International Journal of Modern Engineering Research (IJMER)
www.ijmer.com Vol.2, Issue.1, Jan-Feb 2012 pp-431-435 ISSN: 2249-6645

RANKING AND SUGGESTING POPULAR ITEMSETS IN MOBILE STORES USING MODIFIED APRIORI ALGORITHM

P V Vara Prasad[#1] ,Sayempu Sushmitha[*2], Badduri Divya[*3], Gogineni Riharika[*4], Guntur Vijaya Raghu Ram[*5].

#Asst.Professor,Department of Computer Science and Engineering,
K L University, Green Fields, Vaddeswaram, Guntur District, A.P., INDIA. Pincode : 522502
*Students, B.tech, CSE, K L University

Abstract

We considered the problem of ranking the popularity of items and suggesting popular items based on user feedback. User feedback is obtained by iteratively presenting a set of suggested items, and users selecting items based on their own preferences either the true popularity ranking of items, and suggest true popular items. We consider apriori algorithm with some modifications overcoming the complexity that has been seen in other randomized algorithms. The most effective feature of this algorithm is that it reduces the number of database scans and complexity.

1. INTRODUCTION

1.1 TERMINOLOGY:

In this section we first want to introduce the different terms that we were going to use in our paper as fallows.

1.1.1 Ranking: Ranking is giving rank scores to the most popular item by taking user feedback. The most frequently occurring item is given the highest rank score.

1.1.2 Selection: We focus on the ranking of items where the only available information is the observed selection of items. In learning of the users preference over items, one may leverage some side information about items, but this is out of the scope of this paper.

1.1.3 Imitate: The user study was conducted in very famous mobile stores and which has been used to set of mobiles. The user may check the list and select the set of mobiles which they like most and depending on those like results the new suggestion list has been developed by the algorithm.

1.1.4 Popular: In practice, one may use prior information about item popularity. For example, in the survey the user may select the suggested mobile or they may also select the others. If they selected the already suggested items they will become more popular and if he don't they may get out of the popular list.

1.1.5 Association Rule: Association Rules are if/then statements that help uncover relationships between seemingly unrelated data in the relational database or other information repository. An example of an association rule would be **if a customer buys a nokia mobile, he is 70% interested in also purchasing nokia accessories**.

2. THEORETICAL STUDY

We consider the mobile phone selection and suggesting the best sold mobile and their combinations that were most liked by most of the users. Consider a set of mobiles M: (m1, m2, m3, m4, ….mn) where n > 1. Now we were calculating the set of items in C where were mostly sold and mostly liked by the users, as S

S: (s1, s2, s3, s4, …. sg) where g > 1.

We need to consider an item I, we interpret si as the portion of users that would select item i if suggestions were not made. We assume that the popularity rank scores s as follows:

a) Items of set S were estimated to is as s1 $\geq$ s2 $\geq$ s3 $\geq$ …. sc,
b) s is completely normalized such that it is a probability distribution, i.e., s1 + s2 + s3 + …. +sc = 1.
c) si is always positive for all items i.

3. PROPOSED ALGORITHM AND STUDY

We have some of the systems already existing in the same field and we have also identified some of the disadvantages in them as follows:

- The popularity for any item is given based on the production of that item. This may not give good result because customers may not have the knowledge of true popularity they needed and depend on the results given by the producer.
- The updates are performed regardless of the true popularity by virtual analysis.

- Producer have to analyse things manually and complexity involves in this. Due to this time consumption may be high.

- The algorithms used in this system may fail to achieve true popularity.

We consider the problem learning of the popularity of items that is assumed to be apriori unknown but has to be learned from the observed user's selection of items. We have selected a mobile market and mobile distribution outlets as our data set and examined them completely in all areas where we can give the list of items suggested by the users and we have made web-application to make an survey at real-time and considered the data given by more that 1000 members of different categories of people and applied our proposed apriori algorithm on the set of data and started suggesting the item in the mobile outlets for the actual users, which had helped the mobile phone companies and also the outlet in-charges. We have implemented the same in some of the mobile outlets in INDIA where we got very good results. The actual goal of the system is to efficiently learn the popularity of items and suggest the popular items to users. This was done to the user to suggest them the mostly used mobiles and their accessories, such that they also buy the best and at the same time the outlet owner will also get benefited. The most important feature in our project is suggesting the users by refreshing the latest results every time the user gives the input and changes his like list.

Now we have overcome many of the disadvantages of the existing systems and achieved many advantages with the proposed algorithm and method as follows:

- In our approach, we consider the problem of ranking the popularity of items and suggesting popular items based on user feedback.

- User feedback is obtained by iteratively presenting a set of suggested items, and users selecting items based on their own preferences either from this suggestion set or from the set of all possible items.

- The goal is to quickly learn the true popularity ranking of items and suggest true popular items.

- In this system better algorithms are used. The algorithms use ranking rules and suggestion rules in order to achieve true popularity.

4. PROPOSED ALGORITHM

APRIORI ALGORITHM:

This is to find frequent item-sets using candidate generation. Apriori employs an iterative approach known as level-wise search, where k-itemsets are used to explore (k+1) itemsets. First, the set of frequent 1-itemsets is found by scanning database to accumulate the count for each item and collecting those items that satisfy minimum support. The resulting set is denoted by L1. Next, L1 is used to find L2,the set of frequent 2-itemsets,which is used to find L3 and so on, until no more frequent k-itemsets can be found. The finding of each Lk requires one full scan of database.

To improve the efficiency of the level-wise generation of frequent itemsets, an important property is called apriori property, which is used to reduce search space.

Apriori property:

- All non empty subsets of a frequent itemset must also be frequent.

- Antimonotone property-if a set cannot pass a test, all of its supersets will fail the same test as well.
  Apriori algorithm is a two step process, the two steps are
  - Join step
  - Prune step

Join step:

To find Lk, a set of candidate k-itemsets is generated by joining Lk-1 with itself. This set of candidates is denoted by Ck. Let l1 and l2 be itemsets in Lk-1.The notation li[j] refers to jth item in li. By convention, Apriori assumes that items within a transaction or itemset are sorted in lexicographic order. For the (k-1) itemset, li, this means that the items are sorted such that li[1]<l1[2]<…….<li[k-1].The join Lk-1∞Lk-1, is performed, where members of Lk-1 are joinable if their first (k-2) items are in common.

Prune step:

Ck is a superset of Lk, that is, its members may or may not be frequent but all of frequent k-itemsets are included in Ck. A scan of database to determine the count of each candidate in Ck would result in the determination of Lk. Ck however can be huge. so this could involve heavy computation. To reduce the size of Ck, Apriori property is used as follows:

Any (k-1) itemset that is not frequent cannot be a subset of a frequent k-itemset. Hence, if any (k-1) subset of a candidate k-itemset is not in Lk-1, then the candidate cannot be frequent either and so can be removed from Ck. This subset testing can be quickly done by maintaining a hash tree of all frequent itemsets.

Improving the efficiency of apriori by proposing it with some variations. Several variations are summarized as follows:

- Hash-based technique (hashing itemsets into corresponding buckets):A hash-based technique can be used to reduce the size of candidate k-itemsets, Ck, for k > 1.

- Transaction reduction (reducing the number of transactions scanned In future iterations):A transaction that doesn't contain any frequent k-itemsets cannot contain any frequent (k+1) itemsets. Therefore, such a transaction can be marked or removed from further consideration because subsequent scans of database for j-itemsets where j > k, will not require it.

- Partitioning(Partitioning the data to find candidate itemsets):A partitioning technique can be used that requires just two database scans to mine the frequent itemsets. It consists of two phases. In phase I ,the algorithm sub divides the transaction of D into n overlapping partitions. If the minimum support threshold for transactions in D is min-sup, then the minimum support count for a partition is min-sup x the number of transactions in that partition. For each partition, all frequent itemsets within the partition are found. These are referred to as local frequent itemsets. In phase II, a second scan of D is conducted in which the actual support of each candidate is accessed in order to determine the global frequent itemsets.

- Sampling: The basic idea of the sampling is to pick a random sample S of given data D and then search for frequent itemsets in S instead of D.

- Dynamic itemset counting: In a dynamic itemset counting technique, new candidate itemsets can be partitioned into blocks by start points. The technique is dynamic in that it estimates the support of all of the itemsets that have been counted so far, adding new candidate itemsets if all of their subsets are estimated to be frequent.

Algorithm:
Apriori:

Find frequent itemsets using an iterative level-wise approach based on candidate generation.

Input: D, a database of transactions; min-sup, the minimum support count threshold.

Output: L, frequent itemsets in D.

Method:

```
(1)  L1=find_frequent_1_itemsets(D);
(2)      for(k=2;Lk-1≠Ø;k++)
(3)      {
(4)          Ck = apriori_gen(Lk-1);
(5)          for each transaction t Є D
(6)          {
(7)           C1=subset(Ck,t);
(8)           for each candidate c Є Ct
(9)                c.count++;
(10) }
(11) Lk= {c Є Ck | c.count ≥ min_sup}
(12) }
(13) return L=UkLk;
```

Procedure apriori_gen (Lk-1 : frequent (k-1)-itemsets)

```
(1)      for each itemset l1 Є Lk-1
(2)          for each itemset l2 Є Lk-1
(3)  if(l1[1]=l2[1])^(l1[2]=l2[2])^…….^(l1[k-2]=l2[k-
     2])^(l1[k-1]<l2[k-1]) then
(4)  {
(5)           c=l1∞l2;
(6)           if (has_infrequent_subsets(c,Lk-1)
     then
(7)                delete c;
(8)           else add c to Ck;
(9)  }
(10)     return Ck;
```

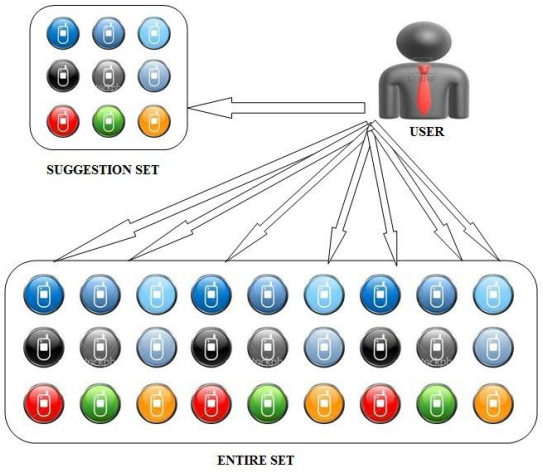
Procedure has_infrequent_subset (c:candidate k-itemset;

Lk-1:frequent(k-1) –itemsets);

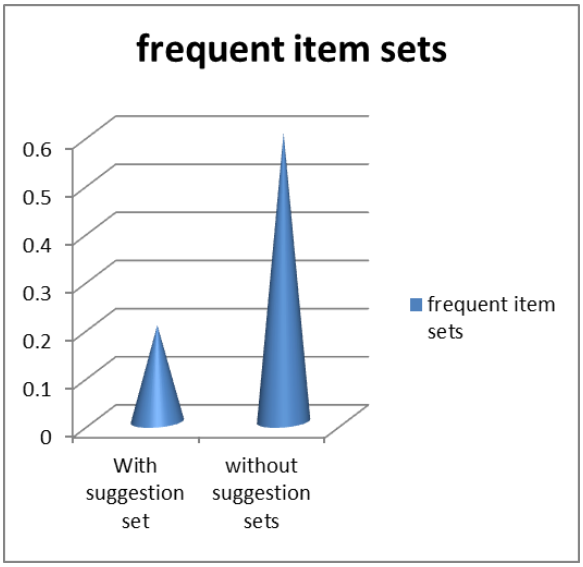
```
(1)      for each (k-1) –subset s of c
(2)           If s Є Lk-1 then
(3)                return TRUE;
(4)  return FALSE;
```

5. RESULTS

The above is the best method of ranking and suggesting the best methods in the scenario of mobile phone outlets in INDIA, which is shown in the following diagram:



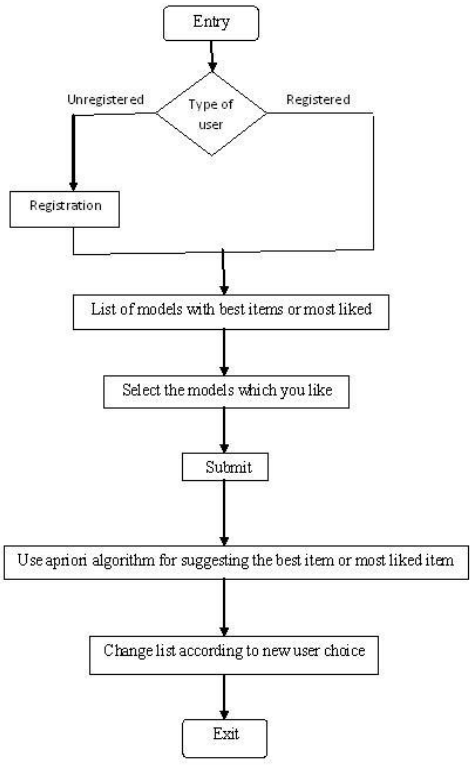
As it was shown in the above diagram we were going to take the most liked items from the users and suggesting the best mobiles or the best set of suggestions that the most of the users liked or ordered.



The confidence of the suggestions were also proved by an traditional confidence calculations as follows

In this section we are going to discuss about algorithms. Till now we have discussed some ranking rules , suggestion rules and Frequency move2set algorithm. We have some problems with these, so we go for an algorithm which suits our requirements well. The algorithm is Apriori algorithm. In order to know these algorithms we need to know some concepts of data mining.

Data flow chart of proposed system:



Frequent itemsets:

Let I={I1, I2, I3,…., Im} be a set of items. Let D, the task-relevant data, be a set of database transactions where each transaction T is a set of items such that T is a subset of I. Each transaction is associated with an identifier, called TID. Let A be a set of items. A transaction T is said to contain A if and only if A is a subset of T. An association rule is an implication of the form $A \geq B$, where A is subset of I, B is subset of I and $A \cap B = \emptyset$. The rule $A \geq B$ holds in the transaction set D with support s, where s is the percentage of transactions in D that contain AUB. This is taken to be the probability ,P(AUB).The rule $A \geq B$ has confidence c in the transaction set D, where c is the percentage of transactions in D containing A that also contain B. This is taken to be the conditional probability, P(B/A). That is,

Support(A=>B) = P(AUB)

Confidence(A=>B) = P(B/A)

Rules that satisfy both a minimum support threshold (min_sup) and a minimum confidence threshold (min_conf) are called strong. The occurrence frequency of an itemset is the number of transactions that contain the itemset. This is also known, simply as the frequency, support count,or count of the itemset. The set of frequent k-itemset is commonly denoted by Lk.

$$\text{confidence}(A \geq B) = P(A / B) = \text{support}(AUB) /$$

$$\text{support}(A) = \text{support}_{\text{count}(AUB)} / \text{support}_{\text{count}(A)}.$$

Mining frequent itemsets:

In general, association rule mining can be viewed as a two-step process:

1. Finding all frequent itemsets: By definition, each of these itemsets will occur at least as frequently as a predetermined minimum support count, min-sup.

2. Generate strong association rules from the frequent itemsets: By definition, these rules must satisfy minimum support and minimum confidence

6. CONCLUSION

All the previous process already proposed were very complex and contains very complicated computations which made the ranking and suggesting the best and popular items have been more and more complex and not getting to the actual end users. Now we have proposed as very simple randomized algorithm for ranking and suggesting popular items designed to account for popularity bias. This was utilized by many of the mobile outlets in the country successfully.

7. REFERENCES

[1] Huidrom Romesh Chandra Singh, T. kalaikumaran, Dr. S. Karthik, Suggestion of True Popular Items, IJCSE, 2010.

[2] Y.Maanasa, V.Kumar, P.Satish Babu, Framework for suggesting POPULAR ITEMS to users by Analyzing Randomized Algorithms, IJCTA, 2011.

[3] V. Anantharam, P. Varaiya, and J. Walrand, "Asymptotically Efficient Allocation Rules for the Multiarmed Bandit Problem with Multiple Plays—Part i: i.i.d. Rewards," IEEE Trans. Automatic Control, vol. 32, no. 11, pp. 968-976, Nov. 1987.

[4] J.R. Anderson, "The Adaptive Nature of Human Categorization" Psychological Rev., vol. 98, no. 3, pp. 409-429, 1991.

[5] Yanbin Ye, Chia-Chu Chiang, A Parallel Apriori Algorithm for Frequent Itemsets Mining, IEEE, 2006.

[6] Cong-Rui Ji, Zhi-Hong Deng, Mining Frequent Ordered Patterns without Candidate Generation.

[7] Huang Chiung-Fen, Tsai Wen-Chih, Chen An-Pin, Application of new Apriori Algorithm MDNC to Exchange Traded Fund, International Conference on Computational Science and Engineering, 2009.

[8] Milan Vojnovi_c, James Cruise, Dinan Gunawardena, and Peter Marbach, Ranking and Suggesting Popular Items, IEEE, 2009.