IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 14, Issue 5 (Sep. - Oct. 2013), PP 70-73
www.iosrjournals.org

Incremental Mining of Sequential Patterns Using Weights

Archana Kollu[1], Vinay Kumar Kotakonda[2]
[1]*(Asst.Prof, Department of Computer Engineering, Padmashree Dr.D.Y.P.I.E.T,Pune/India)*
[2]*(Software Engineer,Bengaluru/ India)*

Abstract : *Real life sequential databases are usually not static. They tend to grow incrementally. So after every update a frequent pattern may no longer remains frequent while some infrequent patterns may appear as frequent in updated database. It is not a good idea to mine sequential database from scratch every time as the update occurs. It would be better if we can use the knowledge of already mined sequential patterns to find the complete set of sequential patterns for updated database. An incremental mining algorithm does the same thing. The main goal of an incremental mining algorithm is to reduce the time taken to find out the frequent patterns significantly i.e. it should mine the set of frequent patterns in significantly less time than a non-incremental mining algorithm. In this work we have approached using weight constraints, in time and space, of an idea of already existing algorithm called WSM.*
Keywords: *Frequent patterns, incremental mining, Sequential databases, sequential patterns, weight constrains*

I. Introduction

Many real life sequence databases grow incrementally. Discovering sequential patterns from the dynamic databases is of great importance in many application domains (e.g., fault detection in network, web access pattern analysis and DNA Sequencing and Bioinformatics). Incremental algorithm should be developed for sequential pattern mining so that mining can be adapted to incremental database updates. However, it is nontrivial to mine sequential patterns incrementally, especially when the existing sequences grow incrementally because such growth may lead to the generation of many new patterns due to the interactions of the growing subsequences with the original ones. In this study, we develop an efficient algorithm, for incremental mining of sequential patterns using weights, by exploring some interesting properties. Our performance study shows that the algorithm can perform better than that of other non-incremental mining algorithm.

Most frequent pattern mining algorithms use a support measure [1] to prune the combinatorial search space. However, support-based pruning is not enough when taking into consideration the characteristics of real datasets. Additionally, after mining datasets to obtain the frequent patterns, there is no way to adjust the number of frequent patterns through user feedback, except for changing the minimum support. Alternative measures for mining frequent patterns have been suggested to address these issues. One of the main limitations of the traditional approach for mining frequent patterns [2] is that all items are treated uniformly when, in reality, items have different importance. For this reason, weighted frequent pattern mining algorithms [3] have been suggested that give different weights to items according to their significance.

Sequential pattern mining is not a new research topic these days. Many researchers have proposed their algorithms for mining sequential patterns which outperform their previous algorithms in one way or the other. All the algorithms have their own advantages and shortcomings. Some algorithms are very efficient in a particular environment (e.g. on a particular data), but not so efficient or even inefficient in other circumstances.

Sequential mining algorithms can mine a static database. But, nowadays, almost all databases are dynamic in nature and they grow incrementally. One way to handle this is to mine the whole database every time an update occurs. But it is highly inefficient and also undesirable. We must find a way to use the already mined information. An incremental mining algorithm does the same [4, 5]. It utilizes the mined information to get new set of frequent sequential patterns instead of mining the whole database from scratch. Note that the ultimate aim of using an incremental mining algorithm instead of non-incremental one is to gain efficiency with respect to time. Otherwise a non-incremental mining algorithm can also serve our purpose of mining very easily. So for incremental mining algorithm we must consider the time requirement of the algorithm to mine complete set of frequent patterns.

II. Problem Definition And Related Work

2.1 Problem Definition

Given an original database D, minimum support threshold min_sup, the set of frequent sequential patterns FP and an update to the original database, the problem is to mine the complete set of frequent sequential patterns for the updated database D’, utilizing the already mined information instead of mining it from scratch. The ultimate aim of an incremental mining algorithm is to reduce the time required to mine D’, at the same time giving the same result as a non-incremental mining algorithm.

2.2 Related work

Our work is mainly based on an incremental mining algorithm called **WSM** [3]. So it would be useful to discuss **WSM** and then proceed with the algorithm proposed by us.

WSM : An overview

Algorithm 1 WSM (α, L, $SDB|\alpha$)

Parameter:

α is a weighted sequential pattern that satisfies the above pruning conditions;
L is the length of α;
$SDB|\alpha$ is the sequence database, ***S*** if α is null, otherwise, it is the α-projected database.

```
1. Scan S once, count the support of each item, and find each weighted frequent item $\beta$ in sequences,
such that $sup(\beta)*MeanWeight \geq min\_sup$.
2. $\beta$ is a weighted sequential item if the following pruning condition is not satisfied.
(a) $\beta$ can be assembled to the last element of $\alpha$ to form a sequential pattern or
(b) $<\beta>$ can be appended to $\alpha$ to form a sequential pattern.
3. for each weighted frequent item $\beta$ do
Add it to a to form a sequential pattern $\alpha'$
if $\alpha'$ is not a sub-sequence of a discovered sequence
then
        output $\alpha'$.
end for
4. for each $\alpha'$ do
 Construct $\alpha'$-projected database $SDB|\alpha'$;
  Call WSM ($\alpha'$, $L+1$, $SDB|\alpha'$);
end for
```

2.3 Pruning Strategies

The main consideration in applying a weight constraint to sequential pattern mining is that the downward closure property [1] may be broken by simply applying a weight constraint. A sequence with lower weight can be a frequent sequential pattern by combining items with higher weight in the sequence. To tackle these problems, a *mean weight* is defined [3].

Condition 1. If $sup(\alpha)*MeanWeight<min\_sup$, then we can prune α.

Condition 2. For all discovered frequent weighted sequential patterns, if the weight of α is lower than MeanWeight, then we can prune α.

The meaning of Condition 2 is that if the importance of items in α is no higher than the mean weight, then it is not important enough. We use Condition 1 for pruning because it can also inherits the downward closure property [1, 4]

Theorem :. Any super sequence of an infrequent weighted sequence is also infrequent. [3]

Proof: Let α be an infrequent weighted sequence, that is $sup(\alpha)*MeanWeight<min\_sup$. Let β be a super sequence of α, i.e., $\alpha\subset\beta$, then we have $sup(\beta)\leq sup(\alpha)$. So $sup(\beta)*MeanWeight\leq sup(\alpha)*MeanWeight<min\_sup$. Thus, β is an infrequent weighted sequence.

Given a sequence database S of input sequences and min_sup, the WSM is to find all frequent weighted maximal sequences in the database. Thus, we also have the following condition for pruning.

Condition 3. Let α be a newly discovered sequence, if $\alpha\subseteq\beta$ and β has already been found, then we can prune α.

So by this theorem we will have one condition , a pattern which is frequent in D is still frequent in D' When Mean Weight(D')>Mean Weight(D) i.e. $sup(p)*M.W.(D')>sup(p)*M.W.(D)\geq min\_sup$,if above condition hold then only we can proceed further otherwise it will become WSM approach for D' one more condition is needed

Given a frequent pattern p in D', we want to discover whether there is any pattern p' with p as prefix where p' was infrequent in D but is frequent in D'. A sequence p' which changes from infrequent to frequent must have $\Delta sup(p') > (1-\mu)*min\ sup$. For every frequent pattern p such that $M.W.(D')*sup_{LDB(p)}<(1-\mu)*min\_sup$ Then there does not exist p' as prefix of p having this property $M.W.(D')*sup(p')<\mu*min\_sup$

2.3 Proposed Algorithm

Input: An appended database D', min_sup, FP and BSFP in D

Output: FP', BSFP' in D'

```
1.FS' ={} ; SFS' ={} ;
2. Scan LDB for single items;
3.adjust the min_sup if it is changed due to the increasing of total number of sequences in D'.
4. Add new frequent item into FS', semi frequent item into SFS according to the updated min-sup.
5. For each new item in FS' do
            WSM(p,D'|p , μ*min_sup , FS',SFS')
    6. For each pattern p in FS do
            If(M.W.(D')*$sup_{LDB}(p) \geq (1-\mu)$*min_sup
            WSM(p, D'|p, μ ,min_sup,FS',SFS');
            Check $\Delta sup(p)$;
    7. If(M.W.(D')*$sup_{D'}(p) \geq$min_sup
              Insert(FS',p);
    8. Else if (M.W.(D')*$sup_{D'}(p') \geq \mu$*min_sup
             Insert(SFS',p);
```

III. Experimental Results

The experiments are carried out on a Windows O.S, CPU 2.40 GHz with 4 GB RAM. We have implemented our algorithm in JAVA programming language using NetBeans editor. The user visits to webpage data is stored in database, created using MYSQL server 5.1.

The data is an Anonymous web data with discrete sequence. 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. Requests are not recorded at the finest level of detail---that is, at the level of URL, but rather, they are recorded at the level of page category (as determined by a site administrator). The categories are "FrontPage", "news", "tech", "local", and "opinion "," market "," weather "," health "," business, sports "," business", "entertainment" and "travel-package. Any page requests served via a caching mechanism were not recorded in the server logs and, hence, not present in the data. 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. Requests are not recorded at the finest level of detail---that is, at the level of URL, but rather, they are recorded at the level of page category (as determined by a site administrator). The categories are "frontpage", "news", "tech", "local", "opinion", "market", "weather", "health", "business", "sports", "business","entertainment" and "travel-package". Any page requests served via a caching mechanism were not recorded in the server logs and, hence, not present in the data. Each category is associated--in order--with a character starting with "f". For example, "frontpage" is associated with f, "news" with n, and "tech" with t.

We have studied our algorithm with different attributes. We get results for our algorithm as follows: The Fig 1 shows the time required by the algorithms on varying minimum support counts. The horizontal axis represents the minimum support value and vertical axis represents the time required in seconds.

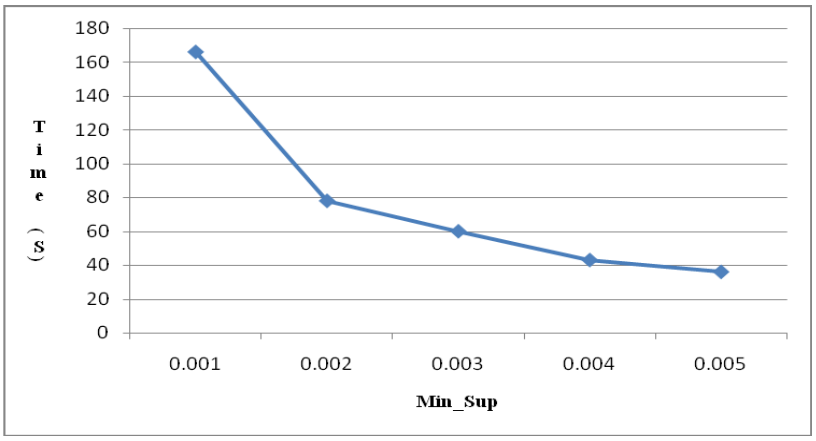

Figure 1 Varying *min_sup*

The Fig 2 below shows the number of frequent patterns generated for original database, ODB and appended database, ADB. We also have semi frequent patterns SFP generated and stored in buffer [4]. The horizontal axis represents the minimum support threshold and the vertical axis represents the number of patterns to be generated and stored.

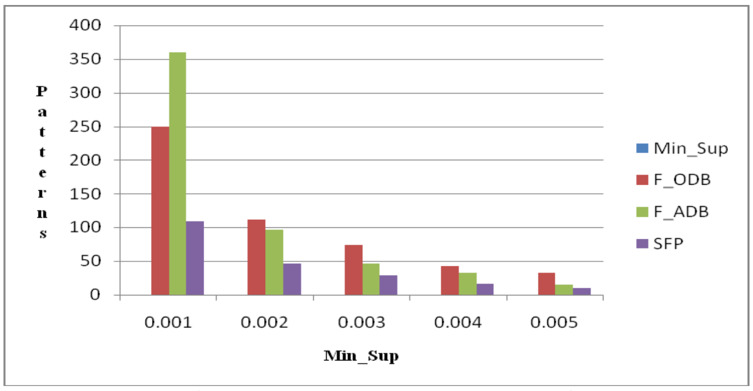

Figure 2 Frequent patterns generated

The Fig 3 demonstrates how our algorithm can be affected by varying buffer ratio . If we set very high, we will have fewer patterns in *SFS*, and then the support update for sequences in *SFS* on *LDB* will be more efficient. However, since we keep less information in *SFS*, we may need to spend more time on projecting databases. In the extreme case = 1, *SFS* becomes empty. On the other hand, if we set the very low, we will have a large number of sequences in *SFS*, which makes the support update stage very slow.

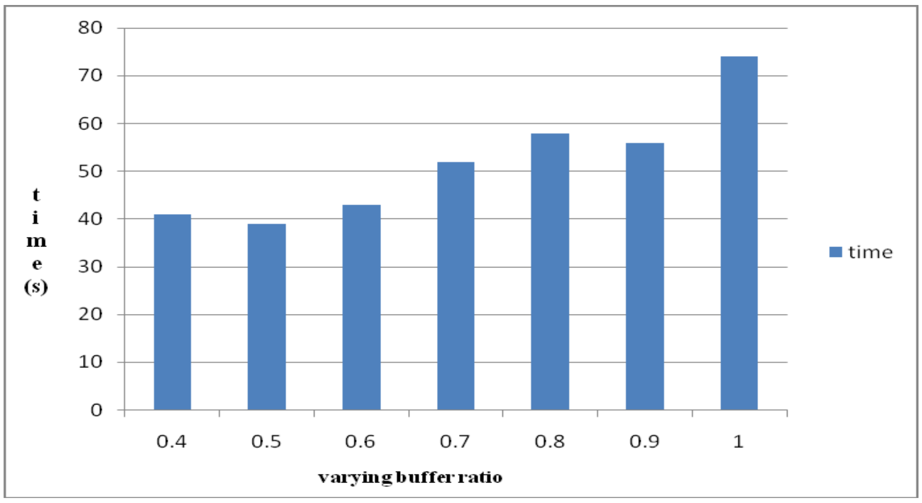

Figure 3 varying buffer ratios

IV. Conclusion And Futurescope

We have proposed an algorithm for incremental mining of sequential patterns using weights. We have to check its efficiency with other existing algorithms and then a new concept of "**time-interval sequential patterns**" has been proposed recently in which not only the order of events but the time between them is also considered. This work can be further extended to develop an incremental mining algorithm for time-interval sequential patterns. We also have the concept of **closed sequential patterns.** Closed sequential patterns represent the information in more compact way and so they have gained much attention of the researchers. So an enthusiastic researcher can also try to develop an efficient algorithm for closed sequential patterns.

References

[1] Agrawal R., Srikant R., *"Mining sequential patterns" , Proc. 11th IEEE Int. Conf. on Data Engineering (ICD E'95), 1995.*

[2] Srikant R., Agrawal R., "*Mining sequential patterns: Generalizations and performance improvements", Proc. 5th IEEE Int. Conf. on Extending Database Technology (IDBT'96).*

[3] WeiCui, Haizhong An *"Discovering Interesting Sequential Pattern in Large Sequence Database", 2009 Second Asia-Pacific Conference on Computational Intelligence and Industrial Applications.*

[4] Cheng H., Yan X., Han J., *"Inc Span: Incremental Mining of Sequential Patterns in Large Database", Proc. ACM KDD Conf. on Knowledge Discovery in Data, Washington (KDD'04), 2004.*

[5] F. Masseglia, P. Ponce let, and M. Teisseire, *"Incremental mining of sequential patterns in large data bases", Data Knowledge Eng., 46:97–121, 2003.*