Location prediction within the mobility data analysis environment Daedalus

Fabio Pinelli[1], Anna Monreale[1 2], Roberto Trasarti[1 2],Fosca Giannotti[1]
Pisa KDD Laboratory
[1]ISTI - CNR, Area della Ricerca di Pisa,
Via Giuseppe Moruzzi, 1 - 56124 Pisa, Italy
[2]Computer Science Dep., University of Pisa,
Largo Pontecorvo, 3 - 56127 Pisa, Italy

ABSTRACT

In this paper we propose a method to predict the next location of a moving object based on two recent results in GeoPKDD project: DAEDALUS, a mobility data analysis environment and *Trajectory Pattern*, a sequential pattern mining algorithm with temporal annotation integrated in DAEDALUS. The first one is a DMQL environment for moving objects, where both data and patterns can be represented. The second one extracts movement patterns as sequences of movements between locations with typical travel times.

This paper proposes a prediction method which uses the local models extracted by *Trajectory Pattern* to build a global model called *Prediction Tree*. The future location of a moving object is predicted visiting the tree and calculating the best matching function.

The integration within DAEDALUS system supports an interactive construction of the predictor on the top of a set of spatio-temporal patterns.

Others proposals in literature base the definition of prediction methods for future location of a moving object on previously extracted frequent patterns. They use the recent history of movements of the object itself and often use time only to order the events. Our work uses the movements of all moving objects in a certain area to learn a classifier built on the mined trajectory patterns, which are intrinsically equipped with temporal information.

1. INTRODUCTION

In the last years, we have witnessed a considerable increase of the number of mobile devices used from the people and an extensive use of wireless communication, such as Bluetooth, Wi-fi and GPRS. Usually, the mobile devices are equipped with positioning sensors that utilize Global Positioning System (GPS) to accurately provide the location of a device. Therefore, nowadays, the movement of people or vehicles within a given area can be observed from the digital traces left behind by the personal or vehicular mobile devices, and collected by the wireless network infrastructures. For instance, mobile phones leave positioning logs, which specify their localization, or cell, at each moment they are connected to the GSM network; analogously, GPS-equipped portable devices can record their latitude-longitude position at each moment they are exposed to a GPS satellite, and transmit their trajectories to a collecting server. The pervasiveness of ubiquitous technologies guarantees that there will be an increasing availability of large amounts of data pertaining to individual trajectories with more localization precision.

The knowledge of the mobile objects positions fosters the deployment of location-based services and applications, which need to know the approximate position of a mobile user in order to provide some functionality. Examples of such services are navigational service, management of traffic and location-based advertising. In a typical scenario, a moving object periodically informs the positioning framework of its current location. Due to the unreliable nature of mobile devices and some limitations of the positioning systems, the location of a mobile object is often unknown for a long period of time. In such case, a method to predict the possible next location of a moving object is required.

In this paper, we introduce a data mining approach to solve the problem of predicting the next location of a moving object. In literature there are different proposals which address this problem. They base the definition of prediction methods for future location on previously extracted frequent patterns, use the recent history of movements of the object itself and often use time only to order the events. In our work, instead, we use the movements of all moving objects in a certain area to learn a classifier built on the mined trajectory patterns, which are intrinsically equipped with temporal information. Moreover, this work is based on two recent results in GeoPKDD[4] project: DAEDALUS[9], a mobility data analysis environment and *Trajectory Pattern*[3], a sequential pattern mining algorithm with temporal annotation integrated in DAEDALUS.

Now, we summarize the four main steps of our approach:

Data Selection : thanks to the Moving Object Data Mining query language provided by DAEDALUS, we can select a portion of data using spatio-temporal primitives.

Digital Object Identifier: 10.4108/ICST.MOBIQUITOUS2008.3894
http://dx.doi.org/10.4108/ICST.MOBIQUITOUS2008.3894

Local Model Building : *Trajectory Pattern* algorithm is executed to extract the movement frequent patterns called T-patterns, which are concise representations of interesting behaviors of moving objects.

Global Model Building : using the local model extracted in the previous step a global model called *Prediction tree* is built.

Prediction : the global model is used to predict the future location of a moving object. To this aim the algorithm uses a concept of distance defined in Section 3.2 to find the best matching pattern and to predict the next movement location.

The rest of the paper is organized as follows. In Section 2 we present some basic notion used in the rest of the paper. In Section 3 we propose our method for the solution. In particular, we describe how to build the prediction tree and present our prediction algorithm. Some experimental results are given in Section 4. In Section 5 we present some related works. The paper concludes in Section 6 with a brief summary of this preliminary work and highlights which are the more promising lines of research in order to better tackle the object of this work.

2. BACKGROUND

In this section we introduce some basic concepts and terminology useful in the rest of the paper. In particular, we review *Trajectory Pattern* algorithm and the DAEDALUS environment, which are used in our approach.

2.1 T-Pattern

During the past half decade, attempts have been made to extend many techniques for knowledge discovery in classical relational or transactional data to knowledge discovery in the context of movement data [8]. Typical techniques adapted to the spatio-temporal context are association rule mining, frequent pattern discovery, clustering, classification, prediction and time-series analysis.
Most approaches tend to define clustering algorithms, which group together moving object trajectories using some notion of trajectory similarity, typically distance-based. When searching for concise representations of interesting behaviors of moving objects, we define local patterns, that are patterns that aim at characterize small portions of the data space. T-Pattern is an example introduced in [3]. The authors develop an extension of the sequential pattern mining paradigm, introduced in [2], which analyzes the trajectories of moving objects. A trajectory of a moving object is a sequence of time-stamped locations, representing the traces collected by some wireless/mobility infrastructure (GSM, GPS, etc). The location is abstracted by using ordinary Cartesian coordinates, as formally stated by the following definition:

DEFINITION 1. *A **Trajectory** or spatio-temporal sequence is a sequence of triples*

$$T =< x_0, y_0, t_0 >, \ldots, < x_n, y_n, t_n >$$

where t_i $(i = 0 \ldots n)$ denotes a timestamp such that $\forall_{0<i<n}$ $t_i < t_{i+1}$ and (x_i, y_i) are points in $\mathbf{R}^2$.

Intuitively, each triple $< x_i, y_i, t_i >$ indicates the object is in the position (x_i, y_i) at time t_i.
Trajectory Pattern is an efficient algorithm to extract a set of frequent temporally-annotated sequences of dense spatial regions extracted from trajectories with respect to two thresholds σ and τ: the former represents the minimum support and the latter a temporal tolerance. The threshold σ denotes the minimum support as well as in the standard frequent sequential pattern algorithms. In the case of *Trajectory Pattern*, it is also used as spatial density threshold, in fact a region is dense if at least σ distinct trajectories cross it. The parameter τ introduces a tolerance for each time value: everyone is represented as a (hyper)cube of side 2τ in R^n_+.
Each T-pattern extracted is a concise description of frequent behaviors, in terms of both space (i.e., the regions of space visited during movements) and time (i.e., the duration of movements).
As an example, consider the following T-Pattern over regions of interest in the center of a town:

$$RailwayStation \xrightarrow{15min} CastleSquare \xrightarrow{50min} Museum$$

$$RailwayStation \xrightarrow{10min} MiddleBridge \xrightarrow{10min} Campus$$

Intuitively, people typically move between railway station to Castle Square in 15 minutes and then to Museum in 50 minutes. Another typical sequence is that travel time from Railway station to Middle Bridge is 10 minutes and then in another 10 minutes people move to Campus. Now we recall the T-patterns definition introduced in [3]:

DEFINITION 2. *A T-pattern, is a pair (S, A), where $S =< R_0, \ldots, R_n >$ is a sequence of regions, and $A = \alpha_1, \ldots \alpha_n \in R^k_+$ is the (temporal) annotation of the sequence. A T-pattern is also represented as $(S, A) = R_0 \xrightarrow{\alpha_1} R_1 \xrightarrow{\alpha_2} \ldots \xrightarrow{\alpha_n} R_n$.*

As explained in [2], the extraction of a representative transition time as annotation is formalized as density estimation problem. Therefore the resulting set of temporal annotations of a sequence is a set of temporal intervals and everyone represents a side of a dense (hyper)cube. A visual example of T-pattern is shown in Fig.1.

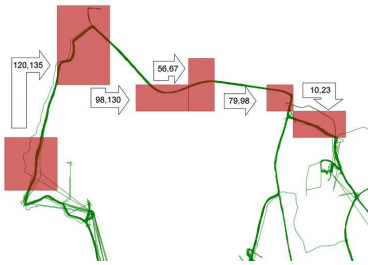


Figure 1: An Example of T-Pattern

Digital Object Identifier: 10.4108/ICST.MOBIQUITOUS2008.3894
http://dx.doi.org/10.4108/ICST.MOBIQUITOUS2008.3894

2.2 Daedalus Framework

DAEDALUS is an environment aimed at supporting the user in specifying and refining mining objectives as well as combining multiple strategies in the specific context of movement data. More specifically, the process of analyzing moving object data can be seen as an interaction between the data mining engine and the end user, where the latter formulates a query describing the patterns of his/her interest and the former returns the required patterns, by exploiting suitable mining algorithms [9]. To represent the moving object DAEDALUS system adopts a Moving Objects Database, namely Hermes [10, 11] as Object-Relational database layer where storing both data and extracted patterns. As a consequence, source raw trajectories are stored in Hermes exploiting the Moving objects capabilities that the DBMS provides. Furthermore, mined models may be stored once a suitable datatype has been defined.
The DAEDALUS system query processing architecture is depicted in Fig. 2.

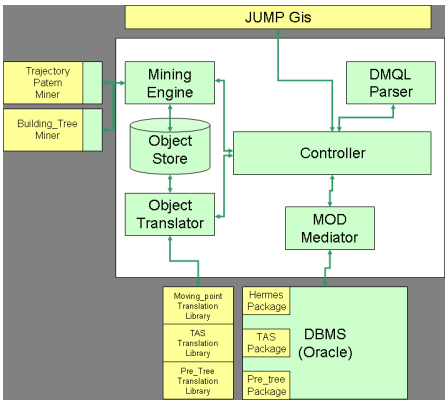


Figure 2: Architecture of DAEDALUS system

The use of the *Trajectory Pattern* miner algorithm can be easily integrated in the DAEDALUS framework. In fact, many factors positively concur to the usage of it. It allows us to select a set of trajectories, to perform the *Trajectory Pattern* miner and finally see the results in a easy way. An example of this is the following query:

```
CREATE MODEL T_Patterns As MINE TPattern_Mining
FROM Trajectories
WHERE TPattern_Mining.density = 0.02 AND
      TPattern_Mining.snr = 4.5 AND
      TPattern_Mining.tt = 50
      TPattern_Mining.type = semi_static
```

The above definition statement exemplifies the discovery of a knowledge discovery task, aimed at unveiling a collection of T-patterns, from the primary data source `Trajectories`, where at least 2% of the overall number of moving objects fall within the individual T-pattern and points of different trajectories that fall in a same T-pattern are within a radius of 4.5 spatial units. Moreover, the maximum threshold of temporal approximation amounts to 50 time units. The last parameter means all the regions don't overlap and are calculated in the first step of the algorithm.

3. PROBLEM DEFINITION

In our work, we collect a dataset of trajectories that describes the movements of a mobile objects collection. Using the spatial-temporal primitives provided by DAEDALUS we select a portion of these trajectories and next we use the *Trajectory Pattern* mining algorithm to extract a set of T-patterns with respect to a temporal threshold τ and a minimum support σ. We assume that the regions extracted by this algorithm are not overlapped.
After the extraction of the T-patterns, the next steps of our method are: building the global model and predicting the next location. In the following we describe these last two steps.

3.1 Prediction Tree

The set of movement sequential patterns, mined by *Trajectory Pattern*, represents a local model. Now we show how to build a global model, called *Prediction Tree*, using the local models extracted.
The *Prediction Tree* is a more compact data structure than a simple list of T-patterns, which helps us searching the best usable pattern for the prediction. It is defined as a triple $\mathcal{T} = (N, E, Root(\mathcal{T}))$, where N is a finite set of nodes, E is a set of labeled edges and $Root(\mathcal{T}) \in N$ is a fictitious node and represents the root of the tree. Each edge belonging to E is labeled with a time interval int. The triple $(u, v, int) \in E$ denotes the edge labeled with the time interval int between the parent node u and the child node v. The time interval int has the form $[time_{min}, time_{max}]$. The edges which link the root node to its child nodes are the only edges of the tree labeled with an empty time interval, denoted by int_ϵ. Each node of the tree (except the root) has exactly one parent and it can be reached through a path, which is a sequence of labeled edges starting with the root node. An example of path is the following:

$$\mathcal{P} = (Root(\mathcal{T}), a, int_\epsilon), (a, b, int_1), (b, c, int_2)(c, d, int_3).$$

Each node $v \in N$, except $Root(\mathcal{T})$, contains entries of the form $< id, region, s >$, where:

- id is the identifier of the node v
- $region$ represents a region of a T-pattern
- if v is the final node of the path
  $\mathcal{P} = (Root(\mathcal{T}), a, int_\epsilon), (a, b, int_1), \ldots, (u, v, int_k)$
  then s is the support of the T-pattern represented by the path $\mathcal{P}$.

Given a path $\mathcal{P}$, if $(u, v, [time_{min}, time_{max}])$ is a edge of $\mathcal{P}$, then $[time_{min}, time_{max}]$ intuitively represents the travel time interval of the transition from the region represented by u to the region represented by v.
The Algorithm 1, presented in the following, describes how to build the *Prediction Tree* given a set of T-patterns. Before describing the Algorithm 1 we introduce the notion of prefix of a T-pattern.

DEFINITION 3. *Let (S, A) and (S', A') be two T-patterns such that $(S, A) = R_0 \xrightarrow{\alpha_1} R_1 \xrightarrow{\alpha_2} \ldots \xrightarrow{\alpha_n} R_n$ and $(S', A') = R_0 \xrightarrow{\beta_1} R_1 \xrightarrow{\beta_2} \ldots \xrightarrow{\beta_k} R_k$. (S', A') is a prefix of (S, A) if and only if $k \leq n$ and $\forall i = 1 \ldots k$ α_i is included in β_i.*

Digital Object Identifier: 10.4108/ICST.MOBIQUITOUS2008.3894
http://dx.doi.org/10.4108/ICST.MOBIQUITOUS2008.3894

The *PredictionTreeBuilding* algorithm has been integrated in DAEDALUS. The input of this algorithm is a set of T-patterns T_p_Set. For each T-pattern we denote by $T_p[j].area$ the jth region of the T-pattern and by $T_p[j].interval$ the jth time interval, that is the travel time interval to reach the region $T_p[j].area$ from the previous region. Each T-pattern, belonging to the set of T-patterns, is inserted in the prediction tree. Given a T-pattern T_p, if the root node does not have a child with the region equal to the first region of T_p, then we add a new child to the root. Next, we append a branch which represents the rest of the elements of T_p to this new node. Otherwise, if the root node has a child with the region equal to the first region of T_p then we search in the sub-tree of this node the path which corresponds to the longer prefix of the T-pattern T_p. Next, we append a branch which represents the rest of the elements of T_p to this path.

Algorithm 1 PredictionTreeBuilding

Require: Set of T-patterns T_p_Set
Ensure: prediction tree $\mathcal{T}$

```
 1: for all T_p in T_p_Set do
 2:    curNode = null
 3:    for all i in Root_Childs do
 4:       if T_p[1].area == Root_Childs[i] then
 5:          curNode = Root_Childs[i]
 6:    if curNode == null then
 7:       Root(T).addChlid(new Node(T_p[1].area, null))
 8:    for all j in T_p do
 9:       node = findChild(curNode, T_p[j].area, T_p[j].interval)
10:       if node! = null then
11:          curNode = node;
12:       else
13:          node = newNode(T_p[j].area, T_p[j].interval)
14:          curNode.addChild(node)
15:          curNode = node
16: return T
```

The $findChild(curNode, T_p[j].area, T_p[j].interval)$ procedure returns the child node of $curNode$ that has the region equal to $T_p[j].area$ and a greater or equal travel time interval than $T_p[j].interval$. The $addChild(newNode)$ procedure appends the node $newNode$ to a node of the tree. It is important to know that the support of each node is set to the highest support among the supports of the T-patterns which are represented by the paths that cross this node.
Now, we show a little example of construction of a tree having a small set of T-patterns. We try to show some interesting features which are useful for understanding how we achieve the problem.

EXAMPLE 1. *We present an example of prediction tree built on a set of T-patterns. Consider the following T-patterns:*

```
C (10,12) D supp:35        A (4,20) A supp:26
A (70,90) C supp:21        A (134,150) D supp:25
A (9,15) B supp:31         A (9,12) B (10,56) E supp:21
B (8,70) E supp:28         F (2,51) D supp:37
```

where, for simplicity, the different labels represent different regions. In fig.3 we show the tree prediction obtained by executing ***PredictionTreeBuilding*** *algorithm (Algorithm 1) over these T-patterns. It is important to note that in some cases one or more T-patterns can be grouped together. For instance the following path of the prediction tree:*

$(Root, < 2, A, 31 >, t_e),$
$(< 2, A, 31 >, < 9, B, 31 >, [9, 15]),$
$(< 9, B, 31 >, < 10, E, 21 >, [10, 56])$

represents both the T-patterns:

```
A (9,15) B supp:31        A (9,12) B (10,56) E supp:21
```

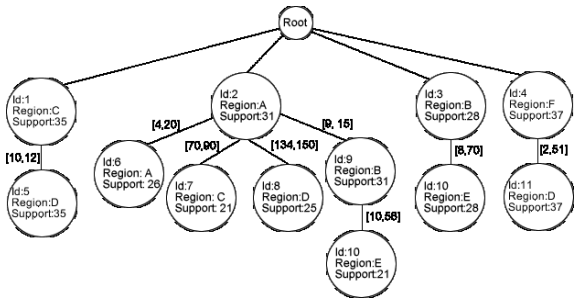


Figure 3: The prediction tree built on example 1

3.2 Mobility Prediction

In this section we show how we predict the next locations for a given moving object in space and in time using the *prediction tree* and the tools provided by DAEDALUS.
The main idea behind our prediction is to find the best pattern that matches a given trajectory. Hence for a given trajectory we compute the best score matching among all admissible paths of the *prediction tree*. Given the best node its children are selected as the next possible locations. In the following definition we present the score function we use to compute the best matching.

DEFINITION 4 (SCORE). *Given a trajectory tr and the region j of a certain path p of the prediction tree , we define punctual score as:*

$$p\_score_{tr,(p,j)} = \frac{\sigma_R}{Distance(R, seg_{t_e})+1}$$

where R is the jth region selected of the path p, seg_{t_e} is the trajectory segment described by the moving object in the time interval t_e, σ_R is the support of the relative t-pattern taken into account and the distance is the minimum spatial distance from the seg_{t_e} and the region R.

We define the score function as:

$$score_{(tr,p)} = \frac{\sum_1^j p\_score_{tr,(p,j)}}{j}$$

where the p_score is the punctual score computed along all the specified path of the prediction tree.

In the Definition 4 we compute the *score* for a node u as the *punctual score* average of nodes belonging to the path which reaches u. In this way we give the same weight to all nodes of a path. It is important to note that the *p_score* is equal to the support σ_R when the segment seg_{t_e} intersects the region R.
The proposed mobility prediction algorithm (Algorithm 2) recursively compares in space and in time all the paths in the

Digital Object Identifier: 10.4108/ICST.MOBIQUITOUS2008.3894
http://dx.doi.org/10.4108/ICST.MOBIQUITOUS2008.3894

Algorithm 2 Prediction

Require: A prediction tree pt, A trajectory $traj$
Ensure: the prediction $pred$

1: $pred.score = 0$
2: $pred.bestNode = pt.root$
3: $pred.predicted_regions\_Set = \{\}$
4: **for all** node n in $pt.root.children$ **do**
5: $pred =$**Visit**$(n, 0, pred, NULL, traj)$
6: **return** $pred$

Procedure Visit

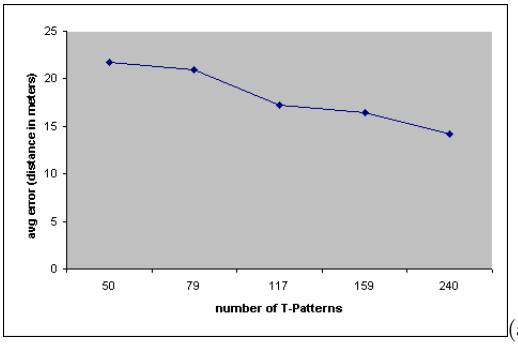
Require: A node $node$, the parent score $parentScore$, current prediction $pred$, time interval $inter$, the trajectory $traj$
Ensure: the new prediction new_pred

1: **if** $node$ is $leaf$ **then**
2: **return** $pred$
3: **if** $inter$ is not $null$ **then**
4: $seg =$**MoveInTime**$(traj, inter.min, inter.max)$
5: **else**
6: $seg = traj$
7: $startPoint =$**GetEnterPoint**$(seg, node.region)$
8: $startTime =$**GetTime**$(seg, startPoint)$
9: **if** $segment$ is $null$ **then**
10: **return** $pred$
11: $total = parentScore +$**PScore**$(node.supp, seg, node.area))$
12: $current\_score = total/$**depth**$(node)$
13: $new\_pred =$**UpdatePred**$(current\_score, pred)$
14: **for all** node n in $node.children$ **do**
15: $new\_pred =$
16: **Visit**$(n,$
17: $total, new\_pred,$
18: $node.getIntervalsToChild(n),$
19: **MoveInTime**$(traj, startTime, \infty))$
20: **return** new_pred

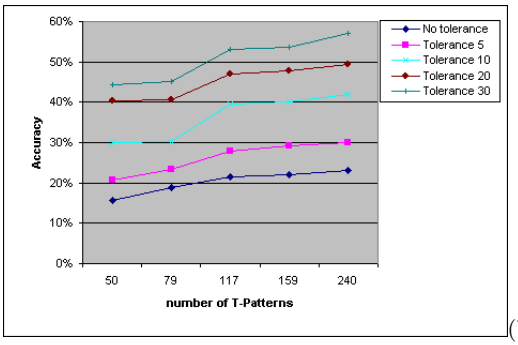
prediction tree with a given trajectory using the *Visit* procedure. For each node, the algorithm computes the *score* for it. If this node has a better *score* value than the previous best score then it becomes the new predictor and its children, which are not reached from the trajectory (in time), are the predicted locations. It is immediate to notice that we don't consider a leaf as possible predictor and for this reason we don't compute the score of this type of node.
Now, we describe some functions used in the Algorithm 2. The *MoveInTime* function extracts the segment of trajectory included in the bounds of an interval of time (seg_{t_e} in the definition 4). The *GetEnterPoint* function instead allows to locate either the point where the segment enters in a specified region or the nearest point of the segment to such region. Finally, the *GetTime* function returns the timestamp when the segment is in a given point.

4. EXPERIMENTS & RESULTS

In this section we present some preliminary tests we have done on Milan dataset provided by GeoPKDD project. In our experiments we have used a set of 2500 trajectories and from this we have constructed a training set of 2150 trajectories and a test set of 350 trajectories. From the training set we have extracted the T-patterns using *Trajectory Pattern* with a temporal threshold equal to 100 seconds and a resolution of the regions of 10 meters. The only parameter we have changed to obtain a different number of T-patterns is the minimum support threshold.
We have used two performance measures for the evaluation of the proposed algorithm. The first one is the *prediction error* which is measured as average of the spatial distance between a predicted location and the real position in a specified time interval. The second one is the *accuracy* value which is the rate of the correctly predicted locations divided by the total number of requests.
In Fig. 4(a) is possible to see that the prediction error decreases when the number of T-patterns increase. The Fig. 4(b) describes the effect of increasing number of T-patterns on the accuracy considering an increasing tolerance value defined as the maximum acceptable spatial error for a prediction.



(a)



(b)

Figure 4: (a) Prediction error; (b) Accuracy

It is important to notice that there are a lot of parameters available in *Trajectory Pattern* that can strongly affect the model and the results. This is part of our future work.

5. RELATED WORKS

During the past years, in the spatio-temporal database context the interest in modeling techniques to predict future moving object locations has increased.
There are different works [13, 7, 6] where frequent patterns and association rules are used to solve the problem of predicting future locations. In [13] the authors propose a data mining algorithm to predict user movements in a mobile computing system. This algorithm is based on three steps: mining the mobility patterns of users, forming mobility rules from these patterns, and finally predicting a mobile user's

Digital Object Identifier: 10.4108/ICST.MOBIQUITOUS2008.3894
http://dx.doi.org/10.4108/ICST.MOBIQUITOUS2008.3894

next movements by using the mobility rules.
In [7] Morzy presents a data mining approach which discovers frequent trajectories and store them in a *FP-Tree*, generates movement rules from each frequent trajectory and then matches the trajectory of a moving object with the database of movement rules to predict the next location. Moreover, three different matching strategies are proposed.
The work in [6] introduces a new method for predicting the location of a moving object. This method uses the past trajectory of the object and combines it with the movement rules discovered in the moving objects database. In order to generate movement rules a modified version of *Apriori* algorithm, called *AprioriTraj*, is used. In addition, the author introduces and analyzes four matching strategies.
Another work about this topic is [5], where Jeung et al. present a novel approach which forecasts future locations of an object in a hybrid manner, that is, using not only motion function but also objects' movement patterns. They modify the *Apriori* algorithm to generate trajectory patterns and then introduce the trajectory pattern tree for indexing patterns discovered. They also propose the hybrid prediction algorithm which provide accurate results for both distant time queries and non-distant time queries.
Another interesting proposal is presented in [12], where the authors use time-series analysis with travel speed simulation to predict future trajectories.
Frequent patterns are also used to build classification model. An interesting approach is presented in [1]. This work proposes a framework of frequent pattern-based classification which includes three steps: feature generation, feature selection and model learning. In the first step, frequent patterns are generated with a specified minimum support. In the second step, feature selection is applied on frequent pattern and finally, a classification model is built from frequent patterns selected.

6. CONCLUSION AND FUTURE WORK

In this paper we presented a novel method to predict the next location of a moving object. This approach uses *Trajectory Pattern* to mine the frequent sequential patterns from a set of trajectories. Furthermore, we introduced the definition of a new data structure called *Prediction Tree*, which represents a global model obtained from the T-patterns extracted. Then we described how we can predict the future location by visiting this tree and calculating a best matching function. This work is based on two important results obtained in GeoPKDD project: DAEDALUS environment and *Trajectory Pattern* algorithm. It is important to notice our prediction definition of future location of a moving object is based on the previous movements of all moving objects in a certain area. Moreover, previously proposed methods usually use the temporal information only to order the events, while we use the T-patterns which are intrinsically equipped with temporal information. Finally, we presented the results of our preliminary experiments. There are different open issues we want to explore in the future:

- **Drop assumption of overlapping regions**: considering a more general set of T-pattern.
- **New score functions**: providing a better score function or a set of functions that can improve the prediction accuracy of the model.
- **Knot**: considering the knot of a trajectory on region during the process give us the possibility of having a better prediction.
- **Optimization**: improving the efficiency of the code give us the possibility to test the algorithm on a very large set of T-Patterns and trajectories.
- **Case of study**: trying our approach in a real case of study using other dataset provided by GeoPKDD project.

7. REFERENCES

[1] H. Cheng, X. Yan, J. Han, and C.-W. Hsu. Discriminative frequent pattern analysis for effective classification. In *ICDE*, pages 716–725. IEEE, 2007.

[2] F. Giannotti, M. Nanni, and D. Pedreschi. Efficient mining of temporally annotated sequences. In J. Ghosh, D. Lambert, D. B. Skillicorn, and J. Srivastava, editors, *SDM*. SIAM, 2006.

[3] F. Giannotti, M. Nanni, F. Pinelli, and D. Pedreschi. Trajectory pattern mining. In P. Berkhin, R. Caruana, and X. Wu, editors, *KDD*, pages 330–339. ACM, 2007.

[4] F. Giannotti, D. Pedreschi, and et al. Geopkdd: Geographic privacy-aware knowledge discovery and delivery (european project), 2008.

[5] H. Jeung, Q. Liu, H. T. Shen, and X. Zhou. A hybrid prediction model for moving objects. In *ICDE*, pages 70–79. IEEE, 2008.

[6] M. Morzy. Prediction of moving object location based on frequent trajectories. In A. Levi, E. Savas, H. Yenigün, S. Balcisoy, and Y. Saygin, editors, *ISCIS*, volume 4263 of *LNCS*, pages 583–592. Springer, 2006.

[7] M. Morzy. Mining frequent trajectories of moving objects for location prediction. In P. Perner, editor, *MLDM*, volume 4571 of *LNCS*, pages 667–680. Springer, 2007.

[8] M. Nanni, B. Kuijpers, C. Korner, M. May, and D. Pedreschi. Spatiotemporal data mining. In F. Giannotti and D. Pedreschi, editors, *Mobility, Data Mining, and Privacy: Geographic Knoweledge Discovery*. Springer-Verlag, 2008.

[9] R. Ortale, E. Ritacco, N. Pelekis, R. Trasarti, G. Costa, F. Giannotti, G. Manco, and C. Renso. Trajectory pattern mining. SEBD, 2008.

[10] N. Pelekis, E. Frentzos, N. Giatrakos, and Y. Theodoridis. HERMES: Aggregative lbs via a trajectory db engine. In *Proceedings of ACM SIGMOD Conference*, Vancouver, Canada, 2008.

[11] N. Pelekis, Y. Theodoridis, S. Vosinakis, and T. Panayiotopoulos. Hermes - a framework for location-based data management. In Y. E. Ioannidis, M. H. Scholl, J. W. Schmidt, F. Matthes, M. Hatzopoulos, K. Böhm, A. Kemper, T. Grust, and C. Böhm, editors, *EDBT*, volume 3896 of *LNCS*, pages 1130–1134. Springer, 2006.

[12] B. Xu and O. Wolfson. Time-series prediction with applications to traffic and moving objects databases. In *MobiDE*, pages 56–60. ACM, 2003.

[13] G. Yavas, D. Katsaros, Ö. Ulusoy, and Y. Manolopoulos. A data mining approach for location prediction in mobile environments. *Data Knowl. Eng.*, 54(2):121–146, 2005.

Digital Object Identifier: 10.4108/ICST.MOBIQUITOUS2008.3894
http://dx.doi.org/10.4108/ICST.MOBIQUITOUS2008.3894