Location Prediction of Mobile Phone Users using Apriori-based Sequence Mining with Multiple Support Thresholds

Ilkcan Keles, Mert Ozer, Ismail Hakki Toroslu, and Pinar Karagoz

Computer Engineering Department
Middle East Technical University
Ankara, Turkey
{ilkcan,mert.ozer,toroslu,karagoz}@ceng.metu.edu.tr
http://www.ceng.metu.edu.tr

Abstract. Due to the increasing use of mobile phones and their increasing capabilities, high amount of usage and location data can be collected. Location prediction is an important task for mobile phone operators and smart city administration to provide better services and recommendations. In this work, we propose a sequence mining based approach for location prediction of mobile phone users. More specifically, we present a modified Apriori-based sequence mining algorithm for next location prediction, which involves use of multiple support thresholds for different levels of pattern generation. The proposed algorithm involves a new support definition, as well. We have analyzed the behaviour of the algorithm under the change of threshold through experimental evaluation and the experiments indicate improvement in comparison to conventional Apriori-based algorithm.

Keywords: Sequential Pattern Mining, Location Prediction, Mobile Phone Users

1 Introduction

In today's world, mobile phones are commonly used devices, such that almost every person has a mobile phone on average. Intensive amounts of basic usage data including base station, call records and GPS records are stored by large-scale mobile phone operators. This data gives companies ability to build their user's daily movement models and helps them to predict the current location of their users. Using these models and prediction schemes companies can arrange more effective advertisement strategies and the city administrators can determine mass people movement patterns around the city.

Location prediction systems usually make use of sequential pattern mining methods. One common method usually follows two steps; extract frequent sequence patterns and predict accordingly. These methods mostly use Apriori-based algorithms for the phase of extracting sequence patterns. Our approach embraces the idea to use historical movement patterns for current location prediction of a person. However due to the huge size of the Call Data Record (CDR) data, it is impossible to use all historical data to model daily behaviour of people. Rather than using whole patterns contained in the CDR data implicitly, we need to devise a control mechanism over the elimination of sequence patterns. However the flexibility of control that conventional Apriori-based sequence mining algorithms provide is not fully adequate due to the necessity of the balance between the accuracy and space cost in our work. It is a well known fact that when minimum support gets lower, number of patterns extracted increases, thereby size of prediction sets for next location of a person gets larger and accuracy of predictions eventually increases. However the larger number of patterns causes larger space cost. Conventional technique to prevent space cost explosion is to increase minimum support value. Yet this time, it decreases the number of frequent patterns and the size of the prediction sets dramatically, and this causes to miss some interesting patterns in data. To prevent possible space explosion and not to miss valuable information in data we propose a modified version of Apriori-based sequence mining algorithm, that works with multiple minimum support values instead of a global one. To the best of our knowledge, this is the first work which uses different minimum support values at different levels of pruning phases of the conventional algorithm. Normally the number of levels for Apriori-based sequence mining algorithms is not pre-configured. However, in our case, we consider a predefined number of previous steps to predict the next one. Therefore, we can set the number of levels in Apriori search tree. Moreover we slightly change the definition of minimum support in our context. We have experimentally compared the performance of the proposed method involving multiple support thresholds in comparison to that of conventional Apriori-based algorithm that uses only a single minimum support value. The experiments indicate that the proposed approach is more effective to decrease the prediction count and memory requirement.

The rest of this paper is organized as follows. Section 2 introduces previous work on location prediction. Section 3 presents the details of the proposed solution. Section 4 gives the information about evaluation metrics and Section 5 presents experimental results of our prediction method. Section 6 concludes our work and points out possible further studies.

2 Previous Work

In recent years, a variety of modification of the minimum support concept in Apriori-based algorithms have been proposed ([2], [3], [4], [5], [6], [7], [9], [10]) for both association rule mining and location prediction problems. In [2], Han and Fu propose a new approach over the conventional Apriori Algorithm that works with association rules at multiple concept levels rather than single concept level. They first create hierarchical model for items, and iterate levels in a top-down manner by defining a unique minimum support value for each level. In [3], Liu et al., propose a novel technique to the rare item problem. They define a modified concept of minimum support which is a minimum item support having different thresholds for different items.

Since then, there are several of variations of Liu et al.'s work have been proposed ([5],[9], [10]). In [5], Baralis et al. propose that decreasing the minimum support by some fixed constant at each pruning level makes more sense rather than using fixed global minimum item support. In [6], Toroslu and Kantarcioglu introduce a new support parameter named as repetition support to discover cyclically repeated patterns. The new parameter helps them to discover more useful patterns by reducing the number of patterns searched. In [7], Ying et al. propose a location prediction system using both conventional support concept and a score value that is related with semantic trajectory pattern in the candidate elimination phase. The score depends on the geographic interpretation of the movement.

Most of the multiple minimum support concept is based on the rare itemset problem. To the best of our knowledge, this is the first work which uses different minimum support values at the different levels of pruning phases of conventional algorithm. In our previous work on location prediction with sequence mining [8], we broadened the conventional pattern matching nature of sequence mining techniques with some relaxation parameters. In this work, we use some of these parameters introduced in [8].

3 Proposed Technique

3.1 Preliminaries

In this work we utilized the CDR data of one of the largest mobile phone operators of Turkey. The data corresponds to an area of roughly 25000 km^2 with a population around 5 million. Almost 70% of this population is concentrated in a large urban area of approximately 1/3 of the whole region. The rest of the region contains some mid-sized and small towns and large rural area with very low population. The CDR data contains roughly 1 million user's log records for a period of 1 month. For each user there are 30 records per day on average. The whole area contains more than 13000 base stations. The records in CDR data contain anonymized phone numbers (of caller and callee or SMS sender and receiver), the base station id of the caller (sender), the time of the operation.

The unnecessary attributes in CDR data, such as city code, phone number etc., are filtered out and date and time information are merged into a single attribute which is used to sort data in temporal order. After sorting, we created sequences of fixed-length corresponding to user's daily movement behavior.

A *sequence* is an ordered list of locations which is expressed as s<i..j>, where i is the starting location and j is the last location in the sequence. A sequence of length k is called *k-sequence.*

In Apriori-based sequence mining, the search space can be represented as a hash tree. A *path* in the tree is a sequence of nodes such that each node is the prefix of the path until the root and, for each node, its predecessor is the node's parent. p<a..b> expresses a path starting with node a and ending with node b.

We say that a path p is equal to a sequence s, denoted by $p = s$, if the length of path p and sequence of s are equal and there is one to one correspondence between the locations of s and the nodes of p.

3.2 Apriori-based Sequence Mining Algorithm with Multiple Support Thresholds (ASMAMS)

In this section, we introduce our algorithm and define related concepts. To build a model which aims to predict the next location of the user, we used a recursive hash tree based algorithm namely Apriori-based Sequence Mining Algorithm with Multiple Support Thresholds (ASMAMS). This algorithm constructs level based models i.e. hash trees whose nodes contain corresponding base station id and frequency count of the sequence corresponding to the path up to this node.

The main novelty of the algorithm in comparison to the conventional algorithm is the level based support mechanism with a new level-based support definition. In contrast to previous approaches that aim to extract all frequent sequences, in this work, we focus on predicting the next item in a sequence. Therefore, we defined a level-based support in order to keep track of the relations between levels. Conventionally, support of a given sequence pattern is defined as the ratio of the number of the sequences containing the pattern to the number of all sequences in the dataset. If the support threshold is lowered, accuracy increases, but space requirement increases as well. On the hand hand, higher threshold values lead to lower space requirement, yet lower accuracy. In order to keep the balance between accuracy and space requirement, in ASMAMS, support of an n-sequence is defined as the ratio of the count of a given sequence s to the count of the parent sequence with length *(n-1)*.

$$support(s) = \frac{\#\ of\ occurrences\ of\ the\ sequence\ s\ with\ length\ n}{\#\ of\ occurrences\ of\ prefix\ of\ sequence\ s\ with\ length\ (n-1)} \tag{1}$$

Before going into the details of the algorithm, we need to define the following four parameters that will be used in the algorithm.

– *levelCount:* It is the height of the hash tree to be constructed. It also states how many previous base station ids to be used in prediction i.e. levelCount - 1.
– *currentLevel:* It denotes the current level throughout the construction of the hash tree.
– *supportList:* It denotes a list of minimum support parameters for each level. Its length is equivalent to levelCount.
– *sequence data set:* A set of fixed-length location id sequences derived from CDR data after preprocessing.
– *tree:* A hash tree where each node stores the location id and the count of sequence represented by a path from root to this node and it is also connected to its children via a hash structure. The root of the tree represents an empty sequence and named as *root*. It is both input and output of this algorithm.

Algorithm 1 ASMAMS

Input: *sequences*,*levelCount*,*supportList*,*currentLevel* ← 1
Output: *tree*

```
 1: function buildModel($sequences$, $levelCount$, $currentLevel$, $supportList$, $tree$)
 2:     $constructTree(sequences, tree, currentLevel)$
 3:     $pruneTree(tree, currentLevel, supportList[currentLevel])$
 4:     if $currentLevel \neq levelCount$ then
 5:         $buildModel(levelCount, currentLevel + 1, supportList, tree)$
 6:     end if
 7: end function
```

ASMAMS algorithm is basically for model construction. As given in Algorithm 1, this task is divided into two phases: tree construction and pruning.

In the construction phase, the data is read sequentially, and new level nodes are added to the corresponding tree nodes. For instance, assume that we are constructing the fourth level of the tree and we have <1,2,3,4> as the sequence. If <1,2,3> corresponds to a path in the input tree, 4 is added as a leaf node as the prefix of this path with count 1. If we encounter the same sequence, the algorithm only increments the count of this node. If the current tree does not contain <1,2,3>, then it is not added to the tree. The construction algorithm is given in Algorithm 2.

Algorithm 2 Construct Tree Algorithm

Input: *sequences*, *tree*, *currentLevel*
Output: *tree*

```
 1: function constructTree($sequences$, $tree$, $currentLevel$)
 2:     for all $s<l_1..l_{currentLevel}> \in sequences$ do
 3:         if $\exists p<root..leaf> \in tree$ s.t $p = s$ then
 4:             $leaf.count = leaf.count + 1$
 5:         else
 6:             if $\exists p<root..leaf> \in tree$ s.t $p = s<l_1..l_{currentLevel-1}>$ then
 7:                 $insert(tree, leaf, l_{currentLevel})$ //add $l_{currentLevel}$ as a child of $leaf$
 8:                 $l_{currentLevel}.count = 1$
 9:             end if
10:         end if
11:     end for
12: end function
```

In the pruning phase, constructed model and the corresponding minimum support value are taken as parameters. In this phase, initially we calculate leaf nodes' support values. If it is below the minimum support value, it is removed from tree, otherwise no action is taken. The detailed algorithm of pruning phase can be found in Algorithm 3.

Algorithm 3 Pruning Algorithm

```
Input: $tree$, $currentLevel$ $minimumSupport$
Output: $tree$
1: function PRUNETREE($tree$, $currentLevel$ $minimumSupport$)
2:     for all $leaf \in tree$ s.t. $depth(leaf) = currentLevel$ do
3:         $support \leftarrow leaf.count/parent.count$
4:         if $support < minimumSupport$ then
5:             delete $leaf$
6:         end if
7:     end for
8: end function
```

3.3 Prediction

In the prediction phase, we use the hash tree constructed by ASMAMS to predict user's next location. The prediction algorithm takes a sequence with length of (levelCount - 1) as input and returns a list of predicted locations.

The algorithm firstly checks whether the constructed tree contains the given sequence. If it contains, children nodes of the corresponding node (at the level (levelCount - 1)) constitute the prediction set. If it does not contain the given sequence, it returns an empty list indicating the model can not predict a location for the given sequence.

Algorithm 4 Prediction Algorithm

```
Input: $sequence$, $tree$
Output: $predictionSet$
1: function PREDICT($sequence$, $tree$)
2:     if $\exists p{<}root..node{>} \in tree$ s.t $p = sequence{<}l_1..l_{currentLevel-1}{>}$ then
3:         $predictionSet \leftarrow node.children$
4:     end if
5: end function
```

3.4 Running Example

In this example, we set level count to 5 and minimum support list to [1/6, 1/2, 1/2, 2/3, 0] and we used the sample sequences shown in the Table 1.

In the first level, the data is traversed sequentially and the first location ids in the sequences are added to the hash tree together with their counts. Then in the pruning phase, their support values are calculated and nodes 2 and 3 are pruned since their support fall below the given minimum support 1/6. In the second level, 2-sequences are added to the hash tree with their counts. After support values are found, the nodes <5,6>, <5,8> and <5,11> are pruned since

id	sequence	-	id	sequence
1	<1, 2, 3, 4, 5>		7	<4, 7, 11, 12, 15>
2	<1, 2, 3, 4, 6>		8	<4, 7, 11, 10, 9>
3	<1, 2, 3, 4, 5>		9	<5, 6, 11, 10, 9>
4	<2, 3, 4, 7, 8>		10	<5, 8, 9, 10, 11>
5	<3, 4, 7, 9, 10>		11	<5, 11, 10, 9, 4>
6	<4, 7, 11, 12, 13>		12	<1, 2, 3, 4, 5>

Table 1: Example Sequences

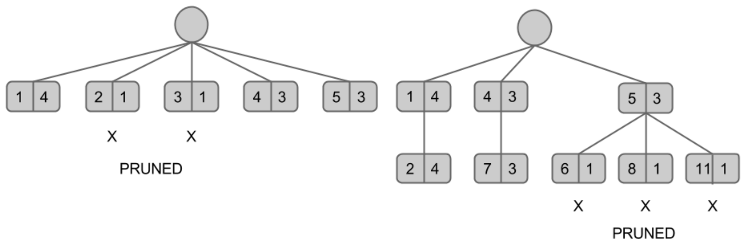


Fig. 1: Hash tree at the end of the first level (left), Hash tree at the end of the second level (right)

their support values are 1/3 and falls below the given minimum support 1/2. The resulting hash trees can be seen in Figure 1.

In the third level, 3-sequences are added to the hash tree. None of the nodes are pruned in this level, since the support values are all 1. In the fourth level, after 4-sequences are added to the hash tree, the node <4,7,11,10> is pruned as it does not have the required support. In the final level (which is the last level of the hash tree), 5-sequences are added to the hash tree. Since the minimum support value for this level is 0, there is no pruning. The resulting hash tree can be seen in Figure 2.

If the hash tree constructed by ASMAMS in Figure 2 is applied on sequence <1,2,3,4>, the algorithm will return 5 and 6 as the output, since these are the children nodes of path <1,2,3,4>.

4 Evaluation

For the experimental evaluation, CDR data obtained from one of the largest mobile phone operators in Turkey has been used. A quick analysis shows that around 76% of the users next location is their current location. We take this value as the baseline for our experiments. For evaluation, we extract sequences from raw CDR data set and try to predict the last element of the sequence using the previous ones. After trying several lengths, we have determined that 5-sequences (i.e., using a given 4-sequence, try to predict the next element of

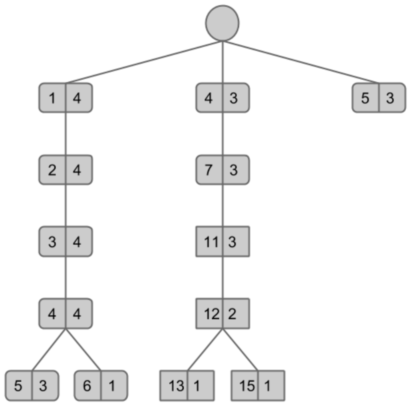


Fig. 2: Hash tree at the end of the final level

the sequence) produces the highest accuracy values. Therefore, we have used 5-sequences extracted from data set, both for training and testing, by using k-fold cross validation in order to assess the quality of predictions made. As training phase, we run ASMAMS on fixed length sequences to build the sequence tree. At the testing phase, for each test set sequence Algorithm 4 introduced in the section 3.5 has been applied and the result of the prediction is compared against the actual last element of the test set sequence. These results are used in the calculations of the evaluation metrics which are introduced below.

Accuracy metric is used for evaluating the number of correctly predicted test set sequences. It simply can be defined as the ratio of true predicted test sequences to the total number of test sequences. However, for some test cases, there may be no relevant path in the tree for test sequence which means either no such training sequence is come up or it is removed from the tree in one of the pruning phases. The first accuracy metric, g-accuracy (general accuracy), is the ratio of number of correctly predicted test sequences to the number of all test sequences. The second one, p-accuracy (predictions' accuracy), is the ratio of the number of correctly predicted test sequences to the number of all test sequences able to be predicted. In the first form of accuracy calculation, the accuracy result superficially drops for cases that no prediction is able to be performed. These accuracy measures have been described in more detail in our earlier work [8].

Memory Requirement metric measures the relative peak RAM requirement during the algorithm's execution. All memory requirement values are projected to the range [0-100], where 100 represents the maximum memory utilization.

Prediction Count metric is used to evaluate average size of the prediction set in correctly predicted test sequences.

Score is introduced since there are 4 different parameters that we want to optimize. It is used for evaluating general performance of our model by combining above metrics into a single one. This metric is only used to determine the parameters for the optimal model, which is explained in the section. It is defined as a weighted sum of *g-accuracy*, *p-accuracy*, *memory requirement*(mem_req) and *prediction count*(pred_count) in Equation 2.

$$Score = w_1 * \textit{g-accuracy} + w_2 * \textit{p-accuracy} + w_3 * \textit{(100-mem\_req)} + w_4 * \textit{(100-pred\_count)} \quad (2)$$

Considering the importance of the parameters the weights are set as follows; w1 = 0.6, w2 = 0.1, w3 = 0.1 and w4 = 0.2.

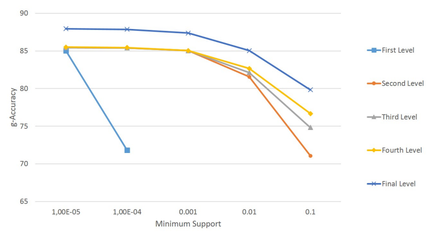
5 Experimental Results

For the experiments we have used 5-sequences (i.e. level count in Algorithm 2 is set to 5), after trying longer and shorter sequences. While shorter sequences, such as 4-sequences or 3-sequences, were superficially increasing prediction count, longer sequences, such as 6-sequences, were decreasing g-accuracy sharply, even though p-accuracy was increasing, since the number of predictable sequences was quickly decreasing. Therefore, 5-sequences seemed as the best length for the data in hand, and shorter or longer sequences' results were not useful.

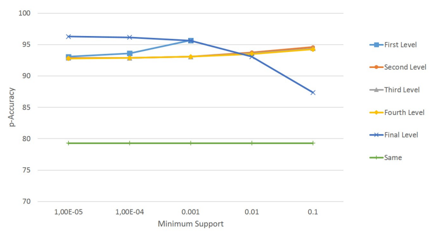
After determining the sequence length and level count for experiments, we first narrow down our search space by setting our support values to [10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}] respectively for each level (including the final level as the last one) as the best support list for our data. We have used the score parameter introduced above to determine this "best" sequence. Then we have tried all possible decreasing combinations as list of support parameters. For every level, we fixed other levels' support values to the support values of the best model and we present results of changing this level's minimum support value according to evaluation metrics. Same percentage value refers to the ratio of being in the same location as previous location and is included in the figure to show the improvement provided by ASMAMS.

In a set of experiments, we have analyzed the effect of the minimum support parameter for all levels. For each level, 5 different minimum support values are used in the experiments, which are 10^{-5},10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}. Meanwhile, for the other levels, minimum support values are fixed as optimal values, such as 10^{-5},10^{-3}, 10^{-3}, 10^{-3}, 10^{-2} respectively.

As it can be seen from the Figure 3 for all levels g-accuracy drops as the minimum support increases. However, this drop is much sharper in the first level. Although, p-accuracy also shows the same trend in the first level, it shows slight increase in intermediate levels, and then, there is also a small drop in the final level. Figure 3 also shows the percentages of locations which are exactly the same as the previous ones for all the experiments as well. Our p-accuracy results show that, the correct prediction (of p-accuracy) can be increased even above 95% with our model.

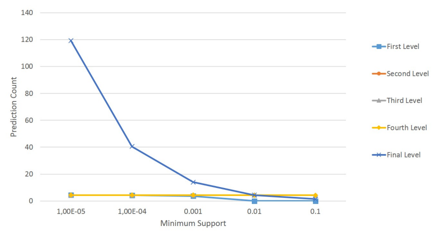


(a)

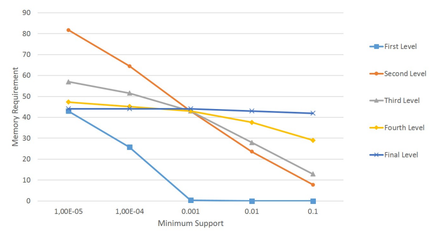


(b)

Fig. 3: Change of g-Accuracy & p-Accuracy



(a)



(b)

Fig. 4: Change of Prediction Count & Memory Requirement

Similar trends can be observed for the prediction count parameter. Sharp drops occur in the first level as the minimum support value increases. However, for intermediate levels these drops are almost negligible. Again, in the final level prediction count decreases much faster also. Figure 4 shows that the prediction count values are at acceptable levels.

The amount of the drop in the memory requirement as the minimum support value increases slows down with the increase of the levels. In the final level, there is almost no drop in the memory requirement. Especially in the first level since most sequences are pruned with high minimum support requirement, the memory requirement drops very quickly.

In addition to above mentioned experiments, we have also applied standard AprioriAll algorithm [1]. The main drawback of AprioriAll algorithm is the size of the prediction set. In order to obtain high accuracy results (g-accuracy) as in our model, the minimum support value must be chosen as a very small value (even zero), so that we can keep as much sequences as possible. However, this results in high prediction count as well as increasing the memory requirement. The accuracy obtained when no minimum support value is given is the upper bound that can be achieved with sequence matching approach. However, for that setting the memory requirement is also the maximum, since the hash-tree keeps all sequences without any pruning. As expected, this maximum accuracy can be obtained only with a very high prediction count, which is more than 133. Since this is unacceptably high, we tested AprioriAll with a non-zero, but very small minimum support value. This resulted slight decrease in accuracy, while dropping the prediction count and the memory requirement significantly with pruning of large portion of hash-tree. Even though the memory requirement has dropped a lot to a very good level, the decreased value of prediction count still stayed unacceptably high value, which is almost 40. Further increases in minimum support values had dropped the accuracy levels to around and below baseline levels. Therefore, they are not acceptable either. However, with ASMAMS we have achieved almost the same accuracy levels of the best and optimal AprioriAll accuracy values with a very low prediction count value, which is 4.43, with a memory requirement less than the half of the optimal (and maximal) results of AprioriAll setting. These results are summarized in Table 2.

G-Accuracy	P-Accuracy	Mem. Req.	Pred. Count	Description
51.47	88.66	0.01	1.29	ApprioriAll Min. Sup: 1e-5
85.04	93.08	44	4.43	ASMAMS Min. Sup. List: [1e-5.1e-3.1e-3.1e-3.1e-2]
86.32	94.15	9.76	39.42	ApprioriAll Min. Sup: 1e-8
89.82	95.38	100	133.48	ApprioriAll Min. Sup: 0

Table 2: The results for ASMAMS and AprioriAll methods

6 Conclusion

In this work, we present an Apriori-based sequence mining algorithm for next location prediction of mobile phone users. The basic novelty of the proposed algorithm is a new, level-based support definition and the use of multiple support thresholds, each for different levels of pattern generation that corresponds to generation of sequence patterns of different lengths. The evaluation of the method is conducted on CDR data of one of the largest mobile phone operators in Turkey. The experiments compare the performance of the proposed method in terms of accuracy, prediction count and space requirement under changing thresholds for each level. Actually, these experiments serve for determination of the best minimum support list for each level to obtain the highest accuracies, as well. We have also compared the performance with conventional method involving a single support threshold. We have observed that our method ASMAMS produces almost the optimal accuracy results with very small prediction sets, whereas the same accuracy can be obtained by AprioriAll with very low support thresholds and much larger prediction sets. Considering that there are more than 13000 different locations, the prediction sets' sizes obtained by ASMAMS with almost optimal accuracy can be considered as quite useful result for the mobile phone operator.

As the future work, we aim to extend this study by adding a region based hierarchy to this model in order to increase prediction accuracy.

Acknowledgements. This research was supported by Ministry of Science, Industry and Technology of Turkey with project number 01256.STZ.2012-1 and title "Predicting Mobile Phone Users' Movement Profiles".

References

1. Agrawal R, Srikant R. Mining sequential patterns. In: Proc of Int Conf Data Engineering. Taipei: IEEE Computer Society, pp. 3–14. (1995)
2. Han, J., Fu, Y.: Discovery of Multiple-Level Association Rules from Large Databases. In: Proc. VLDB '95, pp. 420–431. (1995)
3. Liu, B., Hsu, W., Ma, Y.: Mining association rules with multiple minimum supports. In: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '99). ACM, New York, NY, USA, pp. 337–341. (1999)
4. Li, H., Chen, S., Li, J., Wang, S., Fu, Y.: An improved multi-support Apriori algorithm under the fuzzy item association condition. In: International Conference on Electronics, Communications and Control (ICECC '11), pp. 3539–3542, 9-11 Sept. (2011)
5. Baralis, E., Garza, P.: A Lazy Approach to Pruning Classification Rules. In: Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM '02). IEEE Computer Society, Washington, DC, USA, 35-. (2002)
6. Toroslu, H., Kantarcioglu M.: Mining Cyclically Repeated Patterns. In: Springer Lecture Notes in Computer Science 2114, p. 83. (2001)
7. Ying, J., J., Lee, W., Weng, T., Tseng, V. S.: Semantic trajectory mining for location prediction. In: Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS '11), Divyakant Agrawal, Isabel Cruz, Christian S. Jensen, Eyal Ofek, and Egemen Tanin (Eds.). ACM, New York, NY, USA, 34-43. (2011)
8. Ozer, M., Keles, I., Toroslu, H., Karagoz, P.: Predicting the change of location of mobile phone users. In: Proceedings of the Second ACM SIGSPATIAL International Workshop on Mobile Geographic Information Systems (MobiGIS '13), Chi-Yin Chow and Shashi Shekhar (Eds.). ACM, New York, NY, USA, 43-50. (2013)
9. Hu, Y., Chen, Y.: Mining association rules with multiple minimum supports: a new mining algorithm and a support tuning mechanism. In Decision Support Systems 42, pp. 1-24 (2006)
10. Lin, W., Tseng, M.: Mining Generalized Association Rules with Multiple Minimum Supports. In DaWaK 2001, LNCS 2114, pp. 11-20. (2001)