Different Approaches of Mining Web Navigation Pattern: Survey

Suchita A.Chavan
ME, Software Engineering
MIT, Aurangabad

ABSTRACT
Understanding the navigational behavior of website visitors is a significant factor of success in the emerging business models of electronic commerce and even mobile commerce. In this paper, we describe the different approaches of mining web navigation pattern.

General Terms
Web access pattern, Sequential patterns

Keywords
WAP, PTG,TSP

INTRODUCTION

The World Wide Web is a hypertext body of more than 800 million pages that continues to grow. It exceeds six terabytes of data on roughly three million servers. Almost 1 million pages get added daily, and typically, several hundred gigabytes are changed every month. Hence, it is clear that the Web currently constitutes one of the largest dynamic data repositories. In addition to its ever-expanding size and lack of structure, the World Wide Web has not been responsive to user preferences and interests.

Recently, data mining techniques have been applied to extract usage patterns from Web log data for development of electronic commerce, mobile commerce, improving web page organization.

There are numerous studies on the navigation behavior of website visitors. Most are conducted by the techniques of mining Web access patterns, such as improving the access efficiency of Web pages by the adaptive website system, reorganizing a website dynamically, identifying the target group of Web visitors, strengthening the performance of Web searches, and predicting user behaviour patterns in mobile Web systems.

1. BASIC TECHNIQUES TO DISCOVER WEB ACCESS PATTERN

Research efforts to discover Web access patterns focus on three main paradigms in paper [2] i.e. association rules, sequential patterns, and clustering.

1.1 Association rules.

These are probably the most elementary data mining technique and, at the same time, the most used technique in Web Usage Mining. Association rules are implications of the form X) Y where the rule body X and the rule head Y are set of items within a set of transactions. The rule X)Y states that the transactions which contain the items in X are likely to contain also the items in Y. When applied to Web Usage Mining, association rules are used to find associations among Web pages that frequently appear together in users_ sessions. The typical result has the form: A:html; B:html) C:html" which states that if a user has visited page A.html and page B.html, it is very likely that in the same session the same user has also visited page C.html.

1.2 Sequential patterns:

These are used to discover frequent subsequences among large amount of sequential data. In Web Usage Mining, sequential patterns are exploited to find sequential navigation patterns that appear in users_ sessions frequently. The typical sequential pattern has the following form [14]: the 70% of users who first visited A.html and then visited B.html afterwards, have also accessed page C.html in the same session. Sequential patterns might appear syntactically similar to association rules; in fact algorithms to extract association rules can also be used for sequential pattern mining. However, sequential patterns include the notion of time, i.e., at which point of the sequence a certain event happened. In the above example, pages A, B, and C appears sequentially, one after another, in the user sessions; in the previous example on association rules, information about the event sequence is not considered.

There are essentially two classes of algorithms that are used to extract sequential patterns: one includes methods based on association rule mining; the other one includes methods based on the use of tree structures and Markov chains to represent navigation patterns. Some well-known algorithms for mining association rules have been modified to extract sequential patterns. For instance, [12,13]used AprioriAll and GSP, two extensions of the Apriori algorithm for association rules mining . Ref. [11] argues that algorithms for association rule mining (e.g., Apriori) are not efficient when applied to long sequential patterns, which is an important drawback when working with Web logs. Accordingly, [11] proposes an alternative algorithm in which tree structures (WAP-tree) are used to represent navigation patterns. The algorithm (WAP-mine) [11] and the data structure (WAP-tree), specifically tailored for mining Web access patterns, WAP-mine outperforms other Apriori-like algorithms [11] like GSP.Tree structures are also used .

The GSP Algorithm makes multiple passes over data. The first pass determines the frequent 1-item patterns ($L1$). Each subsequent pass starts with a seed set: the frequent sequences found in the previous pass ($Lk-1$). The seed set is used to generate new potentially frequent sequences, called candidate sequences (Ck). Each candidate sequence has one

more item than a seed sequence. In order to obtain *k*-sequence candidate *Ck*, the frequent sequence *Lk*−1 joins with itself Apriori-gen way. This requires that every sequence *s* in *Lk*−1 joins with other sequences *s* in *Lk*−1 if the last *k-2* elements of *s* are the same as the first *k-2* elements of *s*. For example, if frequent 3-sequence set *L*3 has 6 sequences as follows: *{*((1, 2) (3)), ((1, 2) (4)), ((1) (3, 4)), ((1, 3) (5)), ((2) (3, 4)), ((2) (3) (5))*}*. Inorder to obtain frequent 4-sequences, every frequent 3-sequence should join with the other 3-sequences that have the same first two elements as its last two elements. Sequence *s* = ((1, 2) (3)) can join with *s* =((2) (3, 4)) to generate a candidate 4-sequence because the last 2 elements of *s*, (2) (3), are the same as the first 2 elements of *s*. Then, element (4) can be added to the sequence ((1, 2) (3)). Since element (4) is part of the last element (3, 4) of *s*, ((2) (3, 4)), the new sequence is ((1, 2) (3, 4)). Also, ((1, 2) (3)) can join with ((2) (3) (5)) to form ((1, 2) (3) (5)). The remaining sequences can not join with any sequence in *L*3. Following the join phase is the pruning phase, when the candidate sequences that have any of their contiguous ($k - 1$)-subsequences having a support count less than the minimum support, are dropped. The supports for the remaining candidate sequences are found next to determine which of the candidate sequences are actually frequent (*Lk*). These frequent candidates become the seed for the next pass. The algorithm terminates when there are no frequent sequences at the end of a pass, or when there are no candidate sequences generated. The GSP algorithm uses a hash tree to reduce the number of candidates that are checked for support in the database.

1.3 Clustering techniques:

It looks for groups of similar items among large amount of data based on a general idea of distance function which computes the similarity between groups. Clustering has been widely used in Web Usage Mining to group together similar sessions.

2. ALGORITHMS FOR MINING WEB ACCESS PATTERNS

2.1 PrefixSpan

As per [5], Prefixspan is a pattern-growth method like FreeSpan, which reduces the search space for extending already discovered prefix pattern *p* by projecting a portion of the original database that contains all necessary data for mining sequential patterns grown from *p*. While FreeSpan supports frequent pattern guided projection, PrefixSpan supports prefix guided projection. Thus, projected database for each *f* -list prefix pattern *α* consists of all sequences in the original database *D*, containing the pattern *α* and only the subsequences prefixed with the first occurrence of *α* are included. Although PrefixSpan projects smaller sized databases than FreeSpan, they both still incur non-trivial costs for constructing and storing these projected databases for every sequential pattern in the worst case.

2.1.1 Analysis of prefix span

No candidate sequence needs to be generated by PrefixSpan. Unlike a priori-like algorithms, Prefix-Span only grows longer sequential patterns from the shorter frequent ones. It neither generates nor tests any candidate sequence nonexistent in a projected database. Comparing with GSP, which generates and tests a substantial number of candidate sequences, PrefixSpan searches a much smaller space.

Projected databases keep shrinking. As a projected database is smaller than the original one because only the suffix subsequences of a frequent prefix are projected into a projected database. In practice, the shrinking factors can be significant because 1) usually, only a small set of sequential patterns grow quite long in a sequence database and, thus, the number of sequences in a projected database usually reduces substantially when prefix grows; and 2) projection only takes the suffix portion with respect to a prefix. Notice that FreeSpan also employs the idea of projected databases. However, the projection there often takes the whole string (not just suffix) and, thus, the shrinking factor is less than that of PrefixSpan.

The major cost of PrefixSpan is the construction of projected databases. In the worst case, PrefixSpan constructs a projected database for every sequential pattern. If there exist a good number of sequential patterns, the cost is nontrivial.

The above analysis shows that the major cost of PrefixSpan is database projection, i.e., forming projected databases recursively. Usually, a large number of projected databases will be generated in sequential pattern mining. If the number and/or the size of projected databases can be reduced, the performance of sequential pattern mining can be further improved.

One technique which may reduce the number and size of projected databases is pseudo projection. The idea is outlined as follows: Instead of performing physical projection, one can register the index (or identifier) of the corresponding sequence and the starting position of the projected suffix in the sequence. Then, a physical projection of a sequence is replaced by registering a sequence identifier and the projected position index point. Pseudo projection reduces the cost of projection substantially when the projected database can fit in main memory.

Optimization techniques include (1)bi-level projecting for reducing the number and sizes of projected databases, and (2) Pseudo-projection for projecting memory-only databases, where each projection consists of the pointer to the sequence and offset of the postfix to the sequence.

2.2 Apriori

2.2.1 The general structure of the algorithms

They make multiple passes over the data. In each pass, start with a seed set of large sequences. Then use the seed set for generating new potentially large sequences, called candidate sequences. Find the support for these candidate sequences during the pass over the data. At the end of the pass, determine which of the candidate sequences are actually large. These large candidates become the seed for the next pass. In the first pass, all 1-sequences with minimum support, obtained in the litemset phase, form the seed set. Then present two families of algorithms, which we call count-all and count-some. The count-all algorithms count all the large sequences, including non-maximal sequences. The non-maximal sequences must then be pruned out (in the maximal phase). present one count-all algorithm, called AprioriAll, based on the Apriori algorithm for finding large itemsets .Then present two count-some algorithms: Apriori-Some and Dynamicsome. The intuition behind these algorithms is that since they are only interested in maximal sequences, they avoid counting sequences which are contained in a longer sequence if we first count longer sequences. However, not to count a lot of longer sequences that do not have minimum support. Otherwise, the time saved by not counting sequences contained in a longer sequence may be less than the time wasted counting sequences without minimum support that would never have

been counted because their subsequences were not large. Both the count-some algorithms have a forward phase, in which find all large sequences of certain lengths, followed by a backward phase, where find all remaining large sequences. The essential difference is in the procedure they use for generating candidate sequences during the forward phase. As AprioriSome generates candidates for a pass using only the large sequences found in the previous pass and then makes a pass over the data to find their support. Dynamicsome generates candidates onthe- fly using the large sequences found in the previous passes.

2.2.2 Drawbacks of Apriori:

The apriori-like sequential pattern mining method,though reducing search space, bears three nontrivial, inherent costs that are independent of detailed implementation techniques

A huge set of candidate sequences could be generated in a large sequence database. Since the set of candidate sequences includes all the possible permutations of the elements and repetition of items in a sequence, the a priori-based method may generate a really large set of candidate sequences even for a moderate seed set. For example, two frequent sequences of length-1, (a) and (b) will generate five candidate sequences of length-2: (aa), (ab), (ba), (bb), and ((ab)), where ((ab)) represents that two events, a and b, happen in the same time slot. If there are 1,000 frequent sequences of length-1, such as (a1); (a2); . . . ; (a1000), an a priori-like algorithm will generate

$$1,000 \times 1,000 + \frac{1,000 \times 999}{2} = 1,499,500$$

candidate sequences. Notice that the cost of candidate sequence generation, test, and support counting is inherent to the a priori-based method, no matter what technique is applied to optimize its detailed implementation.

Multiple scans of databases in mining. The length of each candidate sequence grows by one at each database scan. In general, to find a sequential pattern of length l, the apriori-based method must scan the database at least l times. This bears a nontrivial cost when long patterns exist.

The apriori-based method generates a combinatorially explosive number of candidates when mining long sequential patterns. A long sequential pattern contains a combinatorial explosive number of subsequences, and such subsequences must be generated and tested in the a priori-based mining. Thus, the number of candidate sequences is exponential to the length of the sequential patterns to be mined. For example, let the database contain only one single sequence of length 100, (a1 a2 . . . a100), and the min_support threshold be 1 (i.e., every occurring pattern is frequent). To (re)derive this length-100 sequential pattern, the a priori-based method has to generate 100 length-1 candidate sequences (i.e., (a1),(a2); . . . ; (a100)), $100 \times 100 + \frac{100 \times 99}{2} = 14,950$ length-2 candidate sequences, $\binom{100}{3} = 161,700$ length-3 candidate sequences,1 and so on. Obviously, the total number of candidate sequences to be generated is $\sum_{i=1}^{100} \binom{100}{i} = 2^{100} - 1 \approx 10^{30}$

2.3 WAP-tree algorithm

In [1] proposed an algorithm using WAP-tree, which stands for web access pattern tree. This approach is quite different from the Apriori-like algorithms. The main steps involved in this technique are summarized next. The WAP-tree stores the web log data in a prefix tree format similar to the frequent pattern tree (FP-tree) for non-sequential data.

2.3.1 The general structure of the algorithms

1. The algorithm first scans the web log once to find all frequent individual events.
2. It scans the web log again to construct a WAP-tree over the set of frequent individual events of each transaction.
3. It finds the conditional suffix patterns.
4. In the fourth step, it constructs the intermediate conditional WAP-tree using the pattern found in previous step.
5. Finally, it goes back to repeat Steps 3 and 4 until the constructed conditional

WAP-tree has only one branch or is empty. Thus, with the WAP-tree algorithm, finding all frequent events in the web log entails constructing the WAP-tree and mining the access patterns from the WAP tree.

WAP-tree algorithm scans the original database only twice and avoids the problem of generating explosive candidate sets as in Apriori-like algorithms. Mining efficiency is improved sharply,

2.3.2 Drawback

WAP-tree mining is recursively constructs large numbers of intermediate WAP-trees during mining and this entails storing intermediate patterns, which are still time consuming operations& which is rather costly. The mining method using the WAP-tree alleviates both problems of scanning the database repeatedly and generating tremendous candidate sequences.

2.4 Incremental and interactive mining of web traversal patterns

The essence of incremental data mining and interactive data mining is the ability to use previous mining results in order to reduce unnecessary processes when web logs or web site structures are updated, or when the minimum support is changed.

As per [3] ,they propose two novel incremental web traversal pattern mining algorithms for the maintenance of web traversal patterns when a database is updated or a web site structure is changed. Also present an interactive web traversal pattern mining algorithm to find all web traversal patterns when min_sup is adjusted. This algorithm utilizes previous mining results to find new web traversal patterns such that the total mining time can be reduced.For that used the lattice structure to store the previous mining results for incremental Web traversal patterns. The patterns may be obtained rapidly when the database or the website structure is updated. The problem of choosing an appropriate storage structure to store previous mining results now becomes very important.

2.4.1 Drawback

The size of the lattice structure may become too large to be loaded into the main memory.

2.5 BIDE Algorithm

In [7] BIDE, an efficient algorithm for mining frequent closed sequences without candidate maintenance. The frequent closed sequences are regarded as a pattern closure of all frequent sequential patterns .It adopts a novel sequence closure checking scheme called BI-Directional Extension and prunes the search space more deeply compared to the previous algorithms by using the Back Scan pruning method. A thorough performance study with both sparse and dense, real, and synthetic data sets has demonstrated that BIDE significantly outperforms the previous algorithm: It consumes an order(s) of magnitude less memory and can be more than an order of magnitude faster. It is also linearly scalable in terms of database size.

2.5.1 Drawback

It consumes a lot of memory and leads to a huge search space for pattern closure checking

3. Future work

In this survey paper, it is noticed that the former mining algorithms suffer from either repetitive database scan or high memory load. For algorithms with a single database scan, they build special data structures to store the sequences in the database. However, it may be difficult to hold all sequences of the database in the data structure.

So if we use Graph Traverse algorithm which will pay attention to the tracks of website visitors will mine TSP, in which the memory is loaded with the hyperlink structure of the website instead of the sequence database. Then this method can help to improve an efficient and effective way to realize what targets the visitors may reach and how they are achieved.

4. REFERENCES

[1] Ezeife, C. I., & Lu, Y. (2005). Mining Web log sequential patterns with position coded pre-order linked WAP-tree. Data Mining and Knowledge Discovery, 10,5–38.

[2] Facca, F. M., & Lanzi, P. L. (2005). Mining interesting knowledge from Weblogs: A survey. Data and Knowledge Engineering, 53, 225–241.

[3] Lee, Y.-S., & Yen, S.-J. (2007). Incremental and interactive mining of Web traversal patterns. Information Sciences, 178(2), 278–306.

[4] Li, H.-F., Lee, S.-Y., & Shan, M.-K. (2006). DSM-PLW: Single-pass mining of path traversal patterns over streaming Web click-sequences. Computer Networks, 50,1474–1487.

[5] Pei, J., Han, J., Mortazavi-Asl, B., Wang, J., Pinto, H., Chen, Q., et al. (2004). Mining sequential patterns by pattern-growth: The prefixspan approach. IEEE Transactions on Knowledge and Data Engineering, 16(10), 1–17.

[6] Tseng, V.-S., & Lin, K.-W. (2006). Efficient mining and prediction of user behavior patterns in mobile Web systems. Information and Software Technology, 48,357–369.

[7] Wang, J., Han, J., & Li, C. (2007). Frequent closed sequence mining without candidate maintenance. IEEE Transactions on Knowledge and Data Engineering, 19(8), 1042–1056.

[8] Xing, D., & Shen, J. (2004). Efficient data mining for Web navigation patterns.Information and Software Technology, 46, 55–63.

[9] Yan, X., Han, J., & Afshar, R. (2003). CloSpan: Mining closed sequential patterns in large datasets. In Third SIAM international conference on data mining (SDM), SanFrancisco. CA (pp. 166–177).

[10] Yen, S.-J., & Chen, A. L. P. (2001). A graph-based approach for discovering various types of association rules. IEEE Transactions on Knowledge and Data Engineering, 13(5), 839–845.

[11] J. Pei, J. Han, B. Mortazavi-asl, H. Zhu, Mining access patterns efficiently from web logs, in: Pacific-Asia Conference on Knowledge Discovery and Data Mining, 2000, pp. 396–407.

[12] X. Huang, N. Cercone, A. An, Comparison of interestingness functions for learning web usage patterns, in:Proceedings of the Eleventh International Conference on Information and Knowledge Management, ACM Press,2002, pp. 617–620.\

[13] B. Mortazavi-Asl, Discovering and mining user web-page traversal patterns, Master_s thesis, Simon Fraser University, 2001

[14] E.S. Nan Niu, M. El-Ramly, Understanding web usage for dynamic web-site adaptation: A case study, in: Proceedings of the Fourth International Workshop on Web Site Evolution (WSE_02), IEEE, 2002, pp. 53–64.