WhereNext: A location predictor on trajectory pattern mining

Conference Paper · June 2009

DOI: 10.1145/1557019.1557091 · Source: DBLP

4 authors, including:

Anna Monreale
Università di Pisa
46 PUBLICATIONS **617** CITATIONS
SEE PROFILE

Fabio Pinelli
IBM, Research
35 PUBLICATIONS **1,032** CITATIONS
SEE PROFILE

Fosca Giannotti
Italian National Research Council
221 PUBLICATIONS **3,677** CITATIONS
SEE PROFILE

WhereNext: a Location Predictor on Trajectory Pattern Mining

Anna Monreale
Dept. Computer Science
University of Pisa, Italy
ISTI - CNR Pisa, Italy
annam@di.unipi.it

Fabio Pinelli
ISTI - CNR Pisa, Italy
fabio.pinelli@isti.cnr.it

Roberto Trasarti
Dept. Computer Science
University of Pisa, Italy
ISTI - CNR Pisa, Italy
roberto.trasarti@isti.cnr.it

Fosca Giannotti
ISTI - CNR Pisa, Italy
fosca.giannotti@isti.cnr.it

ABSTRACT

The pervasiveness of mobile devices and location based services is leading to an increasing volume of mobility data. This side effect provides the opportunity for innovative methods that analyse the behaviors of movements. In this paper we propose *WhereNext*, which is a method aimed at predicting with a certain level of accuracy the next location of a moving object. The prediction uses previously extracted movement patterns named *Trajectory Patterns*, which are a concise representation of behaviors of moving objects as sequences of regions frequently visited with a typical travel time. A decision tree, named *T-pattern Tree*, is built and evaluated with a formal training and test process. The tree is learned from the *Trajectory Patterns* that hold a certain area and it may be used as a predictor of the next location of a new trajectory finding the best matching path in the tree. Three different best matching methods to classify a new moving object are proposed and their impact on the quality of prediction is studied extensively. Using *Trajectory Patterns* as predictive rules has the following implications: (I) the learning depends on the movement of all available objects in a certain area instead of on the individual history of an object; (II) the prediction tree intrinsically contains the spatio-temporal properties that have emerged from the data and this allows us to define matching methods that striclty depend on the properties of such movements. In addition, we propose a set of other measures, that evaluate a priori the predictive power of a set of *Trajectory Patterns*. This measures were tuned on a real life case study. Finally, an exhaustive set of experiments and results on the real dataset are presented.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Data mining

General Terms

Algorithms

Keywords

Trajectory patterns, Spatio-temporal data mining

1. INTRODUCTION

The last few years, have witnessed a considerable increase in the number of mobile devices along with the use of wireless communication, such as Bluetooth, Wi-fi and GPRS. Such mobile devices are often equipped with positioning sensors that utilize Global Positioning Systems (GPS) to accurately provide the location of a device. This means that 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, whenever they are connected to the GSM network. Likewise GPS-equipped portable devices can record their latitude-longitude position and transmit their trajectories to a collecting server. The pervasiveness of such ubiquitous technologies guarantees that there will be an increasing availability of large amounts of data on individual trajectories, with increasing precision in term of localization.
Knowledge about the positions of mobile objects has led to location-based services and applications, which need to know the approximate position of a mobile user in order to operate. Examples of such services are navigational services, traffic management 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 the limitations of the positioning systems, the location of a mobile object is often unknown for a long period of time. In such cases, a method to predict the possible next location of a moving object is required in order to anticipate or pre-fetch possible services in the next location. Several proposals in the literature use only the history of movements of the individual object on the basis of which future location is simply guessed. The innovative aspect of our work, on the other hand, is in the use of the movements of all objects in a certain area to learn a classifier. Our basic assumption is that people often follow the crowd: individuals tend to follow common paths.

For example, people go to work every day by similar routes, and public transport crosses similar routes in different time periods. Thus, if we have enough data to model typical behaviors, we can use such knowledge to predict the future movement of most individuals: at least those whose past movements are fairly typical.
Our method uses the movement patterns previously extracted by using the *Trajectory Pattern* algorithm developed in [2]. The *Trajectory Pattern* algorithm mines movement patterns as sequences of regions where typical travel times are frequently followed.
The four main steps of our approach are as follows:

Data Selection : by using spatio-temporal primitives, we select a spatial area and a time period, in order to take only the portion of trajectories crossing that area in that time period.

Local Models Extraction : the *Trajectory Pattern* algorithm is executed to extract from the selected trajectories the frequent movement patterns called *Trajectory Patterns* (hereafter T-patterns). The discriminative power of T-patterns is measured against different evaluation functions which consider spatial coverage, dataset coverage, and region separation.

T-pattern Tree Building : the extracted local models, T-patterns, are combined in a prefix tree called *T-pattern Tree*. The nodes of the tree are regions frequently visited and the edges represent travel among regions and are annotated with the typical travel time. Each common prefix of T-patterns becomes common path on the tree. This tree may be viewed as a global model of the underlying mobility data, once augmented with a default path which represents all infrequent trajectories. This method is similar to the use of association rules as predictive rules in rule based classifiers.

Prediction : The *T-pattern Tree* is used to predict the future location of a moving object. To do this, the algorithm uses a concept of distance defined in Section 4.2 to find the best matching pattern and to predict the next movement location. We propose three different best matching methods to classify a new moving object and their impact on the accuracy of prediction has been studied extensively, in relation to variations of the spatio-temporal window during the matching process.

The performance of the classifier is evaluated against a real dataset of 17000 cars equipped with GPS. The results of the experiments show that the prediction is of a good quality when high spatio-temporal precision is required. Clearly this is holds for a specific set of moving objects. In fact the precision decreases as the number of moving objects for which a prediction is possible grows.
The rest of the paper is organized as follows. In Section 2 we review the most recent literature regarding location prediction methods. In Section 3 we introduce the *Trajectory Pattern* mining method. Section 4 is the core of the paper, where the innovative aspects of our work are presented. We describe how to build the *T-pattern Tree* and present our prediction algorithm. In Section 5 a method to evaluate the strength of prediction of the extracted patterns before building the tree is presented. Some experimental results are presented in Section 6. Section 7 concludes the paper and highlights the most promising future lines of research.

2. RELATED WORK

There are several studies that address the problem of predicting future locations. Most of them use a model based on frequent patterns and association rules and define a trajectory as an ordered sequence of locations, where the time is used as a time-stamp [8, 9, 13, 5]. Moreover, some of these approaches try to predict the next location of a moving object by using the movements of all the moving objects in a database [8, 9], while others base the prediction only on the movement history of the object itself [13, 5]. In [8, 9] Morzy introduces a new method for predicting the location of a moving object. In particular, he extracts the association rules from the moving object database. Then, given a trajectory of a moving object, using matching functions he selects the best association rule that matches this trajectory, and then uses it for the prediction. In [8] Morzy uses a modified version of *Apriori* algorithm to generate association rules, and in [9] he uses a modified version of *PrefixSpan* algorithm. All the matching functions in these papers are based on the notions of support and confidence and do not consider any notion of spatial and temporal distance.
In [13] the authors propose a method to predict user movements in a mobile computing system. This algorithm is based on three steps: mining the mobility patterns of an individual user, forming association rules from these patterns, and finally predicting a mobile user's next movements by using these rules. In order to select the rule used for the prediction, the authors consider the notions of support and confidence. The support of each candidate is computed by a distance based on the notion of string alignment. Jeung et al. in [5] present an innovative approach which forecasts future locations of an object in a hybrid manner. They combine predefined motion functions with the movement patterns of the object, extracted by a modified version of the *Apriori* algorithm. The motion functions are linear or non-linear models that capture the object's movements by sophisticated mathematical formulas and are an input of the method decided by the analyst.
In [12] the authors use a time-series analysis with travel speed simulations to predict future trajectories. They use a process based on range querying with spatial-temporal constraints on a moving object database. The prediction is represented by the locations that satisfy the spatio-temporal constraints.
Frequent patterns are also used to build a classification model. An interesting approach based on a framework of frequent pattern-based classification is presented in [1]. There are essentially 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 patterns and finally, a classification model is built from frequent patterns selected.

3. TRAJECTORY PATTERN MINING

Over the last five years, 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 [10]. Some of the typical techniques adapted to the spatio-temporal context are association rule mining, frequent pattern discovery, clustering, classification, prediction, and time-series analysis.

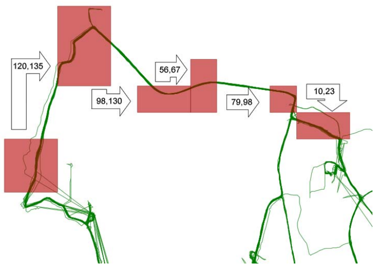


Figure 1: An Example of T-pattern

Most approaches tend to define clustering algorithms, which group together moving object trajectories using some notion of trajectory similarity, which is typically distance-based. When searching for concise representations of interesting behaviors of moving objects, we define local patterns. These are patterns that aim at characterize small portions of the data space. T-pattern was originally introduced in [3], where the authors developed 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 that 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 a *Trajectory Pattern*, it is also used as a spatial density threshold. In fact, the algorithm discretizes the working space through a regular grid with cells of a user-set size. Then the density of each cell is computed by considering each single trajectory and incrementing the density of all the cells that contain any of its points. Finally, a set of the most frequent regions is extracted by means of a simple heuristics considering only the cells with a density greater than σ. 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:

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

Intuitively, people typically move between the railway station to Castle Square in 15 minutes and then to the Museum in 50 minutes. Now we recall the T-patterns definition introduced in [3]:

Algorithm 1: PrefixTreeBuilding($\mathcal{T}_p\_Set$)

```
Input: A Set of T-patterns $\mathcal{T}_p\_Set$
Output: A T-pattern Tree $\mathcal{PT}$
$\mathcal{PT}$ = new T-pattern Tree();
foreach $T_p$ in $\mathcal{T}_p\_Set$ do
    node = Root($\mathcal{PT}$);
    foreach $(i, r)$ in $\mathcal{T}_p$ do
        (edge, n) = findChild(node,r);
        if $\nexists$ n $\vee$ notIncluded(edge.interval,i) then
            v = new Node(r);
            v.support = $T_p$.supp;
            node.appendChild(v,i);
            node = v;
        else
            UpdateInterval(edge,i);
            UpdateSupport(n,$T_p$.supp);
            node = childNode;
        end
    end
end
return $\mathcal{PT}$
```

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 an annotation is formalized as a density estimation problem. Therefore the resulting set of temporal annotations of a sequence is a set of temporal intervals, each of which represents one side of a dense (hyper)cube. An example of a T-pattern is shown in Fig.1.

4. NEXT LOCATION PREDICTION

The goal of the prediction method *WhereNext* is as follows: given a database of trajectories D construct a predictive model WN_D using the set of T-patterns extracted by D; given a new trajectory T use WN_D to predict the next location of T.
WhereNext consists of several steps. First of all, we select the set of interesting trajectories, namely those crossing a specific area and a specific temporal window defined by a spatio-temporal query. In the second step, the *Trajectory Pattern* mining algorithm is applied to the selected trajectories and a set of T-patterns is extracted with respect to a temporal threshold τ and a minimum support σ. Different settings of the two parameters τ and σ return different collection of T-patterns. In order to choose the best T-pattern collection from the extracted ones their predictive power is measured following an evaluation function defined in Section 5. Finally, the selected collection of extracted T-patterns is used to build the predictive model. Below we describe these steps in detail.

4.1 T-pattern Tree

The use of association rules is a consolidated method to build classifiers. The prediction phase consists in matching the "tuple" to be classified against the body of the rule. If a matching is found, the tuple is classified according to the value of the head [7, 6]. Following this direction, the first problem in our case is how to generate $Body \Rightarrow Head$ rules from a T-pattern of the form $R_0 \xrightarrow{\alpha_1} R_1 \xrightarrow{\alpha_2} \ldots \xrightarrow{\alpha_n} R_n$. Clearly, many possible $Body \Rightarrow Head$ rules can be derived from a T-pattern. For example, from $R_0 \xrightarrow{\alpha_1} R_1 \xrightarrow{\alpha_2} R_2 \xrightarrow{\alpha_3} R_3$ we can generate the rules

$$R_0 \xrightarrow{\alpha_1} R_1 \xrightarrow{\alpha_2} R_2 \Rightarrow^{\alpha_3} R_3 \qquad (1)$$
$$R_0 \xrightarrow{\alpha_1} R_1 \Rightarrow^{\alpha_2} R_2 \xrightarrow{\alpha_3} R_3 \qquad (2)$$

Thus, the predicted next location of the trajectory $(l_0, t_0) \rightarrow (l_1, t_1) \rightarrow (l_2, t_2)$ is the region R_3 if some spatio-temporal matching holds with the body of the rule (1), while the predicted next location of the trajectory $(l_0, t_0) \rightarrow (l_1, t_1)$ is R_2 if some spatio-temporal matching holds with the body of the rule (2).
To avoid generating several rules from each T-pattern and to make the prediction phase efficent we adopted a prefix tree, named *T-pattern Tree*, to compactly represent a collection of T-patterns. Hereafter the path of a *T-pattern Tree* indicates a rule.

T-pattern Tree Definition.

The *T-pattern Tree* is a prefix tree and is defined as a triple $\mathcal{PT} = (N, E, Root(\mathcal{PT}))$, where N is a finite set of nodes, E is a set of labeled edges and $Root(\mathcal{PT}) \in N$ is a fictitious node, that 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 taht 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 a path for the node c (denoted by $\mathcal{P}(c, \mathcal{PT})$) is:

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

Each node $v \in N$, except $Root(\mathcal{PT})$, contains entries of the form $\langle id, region, support, children \rangle$, where:

- id is the identifier of the node v
- $region$ represents a region of a T-pattern
- $support$ is the support of the T-pattern represented by the path $\mathcal{P}(v, \mathcal{PT})$
- $children$ is the list of child nodes of v.

Given a path $\mathcal{P}$, if $(u, v, [time_{min}, time_{max}])$ is an edge of $\mathcal{P}$, then $[time_{min}, time_{max}]$ intuitively represents the travel time interval of the transition from the region of the node u to the region of v.

T-pattern Tree Construction.

The *PrefixTreeBuilding* algorithm (see Algorithm 1) describes how to build the *T-pattern Tree* given a set of T-patterns T_p_Set. The following definition introduces the notion of prefix of a T-pattern.

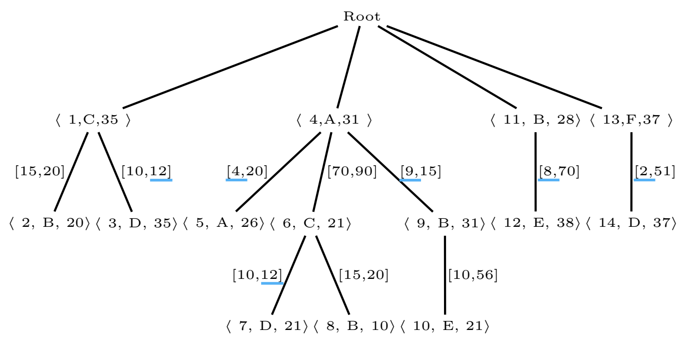


Figure 2: T-pattern Tree construction

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.*

In order to simplify the description of the algorithm we consider that a T-pattern is a sequence of pairs $\langle interval, region \rangle$. Specifically, given the sequence $\langle i_1, r_1 \rangle \ldots \langle i_n, r_n \rangle$ we use i_k to denote the interval of time to reach the region r_k from r_{k-1}. In this representation the first interval i_1 is fictitious and equal to $[0, \infty]$.
Each T-pattern belonging to T_p_Set, is inserted into the *T-pattern Tree*. Intuitively, given a T-pattern T_p, in the tree we search for the path that corresponds to the longest prefix of T_p. Next, we append a branch, which represents the rest of the elements of T_p in this path. A T_p is appended to a path in the tree if this tree is a prefix of T_p.
The $findChild(node, r)$ function returns the child of $node$ that has the region equal to r and a connection edge. The $UpdateInterval(edge, i)$ and $UpdateSupport(n, T_p.supp)$ procedures update the label of $edge$, if the interval i is larger than the interval of $edge$, and the support of n, if its support is smaller than $T_p.supp$.

Example 1. *We present an example of T-pattern Tree built on a set of T-patterns. Consider the following T-patterns:*

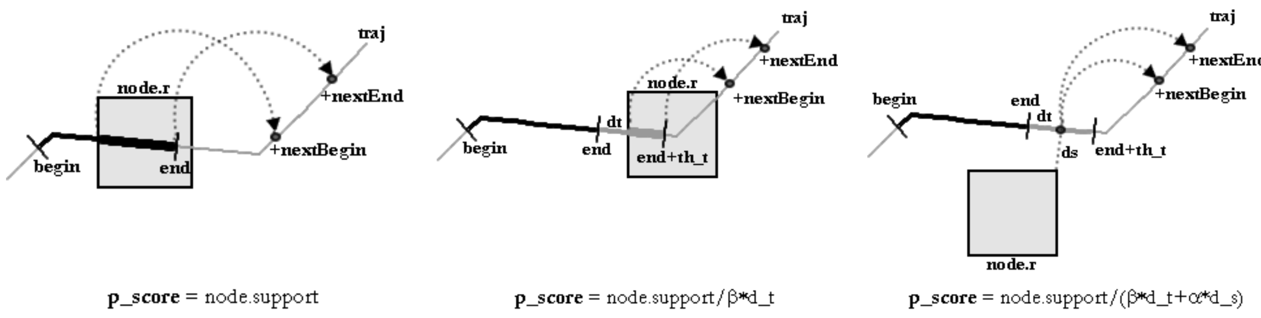
```
<(),C> <(15,20),B> supp:20              <(),C> <(10,12),D> supp:35
<(),A> <(4,20),A> supp:26               <(),A> <(70,90),C> supp:21
<(),A> <(9,12),C> <(10,12),D> supp:21   <(),F> <(2,51),D> supp:37
<(),A> <(9,12),B> <(10,56),E> supp:21   <(),A> <(9,15),B> supp:31
<(),A> <(9,12),C> <(15,20),B> supp:10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

where, capital letters represent different region. Figure 2 shows the corresponding T-pattern Tree. Notice that a path may group together several T-patterns. For instance, the following path of the T-pattern Tree:

$$(Root, \langle 4, A, 31 \rangle, t_\epsilon),$$
$$(\langle 4, A, 31 \rangle, \langle 9, B, 31 \rangle, [9, 15]),$$
$$(\langle 9, B, 31 \rangle, \langle 10, E, 21 \rangle, [10, 56])$$

represents both the T-patterns:

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



(a) (b) (c)

Figure 3: The three possible cases during the punctual score computation.

4.2 Prediction Strategy

In this section we show how to predict the next locations for a given moving object in space and time using the *T-pattern Tree*. The main idea behind our prediction is to find the best path on the tree, namely the best T-pattern that matches the given trajectory. Hence, for a given trajectory we compute the best matching score of all the admissible paths of the *T-pattern Tree*. The children of the best node that produces a prediction are selected as next possible locations. Before computing the score of a whole path we compute a local score for each node of this path: we call it *punctual score*.

4.2.1 Punctual Score

The punctual score $pScore_r$ indicates the goodness of a node w.r.t. a trajectory. It measures the level of reachability of a node r by a trajectory T that has already reached the parent node $r-1$. Considering the moving object that follows the trajectory T, we introduce the notation of $WhereNext_{r-1}$ to identify a spatio-temporal window where the moving object *will be* after the time interval specified in the edge towards r. For example, the Figure 3 the segment [begin,end] is the $WhereNext_{r-1}$ while [nextBegin, nextEnd] identifies the $WhereNext_r$. We then define the punctual score of r according to the spatio-temporal distance between $WhereNext_{r-1}$ and the spatial region of the node r specified by the following three different possible cases:

Case a : the $WhereNext_{r-1}$ intersects the region of the node r. This is the optimal case and the punctual score is equal to the support value of the node. Fig.3(a)

Case b : the $WhereNext_{r-1}$ enlarged by temporal tolerance th_t intersects the region of the node r. In this case, the punctual score is the support value of the node divided by the minimum time distance between the intersection point and the region. Fig.3(b).

Case c : the $WhereNext_{r-1}$ enlarged by temporal tolerance th_t does not intersect the region of the node r. In this case the punctual score is the support value of the node divided by the weighted sum of the spatial and temporal distances between the region and the nearest neighbor point of the enlarged window. Figure 3(c)

Here th_t is a parameter that defines the tolerance for the time window during the process. If $WhereNext_r$ does not exist when the trajectory ends, the punctual score is not defined.

```
Algorithm 2: Prediction($\mathcal{PT}$, traj, $th$)
  Input: A T-pattern Tree $\mathcal{PT}$, A Trajectory $traj$, A
         configuration $th$
  Output: A set of predictions $\{\langle location, score\rangle\}$
  $candidate$ = Visit($\mathcal{PT}$,$traj$, $th$);
  $best$ = findBest($candidate$);
  return $predictions$;
```

4.2.2 Path Score

The overall score of a path is based on the value of the punctual score for each node of the path. We studied three different functions:

Definition 4 (Score functions). *Given a trajectory tr, a path $P = [p_1, ..., p_n]$ and a punctual score $pScore_k$ defined on each $p_k \in P$, we define:*

- $avgScore_{(tr,P)} = \frac{\sum_1^n pScore_k}{n}$
- $sumScore_{(tr,P)} = \sum_1^n pScore_k$
- $maxScore_{(tr,P)} = Max\{pScore_1, ..., pScore_n\}$

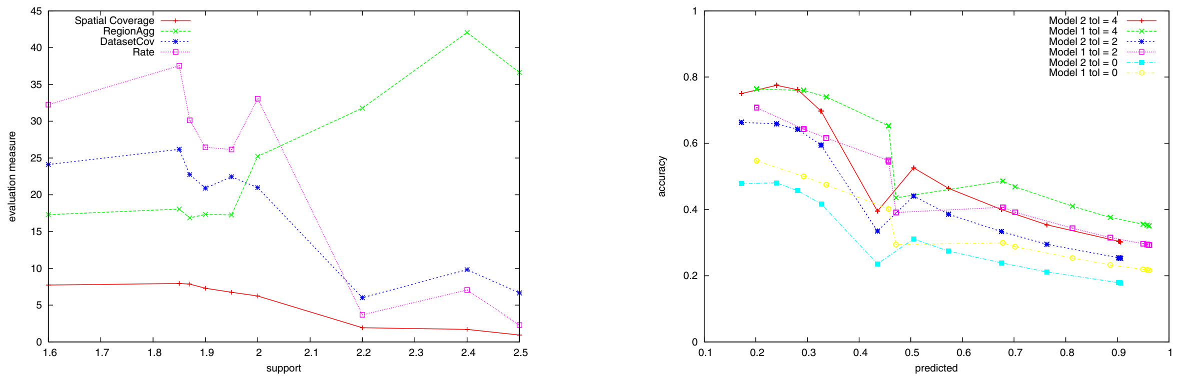
Each method produces a different behavior in term of prediction. The *avgScore* is the natural measure, which tries to generalize the concept of *similarity* as the average distance between the trajectory and each node. The *sumScore* is based on the concept of depth, i.e. it gives priority to the longest path that matches the trajectory. The *maxScore* is optimistic: if a trajectory has a good match with a node, this has priority over the other ones.

4.2.3 Prediction

The *Prediction* algorithm (Algorithm 2) uses a set of input parameters described below:

time tolerance (th_t) : represents the tolerance to enlarge the $WhereNext$ window in Fig.3(b) and Fig.3(c).

space tolerance (th_s) : represents the admissible spatial distance between the trajectory and the region. If the distance is greater than this tolerance, the punctual score is 0 since the trajectory is too far away from the region to be considered acceptable.



(a) (b)

Figure 4: (a) Evaluation Test. (b) Different result for prediction, which uses a different configuration of *Trajectory Pattern*

score threshold (th_{score}) : if the best prediction score is lower than this value the algorithm does not return any prediction.

aggregation function (th_{agg}) : represents the type of aggregation we want to use to compute the score (Def.4).

Given a trajectory T and a *T-pattern Tree* $\mathcal{PT}$, the location *Prediction* algorithm (Algorithm 2) computes the path score for each path of the $\mathcal{PT}$ relative to T, in accordance with the aggregation function selected. When there is no punctual score for a node, such a node is a candidate for the prediction with a score equal to the path score from the root to its parent node. When the tree as been completely visited, the best match is computed by selecting those candidates with the best score, and the region associated whit the candidates is returned as the prediction. There may be several predictions with the same score corresponding to all the children in the last node of the path with the best score.

5. PREDICTIVE POWER ANALYSIS

In this section, we define an evaluation function for estimating the predictive power of the collection of T-patterns before the creation of the *T-pattern Tree* and the quality measures of the performance of the prediction to be done a posteriori.

5.1 A priori prediction power measure

Unfortunately, there are three cases that make our system unable to predict as it fails against the score threshold:

1. the given trajectory is longer than every T-pattern in the tree.
2. the given trajectory is too far away in space from every T-pattern in the tree.
3. the given trajectory is too far away in time from every T-pattern in the tree.

The presence of a great quantity of these cases decreases considerably the performance of our predictor (prediction rate and accuracy). It is a direct consequence of quality of the T-pattern collection selected to build the predictor. Therefore, it is necessary to evaluate a priori the predictive power of a T-pattern set.
To this aim, we have been inspired by the work [1], where the authors proposed an interesting methodology establishing a correspondence between the accuracy and the support threshold of a set of association rules to be used as predictive rules. In our case, the ability to make a good prediction also depends on the spatial characteristics of a set of T-patterns and not only on the support (data coverage). The patterns/rules, considered in our work, are sequences of spatial regions with different size and time interval. Indeed, the size of regions and their incidence over the considered space (spatial coverage) is a key point in the prediction quality. For example, if our set of T-pattern covers only a small portion of the overall space, surely there will be many trajectories for which no prediction will be possible as no useful model (T-pattern) exists. On the other side, if the coverage of the space is obtained with regions of big size, the predicted locations are too wide and therefore not useful for any services. Our evaluation function takes into consideration these three aspects: the spatial coverage, the dataset coverage and the spatial separation.

$$spatialCoverage = \frac{\bigcup_{T_p \in T_p\_Set} space(T_p)}{TotalSpace}$$

The spatial coverage measures the fraction of space covered by the set of T-patterns. Let $space(T_p)$ denote the function which returns the space covered by a single T-pattern and $totalSpace$ denote the space covered by the area where we select the dataset of trajectories. A high value of this function means the possibility to predict almost all the locations in space, while with a low value only a small portions of the space can be predicted.

$$dataCoverage = \frac{|\mathcal{T} \dashv T_p\_Set|}{|\mathcal{T}|}$$

The data coverage measures the support, namely the fraction of trajectories covered by the set of T-patterns. Let $\mathcal{T}$ be the set of trajectories used to extract the T-pattern set T_p_Set and $\mathcal{T} \dashv T_p\_Set$ be the trajectories that support at least one T-pattern in the set. Intuitively, this function computes the coverage of the local model on the training dataset. This function considers both the two aspects of

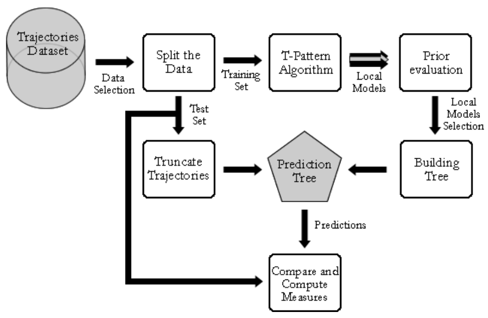


Figure 5: The process flow of our experiments

the problem, i.e., it counts not only how many trajectories match the spatial constraints of the model but also the time constraints represented by the intervals between the regions of T_p.

$$RegionSeparation = \frac{MinimalRegion}{Avg_{r \in T_p \in T_p\_Set} Area(r)}$$

The region separation measures the fraction of regions crossed by the analyzed T-patterns. Let *MinimalRegion* be the minimal spatial granularity in the considered space. The region separation represents two important aspects: the precision during the prediction (smaller regions lead to a more specialized local patterns) and the granularity of our prediction (larger regions lead to less accurate predictions).

$$Rate = spatialCoverage * dataCoverage * regionSeparation$$

The overall evaluation is the combination of all these measures. All measures and the overall pattern quality have values within 0 and 1. The behavior of these measures have been strongly experimented over the dataset described in Section 6. Here we anticipate few results of that section just to highlight different behaviors. In Fig.4(a) the three functions are shown with their values increased by 10000. In particular, we note that the *spatial coverage* decreases with high value of support, because some regions become not dense enough to survive. *Data coverage* has a similar behavior, although in some cases it increments with a lower value of support. Instead *region separation* increases with low value of support because the regions become isolated and then *Trajectory Pattern* mining algorithm can not aggregate them. The *Rate* function shows some local optima that represents good configurations of *Trajectory Pattern* mining algorithm to obtain a set of T-patterns useful for prediction.
To validate this evaluation functions, we tested the algorithm using two different configurations of *Trajectory Pattern* algorithm corresponding to the two peaks of Fig.4(a); the results obtained are in Fig.4(b) where the *Model1* is obtained using support 1.85 and *Model2* using the value 2.0.

5.2 A posteriori prediction performance analysis

In order to asses the quality of the prediction of our algorithm we use three performance measures:

Prediction rate : the number of predicted trajectories over the total number of trajectories we want to predict.

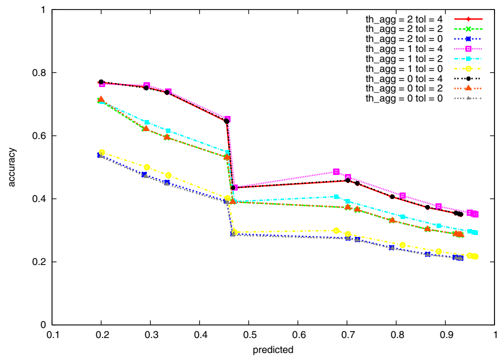


Figure 6: Accuracy versus Prediction rate

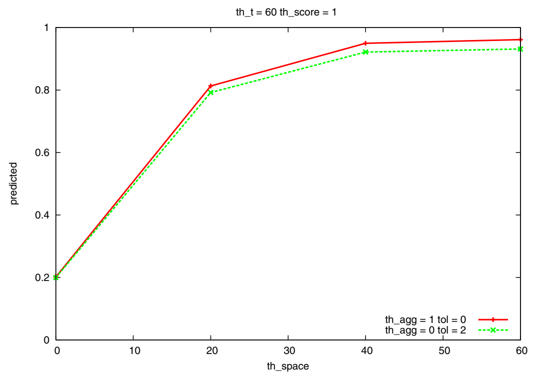
Accuracy : the rate of the correctly predicted locations divided by the predicted ones.

Average Error : the average spatial distance between the predicted location and the real position of the trajectory at the predicted time.

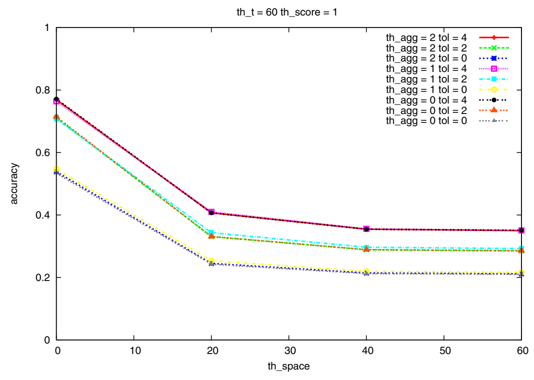
In order to appreciate the quality of the results we also define a *spatial tolerance* that is used as an acceptable distance between the predicted location and the real position of the trajectory. The *spatial tolerance* is expressed in units, where a unit is the minimum granularity in the considered space.

6. EXPERIMENTS AND RESULTS

For our experiments, we used a dataset provided by the GeoPKDD project[4], a set of trajectories of cars equipped with a GPS receiver moving in the city of Milan over a week. In Figure 5 we show the process flow of the experiments. First, we performed a selection in time and space, taking only the trajectories on Wednesday and Thursday in the city center between 7 am and 10 am. On the basis of this subset, we used all the Wednesday data (4000 trajectories) as a training set, and a part of the Thursday data as a test set (a sampling of 500 trajectories). We used two different days of the week for training and testing to evaluate our algorithm in the same time period but with some different local behaviors. After the data preparation from the training set, we extracted different sets of T-patterns using the *Trajectory Pattern* algorithm [11] and changing the support value, the temporal threshold and the resolution. By evaluating the discriminative power of these sets, we selected the local optimum corresponding to the following configuration: a temporal threshold equal to 200 seconds, a resolution of the regions of 100 meters and a support threshold of 1.85%. This configuration extracted 7039 T-patterns, which we used to build the *T-pattern Tree* used in our experiments. The results of the first set of experiments are shown in Figure 6. These summarize the accuracy of the model w.r.t. the prediction rate. Note how the prediction accuracy changes while the number of predicted trajectories grows without any particular thresholds settings, thus summarizing the general trend of the model. The three groups of curves represent the three values of tolerance used: 0, 2, 4. The three curves of each group represent the use of a different aggregation function. The best result is obtained with the tolerance threshold set from 0 to 4 and using the sum



(a)



(b)

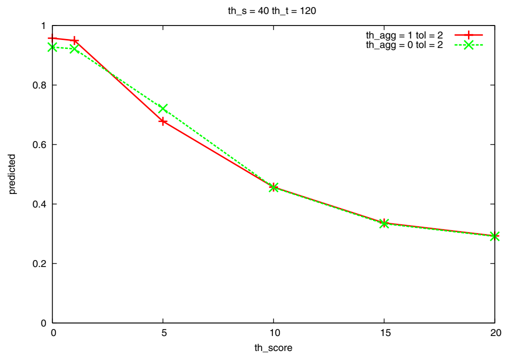
Figure 7: Accuracy and Prediction rate w.r.t th_{space}

as an aggregation function. This result justifies the quality of our approach showing the desirable behavior for a predictor, i.e., if we want to predict more cases we clearly lose accuracy. We obtain very good results in terms of accuracy when the algorithm predicts 45.7% trajectories; up to this value we predict with an accuracy greater than 40% without any tolerance, if we consider some tolerance the results are greater than 54%. This is a suitable trend for the model, for instance, in a streaming context: a traffic analyst might be more interested in getting a higher accuracy rather than predicting each individual trajectory.
We performed another set of experiments in order to understand how the introduced thresholds work with respect to the quality of the results, i.e. accuracy, and the number of predicted trajectories, i.e. Prediction Rate. Figure 8 (a,b) shows the impact of the different set up of the spatial threshold th_{space} w.r.t. the prediction rate and the prediction accuracy. In Figure 8 (a) the prediction rate increases when we adopt a greater spatial threshold. In Figure 8 (b) accuracy descreases when setting a higher value of the th_{space}. Both represent an understandable behavior of the model. The enlargement of the spatial threshold allows the punctual score computation in more situations. This means that we can predict more trajectories but, at same time, we introduce noise into the prediction results. A similar trend is shown in Figure 7(a,b), in fact the th_{score} has the same influence on the algorithm, i.e. a low value of this threshold entails selecting cases when the spatio-temporal distance between the trajectory segment and a region is high. Therefore, by increasing this threshold we are able to predict a greater number of trajectories but with less accuracy. Moreover, the average error is the target of a further analysis – see Figure 8(d), which shows the increase in the average error with a larger amount of predicted trajectories. This behaves in accordance with the accuracy trend.

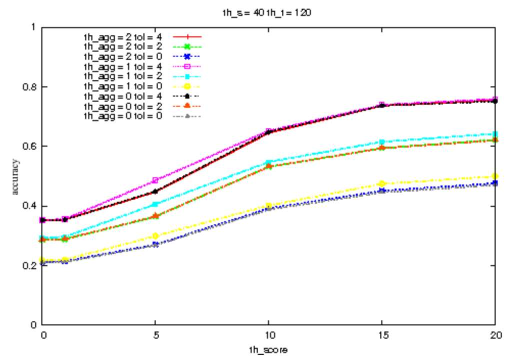
In general, the prediction rate is never below 20%. This is probably due to the dataset coverage of the set of T-patterns used with respect to the test set adopted, i.e. at least 20% of the trajectories in the test set are completely covered by the set of T-patterns that we selected.
The various values of the thresholds influence the quality of the overall results; however they are very useful for allowing end-users to drive the analysis. Indeed, a traffic analyst may be interested in testing different results and playing with the thresholds in order to obtain the desired level of accuracy, average error and prediction rate. Tollerance has a high impact on the results: the most accurate results are obtained with higher values of the tolerance.
Throughout this paper, we have stated that we use the behavior of a population to predict the next location of a mobile object. To justify this hypothesis, we tested our prediction algorithm in a different context. In fact, we built the model taking into account only the history of a user and we predicted his/her next location. We divided the user's history into a training set (90%) and test set (10%) and then we applied the same process described in Figure 5. The result in Figure 8(d) shows that, in this context, the quality of the result is not high, thus reinforcing our theory. In fact, the *Trajectory Pattern* algorithm cannot detect many frequent user behaviors that only have a few trajectories. Moreover, the unavailability of direct competitors in either of the two contexts [12, 13] means that we cannot compare the how our approach performs against other methods.

7. CONCLUSIONS

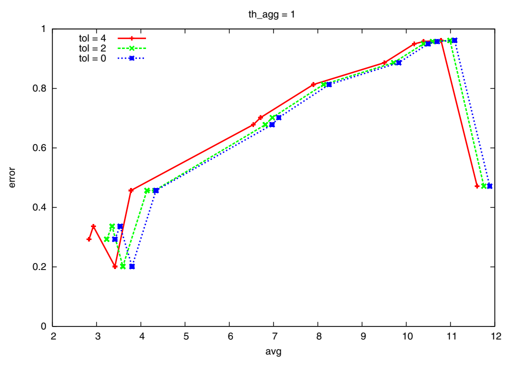
We have introduced a new technique to predict the next location of a moving object. Our definition of a future location prediction for a moving object is based on the previous movements of all moving objects in a certain area without considering any information about the user.
Previously proposed methods use temporal information only to order events, while we used T-patterns which are intrinsically equipped with temporal information. Furthermore, we have defined an evaluation function for a T-pattern set in order to allow us to choose the best one for the construction of a good *T-pattern Tree*.
Our experiments demonstrated that our technique gives reasonably accurate prediction and allows end users to tune the algorithm using a set of thresholds.
Further research will investigate the possibility of using this approach in real-time contexts, where next location prediction is useful. In fact, in real-time services knowing in advance the most likely position of an object may improve the quality of service, e.g. by pre-fetching queries or other resources. Another possible deployment would be in intelligent transportation systems, where the joint location prediction of many vehicles may trigger the dynamic readjustment of road network controllers or alert systems. A very



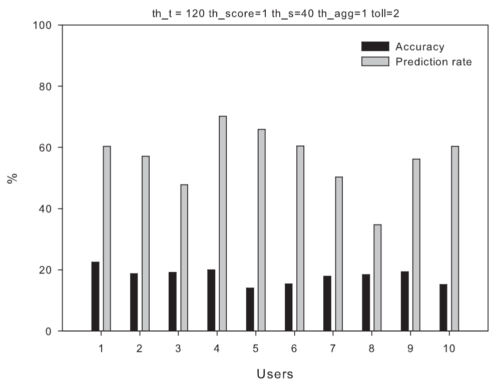
(a)



(b)



(c)



(d)

Figure 8: (a,b,c) Accuracy and Prediction rate w.r.t th_{score} *and average error study.* (d) Accuracy and Prediction rate in users context

critical point of our approach is the fact that the defined score functions give priority to frequency as an attractor of movements. However this can be risky since if there is a T-pattern with a high frequency it will dominate the other ones. Using some context dependent score function might mitigate this risk, and this would be very interesting aspect of a broader real case study.

8. REFERENCES

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

[2] F. Giannotti, M. Nanni, and D. Pedreschi. Efficient mining of temporally annotated sequences. SIAM, 2006.

[3] F. Giannotti, M. Nanni, F. Pinelli, and D. Pedreschi. Trajectory pattern mining. KDD 2007: 330-339.

[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. ICDE 2008: 70-79.

[6] K. Kianmehr and R. Alhajj. Effective classification by integrating support vector machine and association rule mining. IDEAL 2006: 920-927.

[7] B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. KDD 1998: 80-86.

[8] M. Morzy. Prediction of moving object location based on frequent trajectories. ISCIS, volume 4263 of *LNCS*, pages 583–592. Springer, 2006.

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

[10] 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.

[11] R. Ortale, E. Ritacco, N. Pelekis, R. Trasarti, G. Costa, F. Giannotti, G. Manco, and C. Renso. Daedalus: A knowledge discovery analysis framework for movement data. SEBD, 2008.

[12] B. Xu and O. Wolfson. Time-series prediction with applications to traffic and moving objects databases. 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. D.K.E., 54(2):121–146, 2005.