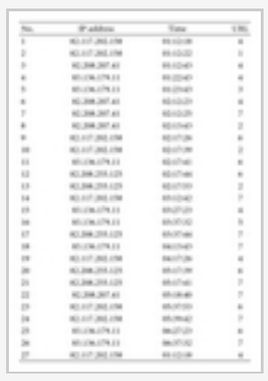


Table 1: Example data.

3.2. Process of Discovering Access Patterns

In this section, we present the process of discovering access patterns with an example.

After data preprocessing, we apply the algorithm to web log data. Then, LES is generated with sorting the data in Table 1 by the attributes of IP Address and Time; here, the time of user session is defined as one hour for simplicity. Then, these data are grouped by one hour for each web user; finally, the sorted data is shown in Table 2.

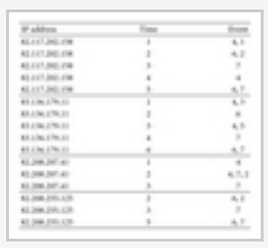


Table 2: Sorted data.

Then, we calculate the support of each event. For example, for the event ⟨2⟩, it occurs three times, which are in "82.117.202.158" at time 2, in "82.208.207.41" at time 2, and in "82.208.255.125" at time 2. After calculating of events support, the candidate event set is obtained as shown in Table 3.

Table 3: Candidate event set.

Finally, a user specified minimum support threshold (MinSup) must be defined. MinSup denotes a kind of abstract level that is a degree of generalization. Choosing MinSup is very important; if it is low, then we can get a detailed event. If it is high, then we can get general events. In this example, MinSup is defined as 75%. In other words, if a web page is accessed by greater than or equal to 75% web users, then this web page can be denoted as a large event. After the process of getting large event set, the LES is obtained as shown in Table 4.

Table 4: Large event set.

After obtaining LES, the infrequent events ⟨1⟩, ⟨3⟩, and ⟨5⟩ are removed from Table 2, and the events are then transformed into a set of tuples (sequence identifier, sequence). We define the IP Address as the sequence identifier and define the event as a sequence. The sequence set is shown in Table 5.

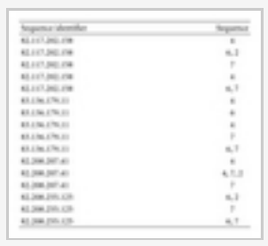


Table 5: Sequence set.

Then, we call the original Gap-BIDE algorithm to find the frequent sequential pattern and prune the patterns. Here, gap is defined as $g(M, N)$, where M is the value of minimum gap, and N is the value of the maximum gap. Assume a pattern P with $g(M, N)$, which can be expressed as $P[M, N]$. This approach is presented like the description of timing constrains with the mingap and maxgap. If the value of M-N is D, then the events in a sequence must occur within D of the events occurring in the previous event.

After calling our improved algorithm, we get the closed patterns as shown in Table 6.

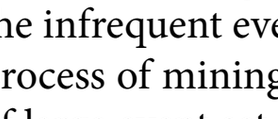


Table 6: Closed patterns.

Useful information can be found from the experimental result. The relationships of web pages are known easily, and user behavior information is shown directly. Each number in the output sequential patterns represents a website or a web user request. For example, the numbers 6 and 7 represent web pages ispit_raspored_god.php and upis_prva.php, respectively. For the closed sequential pattern [6, 7] shown in Table 6, it means 75% (3 out of 4 user sessions) of the web users who access web page upis_prva.php tend to always visit web page ispit_raspored_god.php first. According to the relationship between these two web pages, the design of web pages can be improved. For example, the web designer can add a hyperlink into web page ispit_raspored_god.php that points to web page upis_prva.php. This approach can be applied in many areas. For instance, in the electronic shopping cart, when customers complete their shopping, there can be some hyperlinks in the finished web page that point to some related web pages according to the mining result of purchase history. When web users watch a movie, some hyperlinks that point to some web pages of related movies on the site must be present.

4. Experimental Result and Analysis

4.1. Effect of Parameter in the Process of Getting Large Event Set

The process of getting a large event set aims at extracting the events that satisfy a user defined minimum support of large event set. It can discard the infrequent events to reduce the size of experimental database for reducing the search space and time and maintaining the accuracy of the whole process of mining task. To evaluate the parameter effect, we compare the numbers of large events by changing the values of the minimum support of large event set (MSLE). In this experiment, the experimental data records the access information of website (http://www.vtsns.edu.rs/), which is an institution's official website. The number of original records in the web log file is 5999, and after data preprocessing, there are 269 user sessions in the records. The experimental result is shown in Figure 1. We can see that the smaller the minimum support are, the more generalized the obtained LES becomes. There always exists a value of minimum support, and from the value, the number of large events will not change, or will change very little. This value is always selected to be used as the value of minimum support in the experiment.

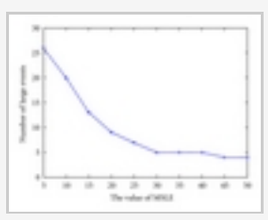


Figure 1: Effect of parameter in the process of getting large event set.

4.2. Comparing with Original Gap-BIDE Algorithm

In this section, we compare our algorithm with the original Gap-BIDE algorithm [19]. The experimental data come from internet information server (IIS) logs for msnbc.com and news-related portions of msn.com for the entire day of September 28, 1999. Each sequence in the dataset corresponds to page views of a user during that twenty-four hour period. Each event in the sequence corresponds to a user's request for a page. There are 989818 anonymous user sessions; we choose the test data by the approach of simple random sampling without replacement from these data. In the experiment, we define minimum support threshold of large event set as 20, minimum support of closed sequential pattern as 10, and the value of gap as $[0, 2]$. We implemented the experiment on a 2.40-GHz Pentium PC machine with 4.00 GB main memory and ran the algorithm in Python 2.7 with JDK 1.6.0. Then, the experimental result is shown in Figure 2. It shows that when applying our proposed algorithm, the cost of time is less than that of the original Gap-BIDE algorithm.

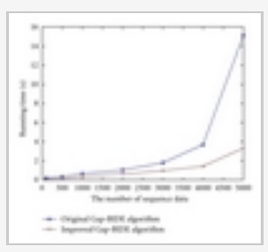


Figure 2: Comparing with original Gap-BIDE.

4.3. Comparing with GSP Algorithm

Previous studies have shown that our proposed algorithm is more effective than original Gap-BIDE algorithm when we apply the algorithms on discovering access patterns. In this section, we want to prove that our proposed algorithm is more effective than previous access pattern mining algorithm. To validate it, we compare our algorithm and GSP algorithm proposed in [17] with an experiment. The experimental data come from Internet information server (IIS) logs for msnbc.com and news-related portions of msn.com for the entire day of September 28, 1999, and we choose the test data by the approach of simple random sampling without replacement from these data. In the experiment, we define minimum support of closed sequential pattern as 10 and the experimental result is shown in Figure 3. It shows that when applying our proposed algorithm to large database, the cost of time is less than that GSP algorithm.

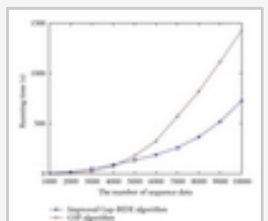


Figure 3: Comparing with GSP algorithm.

5. Conclusion

In this paper, we presented the application of improved Gap-BIDE algorithm for discovering closed sequential patterns in web log data. We improve the algorithm by discarding all infrequent events before generating the frequent candidate events. In the process of data preprocessing, we removed the irrelevant attributes and transformed URLs into code numbers for simplicity, and we removed the missing value data to improve the quality of data. For getting experimental data for the mining task, we transformed the web log data into sequences based on the time constraint. The value of time is determined by an expiration date of the cookies. As a result, we obtained new web access patterns that expressed the order in which websites were access based on the Gap-BIDE algorithm. Compared with the previous web mining approaches, the proposed approach achieves the best performance in terms of getting a large event set of sequence. It reduces the sequences to get more effective and accurate results. We performed some experiments to compare our algorithm with previous algorithms. The experiments show that our algorithm uses less time than the original Gap-BIDE algorithm and cost less time than GSP algorithm in discovering access patterns in large database. In future work, we will try to find a more efficient algorithm for mining the closed gap constraint sequential patterns and will try to achieve a more efficient way for transforming web log files into sequence patterns.

Acknowledgment

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MEST) (no. 2012-0000478).

References

1. L. K. J. Grace, V. Maheswari, and D. Nagamalai, "Analysis of web logs and web user in web mining," *International Journal of Network Security & Its Applications*, vol. 3, no. 1, 2011.
2. K. Saxena and R. Shukla, "Significant interval and frequent pattern discovery in web log data," *International Journal of Computer Science Issue*, vol. 7, no. 1, 2010.
3. K. Suresh and S. Paul, "Distributed linear programming for weblog data using mining techniques in distributed environment," *International Journal of Computer Applications (0975–8887)*, vol. 11, no. 7, 2010.
4. Y. Wang, J. Le, and D. Huang, "A method for privacy preserving mining of association rules based on web usage mining," in *International Conference on Web Information Systems and Mining (WISM '10)*, vol. 1, pp. 33–37, IEEE Computer Society Washington, Washington, DC, USA, 2010.
5. C. Wei, W. Sen, Z. Yuan, and L. C. Chang, "Algorithm of mining sequential patterns for web personalization services," *ACM SIGMIS Database*, vol. 40, no. 2, pp. 57–66, 2009. View at Publisher · View at Google Scholar
6. J. Zhu, H. Wu, and G. Gao, "An efficient method of web sequential pattern mining based on session filter and transaction identification," *Journal of Networks*, vol. 5, no. 9, pp. 1017–1024, 2010. View at Publisher · View at Google Scholar · View at Scopus
7. X. Yu, M. Li, and H. Kim, "Mining access patterns using temporal interval relational rules from web logs," in *Proceedings of the 4th International Conference (FITAT/DBMI '11)*, pp. 80–83, 2011.
8. M. Santini, "Cross-testing a genre classification model for the web," *Genres on the Web*, vol. 42, Part 3, pp. 87–128, 2011.
9. J. J. Rho, B. J. Moon, Y. J. Kim, and D. H. Yang, "Internet customer segmentation using web log data," *Journal of Business & Economics Research*, vol. 2, no. 11, 2004.
10. N. Kejžar, S. K. Èerne, and V. Batagelj, "Network analysis of works on clustering and classification from web of science," in *Proceedings of the 11th Conference of the International Federation of Classification Societies (IFCS '10)*, Part 3, pp. 525–536, 2010.
11. G. Xu, Y. Zong, and P. Dolog, "Co-clustering analysis of weblogs using bipartite spectral projection approach," in *Proceedings of the 14th International Conference on Knowledge-Based and Intelligent Information and Engineering Systems (KES '10)*, vol. 6278, pp. 398–407, 2010.
12. A. A. O. Makanju, A. N. Zincir-Heywood, and E. E. Milios, "Clustering event logs using iterative partitioning," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '09)*, pp. 1255–1263, July 2009. View at Publisher · View at Google Scholar · View at Scopus
13. J. Wang, Y. Mo, B. Huang, and J. Wen, "Web search results clustering based on a novel suffix tree structure," in *Proceedings of the 5th International Conference on Autonomic and Trusted Computing (ATC '08)*, vol. 5060, pp. 540–554, 2008.
14. J. Chen and T. Cook, "Mining contiguous sequential patterns from web logs," in *Proceedings of the 16th International World Wide Web Conference (WWW '07)*, pp. 1177–1178, May 2007. View at Publisher · View at Google Scholar · View at Scopus
15. M. Saravanan and B. Valaramathi, "Generalization of web log datas using WUM technique," in *Proceedings of the 12th International Conference on Networking, VLSI and signal processing (ICNVS '10)*, pp. 157–165, 2010.
16. N. R. Mabroukeh and C. I. Ezeife, "A taxonomy of sequential pattern mining algorithms," *ACM Computing Surveys*, vol. 43, no. 1, article 3, 2010. View at Publisher · View at Google Scholar · View at Scopus
17. S. Ramakrishnan and A. Rakesh, "Mining sequential patterns: generalizations and performance improvements," *Lecture Notes in Computer Science*, vol. 1057, pp. 3–17, 1996.
18. J. Wang, J. Han, and C. Li, "Frequent closed sequence mining without candidate maintenance," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 8, pp. 1042–1056, 2007. View at Publisher · View at Google Scholar · View at Scopus
19. C. Li and J. Wang, "Efficiently mining closed subsequences with gap constraints," in *Proceedings of International Conference on Data Mining*

(SIAM '08), April 2008. View at Scopus

20. X. Yu, M. Li, D. G. Lee, K. D. Kim, and K. H. Ryu, "Application of closed gap-constrained sequential pattern mining in web log data," in *Proceedings of the 2nd International Conference of Electrical and Electronics Engineering (ICEEE '11)*, pp. 649–657, 2011.

21. X. Yu, M. Li, H. Kim, D. G. Lee, and K. H. Ryu, "A novel approach to mining access patterns," in *Proceedings of the 3rd International Conference on Awareness Science and Technology*, pp. 346–352, 2011.