INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH & DEVELOPMENT

ISSN 2278 – 0211 (Online)
ISSN 2278 – 7631 (Print)

Mining of Temporal Optimal High Utility Item Sets from Data Streams

Bhavana Jamalpur
Department Of Computer Science & Engineering
S. R. Engineering College, Anatha Sagar, Waranagal, A. P., India
Professor S. S. V. N. Sharma
Dean, Computer Science and Engineering
Vaagdevi Engineering College, Waranagal, A. P., India

Abstract:
Temporal High Utility Item (THUI) mining has become an emerging research topic in the data mining field, and finding frequent item sets is an important task in data mining with wide applications. Two basic factors to be considered in utility mining. First, the utility (e.g., profitability, time) of each item may be different in real applications; second, the frequent itemsets might not produce the highest utility. In this paper, we propose a novel algorithm named TOUIG (Temporal Optimal Utility Item Set Generation) which can find optimal high utility item sets from database. A novel approach namely, TOUI-tree (Temporal Optimal Utility Item set tree), is also proposed for efficiently capturing the utility of each item set with one-time scanning. The main contributions of this paper are as follows: 1) OUIG is the first one-pass utility-based algorithm for mining temporal optimal utility item sets and the experimental results show that our approach produces optimized solution than other existing utility mining algorithms.

Key words: *Data mining, utility mining, temporal high utility item sets*

1. Introduction

Temporal data mining is a single step in the process of Knowledge Discovery in Temporal Databases that enumerates structures (temporal patterns or models) over the temporal data. Examples of temporal data mining tasks are classification and clustering of time series, discovery of temporal patterns or trends in the data, associations of events over time, similarity based time series retrieval, time series indexing and segmentation. Temporal data mining aims to discover hidden relations between sequences and subsequences of events. The discovery of relations between sequences of events involves mainly three steps:

- Representation and modeling
- Sequencing the data in a suitable form
- Similarity measures between sequences application of models and representations to the actual mining problems.

1.1. Problem Statement

Given a data stream, a pre-defined utility table and a user-specified minimum utility threshold, the problem of mining temporal maximal high utility itemsets using the landmark model is to find the set of TMUIs from the landmark timestamp over the transactional database**.**

1.2. Time Series

A Time Series is an ordered sequence of data points. Typically it's measured at successive times spaced at uniform time intervals. A huge amount of data is collected everyday in the form of event time sequences. Common examples are recording of different values of stock shares during a day, each access to a computer by an external network, bank transactions, or events related to malfunctions in an industrial plant. These sequences represent valuable sources of information not only to search for a particular value or event at a specific time, but also to analyze the frequency of certain events, discover their regularity, or discover set of events related by particular temporal relationships. These types of analyses can be very useful for deriving implicit information from the raw data, and for predicting the future behavior of the process that we are monitoring.

2. The Apriori Algorithm

The key idea of the algorithm is to begin by generating frequent item sets with just one item (1-item sets) and to recursively generate frequent item sets with 2 items, then with 3 items, and so on until we have generated frequent item sets of all sizes. It is easy to generate frequent 1-item sets. All we need to do is to count, for each item, how many transactions in the database include the item. These transaction counts are the supports for the 1-item sets. We drop 1-item sets that have support below the desired minimum support to create a list of the frequent 1-item sets.To generate frequent 2-item sets, we use the frequent 1-item sets. The reasoning is that if acertain 1-item set did not exceed the minimum support, then any larger size item set that includes it will not exceed the minimum support. In general, generating *k*-item sets uses the frequent *k* ¡ 1- item sets that were generated in the previous step. Each step requires a single run through the database, and therefore the Apriori algorithm is very fast even for a large number of unique items in a database.

3. A Study On Purchases Of Cell Phone Face-Plates

A store that sells accessories for cellular phones runs a promotion on faceplates. Customers who purchase multiple faceplates from a choice of six different colors get a discount. The store managers, who would like to know what colors of faceplates customers are likely to purchase together, collected the following transaction database.

Transaction	Faceplate	Colors	Purchased	
1	red	White	green	
2	white	Orange		
3	white	Blue		
4	red	White	orange	
5	red	Blue		
6	white	Blue		
7	white	Orange		
8	red	White	blue	green
9	red	white	blue	
10	yellow			

Table 1: Transactions for Purchases of Different Colored Cellular Phone Faceplates

3.1. Generating Candidate Rules

Trans	red	white	blue	orange	green	yellow
1	1	1	0	0	1	0
2	0	1	0	1	0	0
3	0	1	1	0	0	0
4	1	1	0	1	0	0
5	1	0	1	0	0	0
6	0	1	1	0	0	0
7	1	0	1	0	0	0
8	1	1	1	0	1	0
9	1	1	1	0	0	0
10	0	0	0	0	0	1

Table 2

Now, suppose that we want association rules between items for this database that have a support count of at least 2 (equivalent to a percentage support of 2/10=20%). In other words, rules based mon items that were purchased together in at least 20% of the transactions. By enumeration we can see that only the following item sets have a count of at least 2: Item set support (count)

- {red}-> 6
- {white}-> 7
- {blue}-> 6
- {orange}-> 2
- {green}-> 2
- {red, white}-> 4
- {red, blue}-> 4
- {red, green}-> 2
- {white, blue}-> 4
- {white, orange}-> 2
- {white, green}-> 2

- {red, white, blue}-> 2
- {red, white, green}-> 2

The first item set {red} has a support of 6, because 6 of the transactions included a red faceplate. Similarly the last item set {red, white, green} has a support of 2, because only 2 transactions included red, white, and green faceplates.
The computation of confidence in the second stage is simple. Since any subset (e.g., {red} in the phone faceplate example) must occur at least as frequently as the set it belongs to (e.g. {red, White}), each subset will also be in the list. It is then straightforward to compute the confidence as the ratio of the support for the item set to the support for each subset of the item set. For example, from the item set {red,white,green} in the phone faceplate purchases we get the following

3.2. Association Rules

Rule 1:

{red, white} => {green} with
confidence = support of {red, white, green}/
support of {red, white}
= 2/4 = 50%;

Rule 2:

{red, green} => {white} with
confidence = support of {red, white, green}/
support of {red, green}
= 2/2 =100%;

Rule 3:

{white,green} => {*red*} with
confidence = support of {red, white, green}/
support of {white,green}
= 2/2= 50%;

Rule 4:

{red} => {*white*,green} with
confidence = support of {red, white, green}/
support of {red}
= 2/6 = 33%;

Rule 5:

{white} => {*red*,green} with
confidence = support of {red, white, green}/
support of {white}
= 2/7 =29%;

Rule 6:

{green} => {*red,white*} with
confidence = support of {red, white, green}/
support of {green}
= 2/2 = 50%;

3.3. Pseudocode

Algorithm 1. aPriori algorithm[14]

```
Input:
I //Itemsets
D //Transactions
S //support threshold
Output:
L // large itemsets
```

```
aPriori algorithm
k = 0 // k is used as the scan number
L = Ø
C1 = I //Initial candidates are set to be the items
repeat
k = k + 1
Lk = Ø
for each Ii € Ck do
ci = 0 //Initial counts for each itemset are 0
for each tj € D do
for each Ii € Ck do
if Ii € tj then
ci = ci + 1
for each Ii € Ck do
if ci ≥ s do
Lk = Lk U Ii
L = L U Lk
Ck+1 = aPriori-Gen (Lk)
```

Algorithm 2. aPriori-Gen algorithm

```
 Input:
Li-1 //Large itemsets of size i-1
Output:
Ci //Candidates of size i
Apriori-Gen algorithm
Ci = Ø
for each I € Li-1
for each J ≠ I € Li-1 do
if i-2 of the elements in
I and J are equal
then
Ck = Ck U {I U J}.
```

4. Proposed Work

A formal definition of utility mining and theoretical model was proposed where the utility is defined as the combination of utility information in each transaction and additional resources. Another algorithm named Two-Phase was proposed in which achieves for finding high utility itemsets. It presented a Two-Phase algorithm to prune down the number of candidates and can obtain the complete set of high utility itemsets. In the first phase, a model that applies the“ transaction-weighted downward closure property” on the search space to expedite the identification of candidates. In the second phase, one extra database scan is performed to identify the high utility itemsets. However, this algorithm must rescan the whole database when added new transactions from data streams. It need more times on processing I/O and CPU cost for finding high utility itemsets. Hence, Two-Phase algorithm is just only focused on traditional databases and is not suited for data streams. Although there existed numerous studies on high utility itemsets mining and data stream analysis as described above, there is no algorithm proposed for finding temporal high utility itemsets in data streams. This motivates our exploration on the issue of efficiently mining high utility itemsets in temporal databases like data streams in this research. The goal of utility mining is to discover all the itemsets whose utility values are beyond a user specified threshold in a transaction database. Utility mining is to find all the high utility item sets. Two-Phase algorithm for pruning candidate itemsets and simplify the calculation of utility. Is the sum of the transaction utilities of all the transactions containing X. So Phase I overestimates some low utility itemsets, it never underestimate itemset, Second, one extra database scan is performed to filter the overestimated itemsets in phase II, a progressive transaction-weighted utilization set of itemsets

```
Algorithm –TOUIG (Temporal Optimal
                 Utility Itemsets Generation)
Input:

Step1: Sample Database from any

Step2: constraints minus, minconfidence, minutility
```

```
Output: all utility-frequent Itemsets
step 3. Find all quasi-utility frequent itemsets
   i)      Candidateset=QUF-Apriori(DB,minutility,minsup)
Step4. Pruning  Utility-Infrequent itemsets
          for each C
 for each C in candidateset
   for each T in DB
if  C in T a and U(C,T)>=minUtil
         c.count +=1
return {c in candidateset |c.count>=minsup}
end if
Step 5. End
```

5. Conclusion

Predicting high utility itemsets is the future is one aspect in designing profitable day trading strategies. Technical analysis analyzes, price, volume and other market information, whereas fundamental analysis looks at the facts of the company, market, currency or commodity. Most large brokerage, trading group, or financial institutions will typically have both a technical analysis and fundamental analysis team.

6. References

1. Agrawal, R. and Srikant, R. Fast algorithms for mining association rules. In Proc. of the 20th Int'l Conf. on Very Large Data Bases, 1994, 487-499.
2. Chan, R., Yang, Q. and Shen, Y. Mining high utility itemsets. In Proc. of Third IEEE Int'l Conf. on Data Mining, Nov., 2003, 19-26.
3. Chi, Y., Wang, H., Yu, P. S. and Muntz, R. R. Moment: maintaining closed frequent itemsets over a stream sliding window. In Proc. of the IEEE Int'l Conf. on Data Mining, 2004.
4. Lee, D. and Lee, W. Finding maximal frequent itemsets over online data streams adaptively. In Proc. of 5th IEEE Int'l Conf. on Data Mining, 2005.
5. Li, H.-F., Huang, H.-Y., Chen, Y.-C., Liu, Y.-J., Lee, S.-Y. Fast and Memory Efficient Mining of High Utility Itemsets in Data Streams. In Proc. of the 8th IEEE Int'l Conf. on Data Mining, 2008, 881-886.
6. Li, H.-F., Lee, S.-Y., and Shan, M.-K. Online mining (recently) maximal frequent itemsets over data streams. In Proc. of the 15th IEEE Int'l Workshop on Research Issues on Data Engineering, 2005.
7. Lin, C. H., Chiu, D. Y., Wu, Y. H. and Chen, A. L. P. Mining frequent itemsets from data streams with a time-sensitive sliding window. In Proc. of SDM, 2005.
8. Liu, Y., Liao, W. and Choudhary, A. A fast high utility itemsets mining algorithm. In Proc. of the Utility-Based Data Mining Workshop, 2005.
9. Pei, J., Han, J., and Mao, R. CLOSET: An efficient algorithm for mining frequent closed itemsets. In Proc. of the ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, 2000, 11-20.
10. Tseng, V. S., Chu, C. J. and Liang, T. Efficient mining of temporal high utility itemsets from data streams. In ACM KDD Workshop on Utility-Based Data Mining Workshop, Philadelphia, USA, 2006.
11. Tanbeer, S. K., Ahmed, C. F., Jeong, B.-S. and Lee,Y.-K. Efficient frequent pattern mining over data streams.