Mining Discriminative Distance-Aware Patterns for Trajectory Prediction

Chuancong Gao*, Xin Cao*, Gao Cong*, Man Lung Yiu#, Xiaokui Xiao*
*Nanyang Technological University, Singapore #Hong Kong Polytechnic University, Hong Kong, China
chuancong@gmail.com, xcao1@e.ntu.edu.sg, gaocong@ntu.edu.sg, csmlyiu@comp.polyu.edu.hk, xkxiao@ntu.edu.sg

ABSTRACT

With the proliferation of GPS-enabled devices, massive amounts of GPS data and trajectories are becoming available. Constructed from the GPS records of users, trajectories record the information of their historical movements. This enables the opportunity of discovering the behaviors of user movements. Motivated by this, we propose a new framework for predicting a moving object's next location based on its current trajectory. We define a new type of trajectory pattern named DD-pattern and devise an efficient trajectory pattern mining algorithm which mines highly discriminative patterns. The discovered patterns are then modeled as features in the trajectory prediction using a multi-class classifier. Results of empirical studies show that the proposed framework outperforms two of the state-of-the-art prediction methods significantly. The trajectory patterns defined are indeed useful for predicting objects' future locations. The experimental results also offer insight of the efficiency and scalability of the proposed pattern mining algorithm.

1 INTRODUCTION

The location-based service (LBS) is becoming increasingly popular due to the availability of GPS-enabled mobile devices. Existing LBS offers services to users based on their current locations. If we are able to predict the future location of a user accurately, then it helps enhance the provision of location-based services to the user. For example, when the LBS is informed the future location of a user, it can provide and prepare the corresponding services for the user in advance. As a specific example, a location-based mobile advertising service provider may utilize the predicted user location and display relevant advertisement to attract the user in advance. As another example, if the next destination of the user can be predicted, then the traffic condition around that region can be reported in order to avoid entering congested roads.

With the proliferation of GPS-enabled mobile devices, massive amounts of GPS data are generated and collected. The GPS records from a moving object approximate the object's trajectory, which is a sequence of timestamped locations. Several community-based websites enable users to share their routes and trajectories with the public. For example, GPSies [1], EveryTrail [2], and OpenStreetMap [3] allow users to upload and download personal trails. The substantial number of trajectories from these websites provide valuable information about the behaviors of user movements.

[1] http://www.gpsies.com
[2] http://www.everytrail.com
[3] http://www.openstreetmap.org/traces

The studies [1, 2] on these trajectories reveal two characteristics: (i) trajectories from the same user exhibit similar movement trends, and (ii) trajectories across different users may share certain patterns. For example, a driver takes the same route to the workplace on every weekday. In a dense region (e.g., business area, residential buildings), many pedestrians share similar travel patterns. By taking advantage of these similar movement patterns, it is possible to predict the future movement of an object accurately.

In this paper, we study the problem of predicting future movements using historical trajectories among different users. Several works attempt to predict the future movements of moving objects based on their historical movement. Tao et al. [3] develop a recursive motion function that uses the location records of an object in the last few timestamps to predict its next location. However, it is only applicable to very short range prediction. Ashbrook et al. [4] utilize previous GPS logs of an object to predict its next location in future. Monreale et al. [1] attempt to mine trajectory patterns from GPS logs and then apply these patterns to boost the accuracy of location prediction. They propose a novel kind of trajectory pattern called *time pattern* (T-pattern). A T-pattern describes the amount of time to travel along a sequence of locations, where among every two consecutive locations there is one time span indicating the possible travelling time. However, a T-pattern mined from different trajectories may have very large traveling time spans due to different speeds of different moving objects and their traffic environments, and thus very low discriminative power.

To address these limitations of T-pattern, we propose a new type of trajectory pattern called *distance pattern* (D-pattern). A D-pattern augment the normal sequential pattern of locations with the travel distance ranges between two consecutive locations in the sequence pattern. Note that multiple paths of different distances may exit between two locations and thus the distance range between two locations describes the possible travel distances between the two locations. It is independent of external properties like the speeds of moving objects and the traffic condition. Therefore, the D-pattern is expected to be more reliable than the T-pattern. Furthermore, we extend the D-pattern by DD-patterns to achieve better prediction accuracy.

One challenge of using patterns for trajectory predictions is that as the number of frequent patterns is exponential with the size of the dataset [5], we can easily be overwhelmed by a huge number of patterns. This makes it very difficult not only to design an efficient mining algorithm, but also to utilize a huge number of patterns for prediction. To address the challenge, we further define top-k discriminative DD-patterns for each location of each trajectory.

This is able to greatly reduce the number of patterns discovered while keeping only the patterns with the best discriminative powers, making the number of patterns tractable. However, the existing sequential pattern mining algorithms are not directly applicable to mine such patterns. A simple adaptation of the existing mining algorithms failed to mine the top-k discriminative DD-patterns after running for 1 day in our experiments. To this end, we propose an efficient algorithm for mining the top-k discriminative DD-patterns.

Another challenge is how to effectively make use of the discovered patterns for prediction, where the number of candidate future locations (labels) can be thousands and even more. In this paper, we convert the trajectory prediction problem into a multi-class classification problem, where each discovered pattern forms a feature. The idea is that, we view the historical trajectories (obtained from GPS logs) as the training set. In the training phase, given a trajectory instance, the last location is used as the class label, and the earlier locations can be used to build the features of this instance. At the prediction time, upon receiving an object's current trajectory, we determine the class label (i.e., the target location) of the object by using the features contained in this instance. We apply a multi-class Naive Bayes classifier [6] in the prediction since it is efficient and also offers high accuracy.

In summary, our contributions are two-fold.

- First, we propose a new framework for trajectory prediction using pattern features extracted from historical trajectories. In the framework, we define a new type of pattern, namely the top-k discriminative DD-patterns. We then devise an efficient mining algorithm to discover the top-k discriminative DD-patterns. With the discovered pattern features, together with the so-called last location features, we further build a multi-class classifier based on training trajectories. The classifier is then employed to predict the next location, given an object's current trajectory.
- Second, we study the properties of the paper's proposals empirically on both efficiency and effectiveness. The experimental results show that the framework outperforms two of the state-of-the-art work [1, 7] for trajectory prediction. The results also demonstrate that the mining algorithm offers efficiency and scalability while a simple adaptation of existing algorithm fails to mine the defined patterns.

The rest of the paper is organized as follows: Section 2 formally defines the problem and relevant concepts. Section 3 introduces two types of patterns we used, together with the definition of highly discriminative pattern. Section 4 presents the the mining algorithms. Section 5 presents the trajectory prediction model. We report on the empirical studies in Section 6. Finally, we cover related work in Section 7 and offer conclusions in Section 8.

2 PROBLEM DEFINITION

We first give a definition of trajectory:

Definition 2.1. **Trajectory.** The trajectory t of an object is defined as a sequence of time-stamped locations, namely $t = \{(l_1, \tau_1), (l_2, \tau_2), \cdots, (l_n, \tau_n)\}$ where l_i is the location of the object at time-stamp τ_i, and it satisfies $\tau_1 < \tau_2 < \cdots < \tau_n$.

As discussed in the Introduction, historical trajectories can be collected from sharing websites (e.g., GPSies, EveryTrail, OpenStreetMap). These form the trajectory database $\mathcal{DB}$ in our problem setting. We expect that $\mathcal{DB}$ contains many potential travel patterns, which can be used to provide accurate prediction in our problem.

In this paper, we focus on the *trajectory prediction problem*, and exploit the knowledge of $\mathcal{DB}$ to predict a future location of a given trajectory. In summary, our problem can be formally defined as:

Definition 2.2. **Trajectory Prediction Problem.** Given a past trajectory t of an object and a span s, our problem is to predict a future location l of the object after traveling a span s. The span s can be either a time or distance span depending on the scenario; $s = \langle 0, \infty \rangle$ if it is unknown or unspecified. The prediction process may exploit the knowledge of a trajectory database $\mathcal{DB}$.

Our main objective is to develop a trajectory prediction algorithm with high prediction accuracy. We achieve this by mining a set of highly discriminative patterns as features for building classification model and classifying test trajectories.

3 PATTERNS AND DISCRIMINATIONS

We first introduce several types of trajectory patterns and then discuss the discriminative ability of these patterns.

3.1 Preliminary on Time Pattern: T-pattern

T-pattern was first introduced by Monreale et al. [1] for trajectory prediction. A *time pattern* (i.e., T-pattern) describes the amount of time to travel along a sequence of locations (see Definition 3.1). In general, the travel time between two locations can be affected by different routes, traffic situations, and speeds. Instead of using a fixed value, we denote a travel time by a time span $ts\ \langle ts^{min}, ts^{max} \rangle$ with minimum and maximum time lengths.

Definition 3.1. **T-Pattern.** A T-pattern is defined as $l_1 \xrightarrow{ts_1} l_2 \xrightarrow{ts_2} \cdots \xrightarrow{ts_{n-1}} l_n$, where l_i $(1 \le i \le n)$ is a location and $ts_i = \langle ts_i^{min}, ts_i^{max} \rangle$ $(1 \le i \le n-1)$ is the travel time span between l_i and l_{i+1}.

Observe that a trajectory $t = \{(l_1, \tau_1), (l_2, \tau_2), \cdots, (l_n, \tau_n)\}$ can be converted to a T-pattern format $l_1 \xrightarrow{ts_1} l_2 \xrightarrow{ts_2} \cdots \xrightarrow{ts_{n-1}} l_n$, by setting each time span to $ts_i = \langle \tau_i, \tau_{i+1} \rangle$, i.e., the traveling time from l_i to l_{i+1}.

3.2 Types of Proposed Trajectory Patterns

Trajectory patterns can be used to express potential movement trends of objects. We present the proposed D-pattern and then discuss extended patterns (TT-pattern and DD-pattern).

3.2.1 Distance Pattern: D-pattern. Unfortunately, T-patterns may not capture movement trends very well, especially in realistic situations with various types of moving objects and complicated traffic scenario. A route may correspond to T-patterns with very large time spans because the travel time between locations depends heavily on the properties of moving objects (e.g., speeds) and the environment (e.g., traffic and stops).

We observe that the travel distance between two locations is independent of properties like speed and traffic. Hence, we propose the *distance pattern* (D-pattern) which describes the travel distances

(as different paths between two locations have different distances) along a sequence of locations, as defined below:

Definition 3.2. **D-Pattern.** A D-pattern is defined as: $l_1 \xrightarrow{ds_1} l_2 \xrightarrow{ds_2} \cdots \xrightarrow{ds_{n-1}} l_n$, with $ds_i = \langle ds_i^{min}, ds_i^{max} \rangle$ $(1 \le i \le n-1)$ as the travel distance span between l_i and l_{i+1}.

Each D-pattern extracted is a concise description of frequent behaviors of objects, in terms of both locations and travel distances. For example, a D-pattern

$$a \xrightarrow{\langle 4,6 \rangle} e \xrightarrow{\langle 2,2 \rangle} f$$

says that objects typically typically move from a to e with a travel distance between 4 and 6 and then move to location f with a travel distance 2.

Note that we use a distance span between two locations instead of a fixed value in a D-pattern. This is to characterize the fact that there are usually multiple routes of different travel distances between two locations and objects may follow different routes. For example, as shown in Figure 1, there are two different routes from location a to e, which have different travel distances. Using a fixed distance value in the D-pattern leads to two different patterns, i.e., $a \xrightarrow{4} e$ and $a \xrightarrow{6} e$. However, using a distance span instead of a fixed distance value, we generate a single D-pattern $a \xrightarrow{\langle 4,6 \rangle} e$, which captures the user movement behavior that users typically move from a to e with a travel distance between 4 and 6.

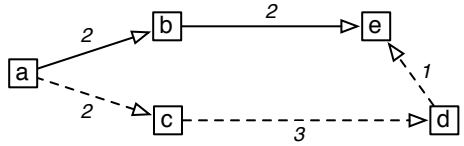


Figure 1: An Example of D-Pattern

As we will see in Section 6, the D-pattern offers much better prediction accuracy than does the T-pattern.

Again, a trajectory $t = \{(l_1, \tau_1), (l_2, \tau_2), \cdots, (l_n, \tau_n)\}$ can be converted to a D-pattern $l_1 \xrightarrow{ds_1} l_2 \xrightarrow{ds_2} \cdots \xrightarrow{ds_{n-1}} l_n$ with each distance span ds_i $(1 \le i \le n-1) = \langle dist(l_i, l_{i+1}), dist(l_i, l_{i+1}) \rangle$, where $dist(l_i, l_{i+1})$ is the known traveled distance between locations l_i and l_{i+1} in the given trajectory t.

We proceed to define the cover relationship between two D-patterns. The cover relationship can be used to check whether a D-pattern is covered by a trajectory. Note that on T-pattern we can also define the similar relationship.

Definition 3.3. **Cover Relationship of D-Pattern.** Let $p = l_1 \xrightarrow{ds_1} l_2 \xrightarrow{ds_2} \cdots \xrightarrow{ds_{n-1}} l_n$ and $q = l'_1 \xrightarrow{ds'_1} l'_2 \xrightarrow{ds'_2} \cdots \xrightarrow{ds'_{m-1}} l'_m$ be two D-patterns. We say that pattern q is covered by p, denoted by $q \sqsubseteq p$, if there exists $1 \le i_1 < i_2 < \cdots < i_m \le n$ such that $l_{i_j} = l'_j$ $(1 \le j \le m)$ and the distance range between l_{i_j} and $l_{i_{j+1}}$, i.e., $\langle \sum_{k=i_j}^{i_{j+1}-1} ds_k^{min}, \sum_{k=i_j}^{i_{j+1}-1} ds_k^{max} \rangle$ is a sub-range of $\langle ds'^{min}_j, ds'^{max}_j \rangle$ $(1 \le j \le m-1)$.

Intuitively, we say q is covered by p if q is more general than p (i.e., p is more specialized). Specifically, q is covered by p if the following two conditions are met (1) all the q's locations appear in p and (2) for two consecutive locations in q, its distance span covers that in p. Consider an extracted D-pattern q and a trajectory p. If D-pattern q is covered by trajectory p, the pattern q can characterize the moving behavior captured by trajectory p.

For example, as shown in Figure 2, given $p = a \xrightarrow{\langle 1,2 \rangle} b \xrightarrow{\langle 3,4 \rangle} c$ and $q = a \xrightarrow{\langle 2,7 \rangle} c$, we have $q \sqsubseteq p$, as (1) both locations a and c of q's are in p, and (2) q's span $\langle 2, 7 \rangle$ between a and c covers p's span $\langle 1+3, 2+4 \rangle$ between a and c. The former checking ensures all the q's locations appear in p; the latter one ensures the span between any two locations in q covers the span between the same two locations in p, thus making sure q is more general than p if q is covered by p.

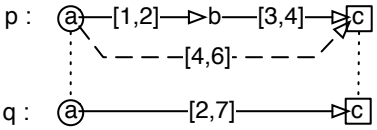


Figure 2: An Example of $q \sqsubseteq p$

3.2.2 Extending Patterns with Spans. We propose *extended patterns* by concatenating a pattern with a span, to further provide more information and predictive power.

Definition 3.4. **Extended Pattern.** Given a pattern p and a span s, we define an extended pattern $p \oplus s$ as pattern p followed by span s. We call $p \oplus s$ as a **TT-pattern** if p is T-pattern and s is a time span, or a **DD-pattern** if p is D-pattern and s is a distance span.

Furthermore, since a trajectory t can be viewed as a pattern, $t \oplus s$ is also an extended pattern.

For example, given a DD-pattern $p = a \xrightarrow{\langle 1,2 \rangle} b \xrightarrow{\langle 2,3 \rangle} c$ and span $s = \xrightarrow{\langle 2,\infty \rangle}$, we obtain the extended DD-pattern $p \oplus s = a \xrightarrow{\langle 1,2 \rangle} b \xrightarrow{\langle 2,3 \rangle} c \xrightarrow{\langle 2,\infty \rangle}$. The $\langle 2, \infty \rangle$ in $p \oplus s$ means that the next location following pattern p should be within the spatial range of 2 to ∞.

The extended patterns are more meaningful than T/D-patterns. As with different span s, the extended pattern $p \oplus s$ could lead to different subsequent target locations. For example, even with the same past traveling history, the driver is expected to arrive at different places after traveling 1-2 miles and 4-5 miles.

The cover relationship on extended patterns is formally defined as follows, taking DD-pattern as an example. DD-pattern's cover relationship is mainly used to test whether a DD-pattern q can be used to characterize DD-pattern p (formed from a trajectory together with its future traveling distance span) DD-pattern q.

Definition 3.5. **Cover Relationship of DD-pattern.** Given two DD-patterns $p = l_1 \xrightarrow{ds_1} l_2 \xrightarrow{ds_2} \cdots \xrightarrow{ds_{n-1}} l_n \xrightarrow{ds_n}$ and $q = l'_1 \xrightarrow{ds'_1} l'_2 \xrightarrow{ds'_2} \cdots \xrightarrow{ds'_{m-1}} l'_m \xrightarrow{ds'_m}$, we say q is covered by p, denoted by $q \sqsubseteq p$, if:

(1) There exists $1 \leq i_1 < i_2 < \cdots < i_m \leq n$, such that $l_{i_j} = l'_j (1 \leq j \leq m)$ and the distance range between l_{i_j} and $l_{i_{j+1}}$, $\langle \sum_{k=i_j}^{i_{j+1}-1} ds_k^{min}, \sum_{k=i_j}^{i_{j+1}-1} ds_k^{max} \rangle$, is a sub-range of $\langle ds_j'^{min}, ds_j'^{max} \rangle$ $(1 \leq j \leq m-1)$;
(2) ds'_m intersects with the distance span starting from l_{i_m}, i.e., $\langle \sum_{k=i_m}^{n} ds_k^{min}, \sum_{k=i_m}^{n} ds_k^{max} \rangle$.

The first condition tests whether q's D-pattern part is covered by p's, and the second condition ensures that the last span of q intersects with the concatenation of all the rest spans after p's last matched location l_{i_m}. The following example illustrates the cover relationship. Given D-pattern $q = a \xrightarrow{\langle 1,2 \rangle} b$, DD-pattern $q^* = q \oplus \xrightarrow{\langle 1,3 \rangle}$, D-pattern $p = a \xrightarrow{\langle 1,1 \rangle} b \xrightarrow{\langle 2,2 \rangle} c$, and DD-pattern $p^* = p \oplus \xrightarrow{\langle 5,\infty \rangle}$, we have $q \sqsubset p$ but $q^* \not\sqsubset p^*$, as spans $\langle 1, 3 \rangle$ and $\langle 2+5, \infty \rangle$ do not intersect with each other. Intuitively, q can characterize p while q^* cannot describe p^*.

We next explain why the second condition check "intersect". This is because p's last span ds_n could be quite large or even infinite. For example, it is possible that we need to predict a driver's next location after driving more than 5 miles. In this case, the future uncertain last span $ds_n = \langle 5, \infty \rangle$ in the test trajectory p. In the second condition, if the span of a DD-pattern p intersects with the last span ds_n, then p can be used to characterize p and can be used to predict the next location of p if p is a test trajectory.

3.3 Discriminative Ability of Patterns

We first introduce the support and confidence measures of patterns. Then we explain why and how we find patterns with discriminative power.

3.3.1 Support and Confidence of Patterns. Suppose that we are given a trajectory database $\mathcal{DB}$ (where each trajectory in $\mathcal{DB}$ is also a D-pattern) and a D-pattern p, the *matched database* $\mathcal{MDB}_p$ stores the set of matched trajectories with respect to p. Each *matched trajectory* t_p is the sub-trajectory of a trajectory $t \in \mathcal{DB}$ where $p \sqsubseteq t$. Specifically, it is from the first location of t to the last location matched by p in t ($p \sqsubseteq t_p \sqsubseteq t$). The *projected database* $\mathcal{PDB}_p$ contains the set of projected trajectories of all the matched trajectories. Each *projected trajectory* of t is the rest part after removing t's matched trajectory from t. Extremely, if p contains no location, we have $\mathcal{DB} = \mathcal{PDB}_p$.

We proceed to discuss the support of a D-pattern p, as in the pattern mining literature. The *support* of p is defined as $sup_p = |\mathcal{MDB}_p| = |\mathcal{PDB}_p|$, i.e., the size of the matched database (frequency of p in the whole trajectory database). Pattern p is called *frequent* if $sup_{min} \leq sup_p$ where sup_{min} is a user-specified minimum support threshold.

Similarly, as for a DD-pattern $p^* = p \oplus s$, we could also define sup_{p^*} according to the definition of DD-pattern's sub-pattern relationship. Given p^*, a *matched trajectory* t_{p^*} with regard to p^* is the sub-trajectory of a trajectory $t \in \mathcal{DB}$ where $p^* \sqsubseteq t$, and it is from the first location of t to the last distance span matched by p^* in t ($p^* \sqsubseteq t_{p^*} \sqsubseteq t$). Each *projected trajectory* of t with regard to p^* is the rest part after removing t_{p^*} from t. The *matched database* $\mathcal{MDB}_{p^*}$ contains the set of matched trajectories, and the *projected database* $\mathcal{PDB}_{p^*}$ contains the set of projected trajectories. Finally, the *support* of p^* is defined as $sup_{p^*} = |\mathcal{MDB}_{p^*}| = |\mathcal{PDB}_{p^*}|$. It is obvious that $sup_{p^*} \leq sup_p$, because if $p^* \sqsubseteq t$ it must hold true that $p \sqsubseteq t$ (the DD-pattern's sub-trajectory relationship has one more constraint than that of the D-pattern).

Given a DD-pattern p^* and a following location l, the *confidence* of p^* with respect to l is defined as: $conf_{p^*}^l = sup_{p^* \oplus l} / sup_{p^*}$. A high confidence value $conf_{p^*}^l$ means that l has a high probability as the following location of p^* after a traveling span s. Note that the confidence is just a natural measure to characterize the discriminative ability of the pattern. There exist other discriminative measures, such as the information-gain ratio or the fisher score [8]. In fact, the proposed methods in this paper are applicable to any discriminative measure.

3.3.2 Classification Training Trajectory Set. In this paper, we take the classification approach to tackle the trajectory prediction problem. We build the training set using the trajectory database $\mathcal{DB}$. Following we discuss how to determine the labels in the training set.

Let $t = l_1 \xrightarrow{ts_1} l_2 \xrightarrow{ts_2} \cdots \xrightarrow{ts_{n-1}} l_n$ be a training trajectory in $\mathcal{DB}$. It can be viewed as $n-1$ different training sub-trajectories, with each sub-trajectory's next location as its *classification label*. Each sub-trajectory starts from the first location of the original trajectory and ends before the label location. This reveals all the (trajectory, label) relationships in the original training trajectory.

Our training set is the *sub-trajectory database* $\mathcal{DB}^{sub}$, which includes sub-trajectories of each trajectory t in $\mathcal{DB}$.

3.3.3 Sub-Trajectory-Centric Discriminative Pattern. Our prediction model works as a multi-class classification model, with whether a training/testing trajectory contains a pattern as a classification feature. It is well known that the features are decisive to the classification accuracy. Hence, it is important to generate effective features.

One straightforward way is to apply frequent sequential pattern mining methods on the set of training sub-trajectories $\mathcal{DB}^{sub}$, subject to a specified minimum support sup_{min} and a confidence threshold. This is however impractical because: (1) The mining and prediction process will be inefficient and overwhelmed by a huge number of patterns. It is known that the number of frequent patterns is exponential to the size of the dataset [5]. A dataset containing thousands of trajectories could contain millions of frequent patterns. Even worse, we need to find patterns from sub-trajectories in the database rather than just trajectories. Moreover, the prediction process will also become very slow since we need to match a predicting trajectory to the entire set of patterns to get all its contained features. (2) It is very difficult to set a proper confidence threshold due to the large number of locations (labels) in our training data. A high confidence threshold may prevent discovering the patterns of many locations; however, a low confidence may generate a huge number of patterns with low discriminative ability.

To this end, we propose to mine a subset of highly discriminative patterns, and use them as our prediction model's features. The problem is which set of discriminative patterns should be chosen.

A simple method is to choose top-n patterns with the highest confidences among all frequent patterns. If a trajectory is covered

by one of the top-n patterns, the pattern's target location has a high chance to be the trajectory's next location. Normally, a group of top-n highly discriminative patterns only cover a small subset of all training trajectories, as many of them share common sub-patterns and cover similar trajectories. Information in those uncovered trajectories will not be captured in found patterns, resulting in losing important patterns and deteriorating the prediction accuracy. Moreover, these top-n patterns ignore the location labels. As some locations may not be associated with any selected pattern, such locations cannot be predicted. On the other hand, some sub-trajectories my be covered with too many patterns, and may cause over-fitting when training the classification model.

In this paper, we propose to select highly discriminative frequent patterns that cover most of the training sub-trajectories. Specifically, we design the following sub-trajectory-centric strategy: For each sub-trajectory t with label l (t's next location), we select top-k (e.g., $1 \leq k \leq 3$) frequent DD/TT-patterns with the highest confidence values for t with regard to l, to represent the best characteristics of the sub-trajectory t. This way ensures that the selected patterns cover most of the training sub-trajectories with an appropriate minimum support threshold. In other words, the selected patterns capture important characteristics from each training sub-trajectory. Moreover, the total number of patterns, each of which is used as a prediction feature, is not large (depending on the value of k). This is beneficial to the efficiency of the prediction process. Note this is equivalent to finding top-k frequent DD/TT-patterns for each location (excluding the first one) of each trajectory in $\mathcal{DB}$.

4 TRAJECTORY PATTERN MINING ALGORITHMS

We devise three approaches for mining the highly discriminative patterns defined in Section 3.3.3. They use the same set of training trajectories to mine patterns, and return the same set of top-k patterns for each sub-trajectory contained by a training trajectory.

4.1 Baseline Two-Step Approach

In this approach, for each trajectory we generate a set of sub-trajectories as introduced in Section 3.3.3, obtaining a sub-trajectory dataset $\mathcal{DB}^{sub}$ from the original training dataset $\mathcal{DB}$. This approach works on $\mathcal{DB}^{sub}$ to find the set of highly discriminative patterns by the following two steps:

(1) Mine: Extending the paradigm pattern-growth framework (such as PrefixSpan [9]) to accommodate the checking of containment relationship for T/D-patterns so that we can mine the complete set of frequent patterns on the training sub-trajectories $\mathcal{DB}^{sub}$.
(2) Filter: For each sub-trajectory t in $\mathcal{DB}^{sub}$, among the patterns covered by t, we select the top-k patterns with the highest confidences with respect to t's label (the next location). The union set of each sub-trajectory's top-k patterns is returned as the final result.

As step (ii) is straightforward, we next explain step (i). Following the definitions of the projected trajectory and projected trajectory database in Section 3.3, we define the *projected sub-trajectory* and *projected sub-trajectory database* by simply substituting "trajectory" by "sub-trajectory" in the definitions.

We start from an empty set $\mathcal{P}$ of prefix patterns, and recursively use each discovered pattern p in $\mathcal{P}$ as a prefix pattern to generate longer frequent patterns, which are in turn added into $\mathcal{P}$. The process continues until $\mathcal{P}$ cannot be extended. Specifically, for each prefix pattern p, we find a set of locally frequent locations whose support (on $\mathcal{DB}$ by counting sub-trajectories of the same trajectory as one, instead of on $\mathcal{DB}^{sub}$) is larger or equal than sup_{min} in p's projected sub-trajectory database $\mathcal{PDB}_p^{sub}$. For each locally frequent location l, we compute the time/distance span s which is from p to l as follows: for each sub-trajectory t containing p, we compute the time/distance from p's last location in t to l; then we find the minimal left boundary s^{min} and the maximal right boundary s^{max} from all these time/distance values; and finally we set $s = (s^{min}, s^{max})$ so that s covers all these values. We use s and l to generate a new prefix pattern $p' = p \oplus \xrightarrow{s} l$ if l has not appeared in p, and add it to $\mathcal{P}$. Each visited TT/DD-pattern $p^* = p \oplus \xrightarrow{s}$ is returned as one of the output patterns.

Note that when calculating $conf_{p^*}^{l}$, we use sup_{p^*} and $sup_{p^* \oplus l}$ on $\mathcal{DB}$ but not on $\mathcal{DB}^{sub}$, as the latter is incorrect — several sub-trajectories containing p^* in $\mathcal{DB}^{sub}$ correspond to one trajectory in $\mathcal{DB}$. This can be solved by counting projected sub-trajectories of the same trajectory only once, when calculating $conf_{p^*}^{l}$.

However, this approach is computationally prohibitive. As it is known that the number of frequent patterns increases exponentially with the size of the trajectory dataset. Given thousands of trajectories, the number of frequent patterns could easily exceed millions and the mining algorithm often cannot finish in days as to be shown in our experiments. In addition, this approach must work on sub-trajectory dataset $\mathcal{DB}^{sub}$, which is much larger than the original dataset $\mathcal{DB}$. The step (ii) is also very expensive due to the expensive testing of the containment relationship on millions of patterns generated from step (i) for all sub-trajectories in $\mathcal{DB}^{sub}$.

Figure 3 gives a brief description of the two-step approach and the other two approaches to be presented.

4.2 Sub-Trajectories-based Direct Mining Approach

We propose the sub-trajectories based direct mining approach (SbD) to mine highly discriminative frequent patterns from $\mathcal{DB}^{sub}$ directly, avoiding the filter step in the first approach. The idea is inspired by the fact that the set of highly discriminative feature patterns is only a small subset of all the frequent trajectory patterns, and thus we hope to mine the set of required patterns directly.

We still utilize the modified pattern-growth framework presented in Section 4.1. The basic idea is presented as follows: for each sub-trajectory t in $\mathcal{DB}^{sub}$, we only maintain its list of top-k patterns with the highest confidences during mining. When a frequent prefix pattern p is discovered, we know p is contained in all the sub-trajectories in $\mathcal{DB}_p^{sub} = \{t |\ t$'s projected sub-trajectory w.r.t. p is contained in $\mathcal{PDB}_p^{sub}\}$. Furthermore, after we get a prefix D-pattern p, we could also get a set of DD/TT-pattern w.r.t. the next location (label) l of each sub-trajectory $t \in \mathcal{DB}_p^{sub} - p^* = p \oplus \xrightarrow{s^l}$

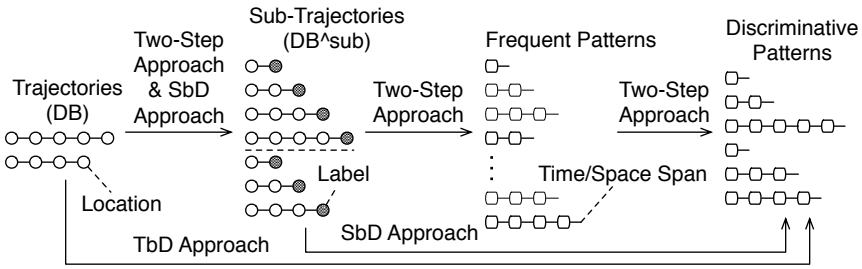


Figure 3: Three Different Approaches

(s^l is the span from p to l), and p^*'s support and confidence can be computed (as in Section 4.1). Hence, we can safely use p^* to update the top-k list of each sub-trajectory t in $\mathcal{DB}_p$, if p^* is better than the current top-k frequent patterns of t. Finally, we output the union of all the top-k patterns of every sub-trajectory as the final result.

As every frequent TT/DD-pattern p^* will be discovered by the algorithm, and they all are used to update each sub-trajectory t's top-k list where $p^* \sqsubseteq t$, this approach outputs exactly the same set of patterns as does the two-step approach.

While the time-consuming filter step is eliminated in this approach, it still works on the scale of all the sub-trajectories in $\mathcal{DB}^{sub}$. We next develop an approach that is able to mine $\mathcal{DB}$ directly.

4.3 Trajectory-based Direct Mining Approach

The two approaches in Section 4.1 and Section 4.2 must work on sub-trajectory dataset $\mathcal{DB}^{sub}$. Based on SbD, we develop a trajectory-based direct mining approach (TbD) that is able to mine patterns on trajectory dataset $\mathcal{DB}$ directly, which is much smaller than $\mathcal{DB}^{sub}$.

Recall that each sub-trajectory's label is just its next following location (in Section 3.3) in the original trajectory in $\mathcal{DB}$ which generates the sub-trajectory. As there is a one-to-one relationship between each sub-trajectory and its label (next location), we can maintain each sub-trajectory's top-k discriminative pattern list on its next location in $\mathcal{DB}$, instead of explicitly on the sub-trajectory.

Furthermore, recall that in approach SbD for each found pattern p we update each sub-trajectory t's top-k list with $p^* = p \oplus \xrightarrow{s^l}$ according to $conf^l_{p^*}$, where t contains p, l is the label of t, and s^l is the time/distance span from p to l. When we mine patterns on trajectory dataset $\mathcal{DB}$, the generated projected trajectory dataset for each prefix pattern p is different from those generated by SbD on dataset $\mathcal{DB}^{sub}$. However, we can still use the previous manner of SbD to extend each prefix pattern p using local frequent locations in $\mathcal{PDB}_p$. Specifically, we return all local locations (both frequent and infrequent) to update each location's top-k pattern list, while only using those frequent ones to extend current prefix pattern for longer prefix patterns.

We detail TbD in Algorithm 1, using D-pattern as an example. A special subscript form $_{(l)}$ is used here to denote variables with respect to current prefix pattern and location l. For example $ds_{(l)}$ is used to denote the distance span between current prefix and l.

TbD takes the trajectory dataset $\mathcal{DB}$ and a minimum support threshold sup_{min} as input, and the set of discovered patterns are stored in *features*. TbD starts the mining process by invoking mine with an empty D-pattern *patt*, and trajectory database $\mathcal{DB}$ as the projected database with respect to *patt* (since $\mathcal{DB} = \mathcal{PDB}_{patt}$ when *patt* is empty) (line 2).

Function mine works by extending the current prefix pattern *patt* with one of the frequent locations i found by function extend. For each extended pattern, we compute its confidence (line 4), and use it to update i's top-k feature list with the pattern $patt \oplus s$ generated with the prefix and the distance span s from *patt*'s last location to i. In addition, the extended pattern *patt* also becomes the next prefix pattern (line 6).

Function extend is mainly for computing the set of next local frequent locations used to extend the current prefix and the set of distance spans between the prefix and each of the local frequent location. Note that in function extend we have $ds^{min}_{(l_i)} = ds^{max}_{(l_i)}$ since in existing trajectories the distance span is just a certain value. Hence we only need to keep one value s instead of keeping both minimum and maximum values of the traveling distance span. Finally, we return the union of all the features of each location in each trajectory as the final feature set.

5 PREDICTION MODEL CONSTRUCTION

The algorithms in Section 4 return a set of highly discriminative frequent patterns, denoted by *features*. We present how to utilize the set of patterns to extract features in Section 5.1. We further discuss using *last location patterns* to enrich the feature space formed by *features*, thus improving classification accuracy, in Section 5.2.

5.1 Extracting Features

We next focus on extracting features using the set of highly discriminative patterns *features*. After we have features extracted from both training and test datasets, it is straightforward to employ an existing classification method that is capable of processing multi-class classification to build a classifier.

Each discovered pattern in *features* forms a binary feature as the input for a classification method. Given a sub-trajectory $t = l_1 \xrightarrow{ds_1} l_2 \xrightarrow{ds_2} \cdots l_{i-1} \xrightarrow{ds_{i-1}}$ in $\mathcal{DB}^{sub}$, if t contains a pattern $p \in features$ where id_p is the pattern p's ID, the corresponding feature is set at 1, and 0 otherwise. Hence, we can associate each sub-trajectory t with a set of patterns contained by t. Specifically, t is represented by a tuple $\langle l_i, \{id_p | p \sqsubseteq t\}\rangle$, where l_i is t's next location (label).

Algorithm 1: TbD - Trajectory based direct mining

Input: Trajectory Database $\mathcal{DB}$, sup_{min}
Output: Feature Set *features*

$patt \leftarrow$ a empty D-pattern with no location
mine($patt, \mathcal{DB}, sup_{min}$) /* $\mathcal{DB} = \mathcal{PDB}_{patt}$ */
$features \leftarrow \emptyset$
foreach $traj \in \mathcal{DB}$ **do**
  foreach *not encountered location loc of* $traj$ **do**
    $features \leftarrow features \cup features_{(traj,loc)}$
return *features*

Function: mine($patt, \mathcal{PDB}_{patt}, sup_{min}$)
Input: Current Prefix Pattern $patt$, Current Projected Database $\mathcal{PDB}_{patt}$

foreach $\langle l_i, ds_{(l_i)}, \mathcal{PDB}_{(l_i)}\rangle \in$ extend($\mathcal{PDB}_{patt}$) **do**
  $patt^* \leftarrow patt \oplus \xrightarrow{ds_{(l_i)}}$
  $patt' \leftarrow patt \oplus \xrightarrow{ds_{(l_i)}} i$
  $conf_{patt^*} \leftarrow sup_{patt'}/sup_{patt^*}$
  foreach $\{t_k | pt_k \in \mathcal{PDB}_{patt} \wedge pt_k$ *is* t_k*'s projected trajectory*$\}$ **do**
    update $features_{(t_k,l_i)}$ with $patt^*$
  if $sup_{patt'} \geq sup_{min}$ **then**
    mine($patt', \mathcal{PDB}_{(l_i)}$) /* $\mathcal{PDB}_{patt'} = \mathcal{PDB}_{(l_i)}$ */

Function: extend($\mathcal{PDB}_{patt}$)
Input: Current Projected Database $\mathcal{PDB}_{patt}$
Output: Local Frequent Entry Dictionary dic

foreach $pt = \xrightarrow{ds_m} l_{m+1} \cdots \xrightarrow{ds_{n-1}} l_n$ *in* $\mathcal{PDB}_{patt}$ **do**
  $d \leftarrow 0$
  for $i \leftarrow m+1$ *to* n **do**
    $d \leftarrow d + ds_{i-1}^{min}$ /* $ds_{i-1}^{min} = ds_{i-1}^{max}$ */
    if l_i *not encountered in* pt **then**
      $ds_{(l_i)} \leftarrow \langle min(ds_{(l_i)}^{min}, d), max(ds_{(l_i)}^{max}, d)\rangle$
      $\mathcal{PDB}_{(l_i)} \leftarrow \mathcal{PDB}_{(l_i)} \cup \{\xrightarrow{ds_m} l_{m+1} \cdots \xrightarrow{ds_{n-1}} l_n\}$
return $\{\langle l_i, ds_{(l_i)}, \mathcal{PDB}_{(l_i)}\rangle | l_i$ appeared in $\mathcal{PDB}_{patt}\}$

A straightforward method of deriving the feature set for t is to check for each pattern to see if it is contained by t. This is however expensive and unnecessary. Consider the fact that several sub-trajectories $\mathcal{S}_t$ in $\mathcal{DB}^{sub}$ are generated from a single trajectory t in $\mathcal{DB}$, and each sub-trajectory in $\mathcal{S}_t$ is contained by another sub-trajectory in $\mathcal{S}_t$ with one more location at the end, we have the following theorem:

Lemma 5.1. *Consider a DD-pattern* $p = l_1 \xrightarrow{ds_1} l_2 \xrightarrow{ds_2} \cdots \xrightarrow{ds_{n-1}} l_n \xrightarrow{ds_n}$ *and a DD-pattern* $p^* = l_1 \xrightarrow{ds_1} l_2 \xrightarrow{ds_2} \cdots \xrightarrow{ds_{n-1}} l_n \xrightarrow{ds_n} l_{n+1} \xrightarrow{ds_{n+1}}$ *that contains p. For a DD-pattern* $p' = l'_1 \xrightarrow{ds'_1} l'_2 \xrightarrow{ds'_2} \cdots \xrightarrow{ds'_{m-1}} l'_m \xrightarrow{ds'_m}$, *we have* $p' \not\sqsubseteq p^*$ *if* $p' \not\sqsubseteq p$ *and one of the following conditions is true, :*

(1) *There exists* $1 \leq i_1 < i_2 < \cdots < i_m \leq n$ *such that* $l_{i_j} = l'_j$ $(1 \leq j \leq m)$, *but not all* $ds'^{min}_j \leq \sum_{k=i_j}^{i_{j+1}-1} ds_k^{min} \leq \sum_{k=i_j}^{i_{j+1}-1} ds_k^{max} \leq ds'^{max}_j$ $(1 \leq j \leq m-1)$.
(2) p'*'s D-pattern part is contained by* p*'s D-pattern part, but* $ds'^{max}_m < \sum_{k=i_{m+1}}^{n} ds_k^{min}$.

Proof. If (i) is true, according to the definition of DD-pattern's "containing" relationship, it is obvious $p' \not\sqsubseteq p^*$. If (ii) is true then p''s D-pattern is also contained by p^*'s D-pattern, as $p \sqsubset p^*$. Since $ds'^{max}_m < \sum_{k=i_{m+1}}^{n} ds_k^{min}$, we know that $ds'^{max}_m < \sum_{k=i_{m+1}}^{n} ds_k^{min} < \sum_{k=i_{m+1}}^{n+1} ts_k^{min}$. Therefore, we know that the two spans $\langle ds'^{max}_m, ds'^{max}_m\rangle$ and $\langle \sum_{k=i_{m+1}}^{n+1} ds_k^{min}, \sum_{k=i_{m+1}}^{n+1} ds_k^{max}\rangle$ have no overlap, and thus $p' \not\sqsubseteq p^*$ according to the definition. □

Note that $p' \not\sqsubseteq p$ is also true when p''s D-pattern part is contained by p's D-pattern part and $ds'^{min}_m > \sum_{k=i_{m+1}}^{n} ds_k^{max}$. However, in the case it is possible that $p' \sqsubseteq p^*$ since the span after p^* is larger than that after p.

With Lemma 5.1, given a set of sub-trajectories $\mathcal{S}_t$ generated from a trajectory t in $\mathcal{DB}$, we generate the feature sets for each of them as follows:

(1) Initialize a feature set fs with $features$, and let s be the sub-trajectory containing a single location in $\mathcal{S}_t$.
(2) Generate features for s using patterns in fs and then remove s from $\mathcal{S}_t$.
(3) Remove the patterns that cannot be contained by sub-trajectories in $\mathcal{S}_t$ from fs according to Lemma 5.1.
(4) Goto Step (ii) with the next sub-trajectory s' in $\mathcal{S}_t$ that contains one more location than s. Return when $\mathcal{S}_t = \emptyset$.

We can use any multi-class classification method to predict the next location. In our experiments, we use the Naive Bayes classifier. It is efficient for both training and test phases, which is important on large datasets containing hundreds of thousands sub-trajectories.

In summary, given a test trajectory t, we predict its next location as follows: (i) Generate the features for t. (ii) Use the pre-built classifier to predict the next location of t.

5.2 Using Last Location Features

We find that some sub-trajectories cannot be covered by any pattern in *features* and hence contain no features. To improve the coverage of our method, we propose to extract a type of new features, called *last location features*.

We proceed to describe *last location features*. Recall a trajectory t in $\mathcal{DB}$ generates several sub-trajectories $\mathcal{S}_t$ in $\mathcal{DB}^{sub}$. For each sub-trajectory $s = l_1 \xrightarrow{ds_1} l_2 \xrightarrow{ds_2} \cdots l_n$, $s \in \mathcal{S}_t$, we add a pattern $p_l = l_n \xrightarrow{\langle 0,\infty\rangle}$ into s's feature set, where p_l is a single-location pattern representing the last location of s. By doing so, we ensure that all sub-trajectories in $\mathcal{S}_t$ can be covered by at least a feature. With this strategy, the coverage of our method will become larger.

Incorporating last location features can also increase the prediction accuracy. This is because the last location is the one right before the label location, and thus the last location has strong influence on the label. However, last location patterns generally have lower confidence values comparing with longer patterns, and thus

many of them cannot be chosen as highly discriminative patterns. That is why we include them explicitly.

Actually, the Naive Bayes classifier reduces to the first-order Markov model when classifying a sub-trajectory containing only the last location pattern.

Lemma 5.2. *When classifying a sub-trajectory which contains only the last location pattern, the Naive Bayes classifier reduces to the first-order Markov model.*

Proof. Assuming t is the predicting sub-trajectory, l is the location to be predicted, p_i is one of the patterns, and f_i is the feature w.r.t. p_i, we have:

$$P(l|t) = P(l|f_1 f_2 \cdots) = P(l|p_1 p_2 \cdots)$$

However, as the sub-trajectory contains only the last location pattern, given l_p as the last location, we have:

$$P(l|t) = P(l|p_1 p_2 \cdots) = P(l|l_p)$$

which is just the definition of the first-order Markov model. □

5.3 Predicting Route

We proceed to present an algorithm for predicting the following route with a specified number of locations m. The algorithm uses the Naive Bayes classifier to compute probability values, and predicts a route with the highest probability score.

We present the proposed method at $m = 2$ and this can be trivially extended to larger m. Suppose we have a trajectory $l_n \cdots l_i$ (with time/distance span omitted) and we predict its next two locations. We have $P(l_{n+2} l_{n+1} | l_n \cdots l_i) = P(l_{n+2} | l_{n+1} l_n \cdots l_i) \times P(l_{n+1} | l_n \cdots l_i)$. Hence, we can predict the route by predicting its first location, and then its second location. The probability score of a route is the product of the probability scores of each location in each step, and we return the route with the highest score.

When we predict the location l_{n+j} ($j \geq 2$) using D-pattern, we need to estimate the distance span ds_{n+j-2} from l_{n+j-2} to l_{n+j-1}, which is not available in the given sub-trajectory $l_1 \xrightarrow{ds_1} \cdots \xrightarrow{ds_{n-1}} l_n$. According to the definition of D-pattern in Section 3.2.1, we can estimate ds_{n+j-2} by $dist(l_{n+j-2}, l_{n+j-1})$, the distance between l_{n+j-2} and l_{n+j-1}, since there is no location between them. If we use the T-pattern for prediction, we need to estimate the traveling time span, which is inaccurate and may harm the prediction accuracy.

Algorithm 2: Top-k Route Sequence Recommendation

Input: Predicting Trajectory $ptraj$, Route Length L, Top-k Value K
Output: Top-k Routes

```
Initialize an empty max-priority queue q
enqueue(q, ⟨1, ptraj⟩)
routes ← ∅
while |routes| < K ∧ |q| > 0 do
    ⟨tprob, traj⟩ ← dequeue(q)
    if |traj| = L then add(routes, traj)

    else
        foreach ⟨prob, label⟩ ∈ topKPredict(traj) and
          label ∉ predicted route do
            d ← dist(lastLoc(traj), label)
            enqueue(q, ⟨tprob × prob, traj ⊕ --d--> label⟩)
return routes
```

6 EXPERIMENTAL STUDY

We compare the accuracy of the proposed prediction method and two state-of-the-art baseline models. We also evaluate the efficiency and scalability of our three algorithms for mining the proposed patterns. All the experiments are conducted on a workstation with Intel Xeon CPU (2.67 GHz) and 24 GB of memory. Algorithms are implemented in C#, running under Windows 7 X64. 5-fold cross-evaluation is used when the evaluation is in terms of accuracy.

6.1 Datasets

We use 4 publicly available datasets in the evaluation. Each trajectory in these datasets consists of a sequence of GPS points with a fixed sampling interval [4]. Table 1 gives the detailed characteristics of each dataset.

The first dataset contains GPS information collected from moving trucks delivering concrete to several construction places around the Athens metropolitan area in Greece, denoted by trucks. The dataset consists of 725 continuous trajectories of 50 trucks for 33 days, with GPS information collected every 30 seconds. Using the grid-based clustering with the cell size of $340 * 340$ square meters, we divide the space of trucks into 1,673 locations.

We generate two synthetic datasets using Network-based Moving Object Generator [5], which generates spatio-temporal data on real road network data based on user-specified parameters. One generated dataset, denoted by sanjoaquin, is based on the road network of San Joaquin, CA, with 1,000 objects moving for 1,000 timestamps. After dividing it into about 100×100 cells [6], we get 6,348 trajectories on 4,035 locations. Similarly, we generate another dataset, denoted by sanfrancisco, based on the road network of San Francisco (bay area). We obtain 8,873 trajectories on 3,245 locations after grid-based clustering, using the same parameters.

The fourth dataset we use is a large real dataset uber [7], which comes from GPS logs taken from Uber customer-service cars driving around San Francisco, with 25,000 trajectories. After clustering with the cell size of $340 * 340$ square meters and removing single location trajectories, we have 24,499 trajectories on 4,500 locations.

6.2 Prediction Accuracy

We report the evaluation results of the proposed prediction method. We compare the prediction effectiveness of our method, denoted by TbD, with two state-of-the-art baselines WhereNext [1] and Markov-Topic [7]. Markov-Topic is based on both hidden Markov model and topic model and hence can predict every test sub-trajectory.

[4] Our algorithm also works on trajectory datasets with arbitrary intervals.
[5] http://iapg.jade-hs.de/personen/brinkhoff/generator/
[6] As the generator outputs non-GPS coordinates, we divide the dataset with a specified cell number.
[7] http://www.infochimps.com/datasets/uber-anonymized-gps-logs

Table 1: Detailed Dataset Characteristics

		Raw Data			Clustered Data				
	Type	#Obj.	#Point	Interval	#Trajectory	#Loc.	#Event	Avg. Len.	#Instance
trucks	Real	50	112,203	30 sec.	725	1,681	96,660	66.66	95,935
sanjoaquin	Synthetic	1,000	107,473	Fixed	6,348	4,035	195,212	30.76	188,864
sanfrancisco	Synthetic	1,000	109,025	Fixed	8,873	3,245	198,406	22.36	189,533
uber	Real	N/A	1,128,663	4 sec.	24,499	4,500	342,161	13.97	317,662

However, both TbD and WhereNext may not be able to predict for all the test sub-trajectories, since the frequent patterns used by them may not cover all the test sub-trajectories. Our improved prediction model incorporating the last location features, denoted by TbD*, is able to predict for all the test sub-trajectories. Figure 4 illustrates the coverage ability of different methods.

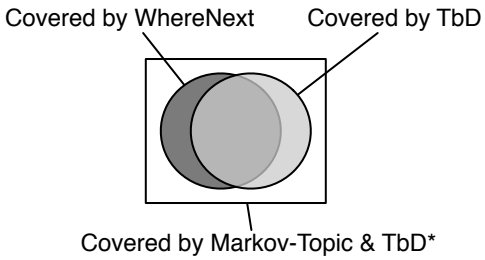


Figure 4: Sub-Trajectories Predictable by Different Algorithms

Since each algorithm has a different coverage, we report the accuracy of each method on three sets of test sub-trajectories, namely, the subset predictable by WhereNext, the subset predictable by TbD, and the whole test data. For sub-trajectories not covered by a method, we calculate the accuracy by assuming that the method returns wrong results on them. For example, when we evaluate on the whole test data, if a test sub-trajectory is not covered by patterns mined by TbD, we assume that TbD predicts it incorrectly.

Figure 5 compares the accuracy of the four methods, i.e., TbD, TbD*, Markov-Topic, and WhereNext, on the set of test sub-trajectories predictable by TbD. For TbD and TbD* we give the results of using TT-pattern (called TbD-T and TbD-T*) and DD-pattern (called TbD-D and TbD-D*), respectively. Table 2 shows the accuracy and coverage of them on all the test trajectories, and the set of trajectories predictable by WhereNext. For the TbD-series methods and WhereNext we use the same minimum support of 2 on the first three datasets, and 10 on uber, which is much larger. The feature number per sub-trajectory is set at 3 in the TbD-series methods. For Markov-Topic, we adopt a topic number of 10 as suggested in the work [7].

Comparison with baselines. Our method TbD-D* significantly outperforms the two baseline methods on the three subsets of test trajectories on different datasets in terms of accuracy, while Markov-Topic performs better than WhereNext. Markov-Topic predicts all testing trajectories as does TbD-D*. Baseline WhereNext generally has the lowest accuracy on all datasets, even on the set of test trajectories predictable by itself. Note that the coverage of WhereNext

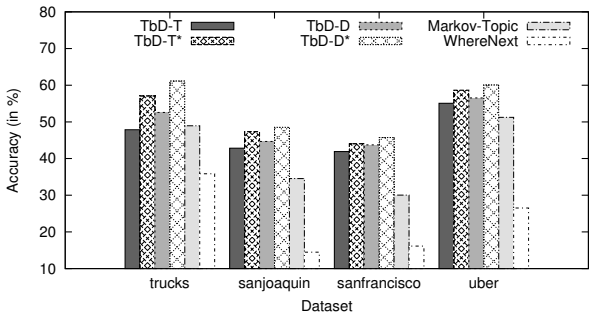


Figure 5: Accuracy on sub-trajectories predictable by TbD

on the set of trajectories predictable by TbD (Figure 5) is 86.82%, 81.47%, 98.05%, and 86.24% on trucks, sanjoaquin, sanfrancisco, and uber, respectively, while all the other methods cover all the test trajectories. Although WhereNext also uses T-patterns (not TT-patterns) in prediction, instead of using each T-pattern as a feature for classification, it builds a pattern tree using T-patterns. When predicting a test trajectory, it finds the best matching paths on the pattern tree based on the score of each path. However, if a trajectory is shorter than any T-pattern in the tree, it cannot be predicted. In addition, WhereNext uses the whole set of frequent patterns, rather than the highly discriminative patterns as does TbD-D*. We also tried to build a pattern tree using the same set of patterns used by TbD-D* for the WhereNext method. However, its coverage drops significantly and its accuracy remains similarly.

Comparing TT-pattern and DD-pattern. TbD-D and TbD-D* (using DD-pattern) preforms 1% to 5% better than TbD-T and TbD-T (using TT-pattern), which demonstrates the effectiveness of using DD-pattern over TT-pattern in prediction.

Usefulness of the last location feature. TbD-T* and TbD-D* (using last location features) generally outperform their counterparts TbD-T and TbD-D (without using last location features) by 2% to 9%. Furthermore, when using the last location features, the coverage of TbD-T* and TbD-D* is greatly improved. For example, the coverage of TbD-D* is 100% while TbD-D is 71.59% on trucks and 95.92% on sanjoaquin.

Number of patterns. Figure 6 gives the number of patterns used by TbD-D and WhereNext under the same minimum support. We can see that TbD uses much fewer patterns than does WhereNext since TbD uses only highly discriminative patterns while WhereNext uses all frequent patterns. For trucks, the number of patterns used by TbD is significantly smaller than those on the other datasets.

Table 2: Accuracy & Coverage Evaluation (in %)

(a) On Sub-Trajectories Predictable by WhereNext

	TbD-T	TbD-T*	TbD-D	TbD-D*	Markov-Topic	WhereNext
Accuracy						
trucks	51.03	58.82	55.66	**61.91**	47.42	38.94
sanjoaquin	47.03	49.99	48.82	**51.16**	35.09	17.72
sanfrancisco	42.24	44.26	44.14	**45.96**	29.93	16.44
uber	56.29	58.22	57.63	**59.52**	49.45	30.73
Coverage						
trucks	96.01	All	95.24	All	All	All
sanjoaquin	99.44	All	99.26	All	All	All
sanfrancisco	99.74	All	99.77	All	All	All
uber	99.97	All	99.97	All	All	All

(b) On All Sub-Trajectories

	TbD-T	TbD-T*	TbD-D	TbD-D*	Markov-Topic	WhereNext
Accuracy						
trucks	34.90	52.78	37.64	**55.50**	46.24	23.55
sanjoaquin	41.34	46.59	42.86	**47.56**	34.56	13.95
sanfrancisco	40.94	43.53	42.73	**45.17**	29.82	15.79
uber	51.21	58.06	52.47	**59.36**	51.17	24.60
Coverage						
trucks	72.92	All	71.59	All	All	60.48
sanjoaquin	96.53	All	95.92	All	All	78.73
sanfrancisco	97.73	All	97.73	All	All	96.04
uber	92.99	All	92.82	All	All	80.07

This indicates why the coverage of TbD on all the sub-trajectories on trucks is around 70%, while close to 99% on the other datasets.

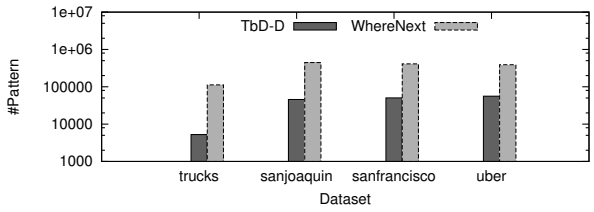
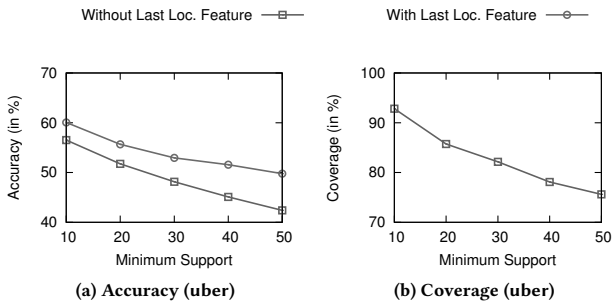


Figure 6: Comparison of number of patterns

The effect of the minimum support. This experiment is to evaluate the effect of the minimum support on the accuracy and coverage of TbD. To better reflect the accuracy of TbD, we report its accuracy on the set of trajectories predictable by TbD in this experiment. Its coverage is reported on the whole test dataset.

Figure 7 shows the results on dataset uber using TbD-D and TbD-D*, where the number of discriminative features per sub-trajectory is set at 3. We observe that both accuracy and coverage increase at a lower minimum support. It is easy to understand that the coverage increases as more distinct patterns are discovered at a lower minimum support. The accuracy improves because when we choose top-k best discriminative patterns per sub-trajectory at a lower minimum support, we could choose among a larger set of candidate frequent patterns. We also notice that when the minimum support decreases, TbD and TbD* achieve similar accuracy. This is because at a lower minimum support, the highly discriminative patterns have more influence on the prediction results. As we show in Section 6.3, the proposed mining algorithm TbD still runs very fast at a very low minimum support. Hence, we suggest to adopt a low minimum support. The results on the other three datasets are qualitatively similar and are ignored due to space limitations.

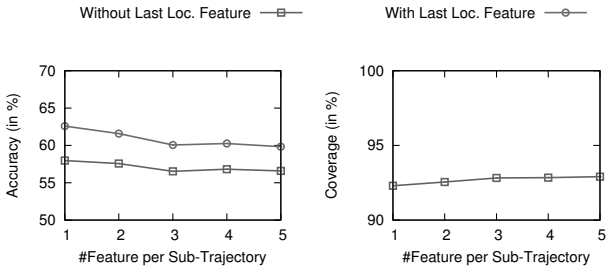


(a) Accuracy (uber) **(b) Coverage (uber)**

Figure 7: Accuracy & Coverage w.r.t. sup_{min}

The effect of the number of selected features per sub-trajectory. This experiment is to study the effect of the number of selected

discriminative patterns per sub-trajectory (denoted by k) on the accuracy and coverage. Figure 8 shows the results on dataset uber, with the same minimum support of 2. It can be observed that when k increases, the accuracy drops. A possible reason would be that larger k values will generate more less-discriminative patterns for each sub-trajectory. As expected, when k increases, the coverage increases, since more patterns are selected as features and thus more sub-trajectories can be covered. The results on the other three datasets are qualitatively similar and are ignored due to space limitations.



(a) Accuracy (uber) (b) Coverage (uber)

Figure 8: Accuracy & Coverage w.r.t. #Feature per Sub-Trajectory

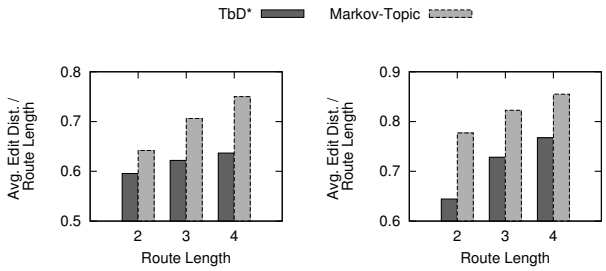
With a larger k, we can achieve better coverage but worse accuracy, and a smaller k results in better accuracy but decreases the coverage. Hence, there is a tradeoff between the coverage and the accuracy when selecting a proper value for k. Considering both the coverage and the accuracy, we recommend to set k at 3.

6.2.1 Route Prediction. We evaluate our route prediction method by comparing it to Markov-Topic [7]. We do not extend WhereNext for route prediction due to its too low single-location prediction accuracy. We use the Levenshtein edit distance [10] to measure the difference between the predicted route and the actual route. The route length is set at 2, 3, and 4, respectively. The score (the lower the better) on a specified route length is measured by the average edit distance of every route divided by the specified route length.

Figure 9 gives the results on datasets trucks and sanfrancisco for TbD-D* and Markov-Topic. As we see, TbD* achieves significantly better accuracy than does Markov-Topic. This could be due to that our single-location prediction model performs better than does Markov-Topic, and the route prediction is directly based on the single-location prediction. The results on the other two datasets are qualitatively similar and are ignored due to space limitations.

6.3 Efficiency of Mining Algorithms

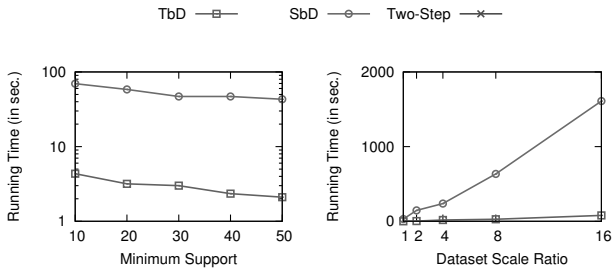
It is also important to study if our mining algorithm can mine highly discriminative patterns efficiently and scale to larger datasets (even at a low minimum support since it has been shown in earlier experiments our method achieves better coverage and accuracy at a low minimum support). This experiment is to evaluate the efficiency and scalability of the proposed algorithms.



(a) trucks (b) sanfrancisco

Figure 9: Route Sequence Accuracy w.r.t. Route Length

We compare the running time of our trajectory-based direct mining approach (TbD) with the baselines, two-step approach (Two-Step) and the sub-trajectory-based direct mining approach (SbD), under different minimum support thresholds. Figure 10a shows the results on the largest dataset uber. As can be seen, TbD takes less than 10 seconds at the minimum support 10, while the SbD approach is generally 10 times slower. The Two-Step approach actually could not even finish after 1 day on the dataset, and is not included in the figure.



(a) Runtime w.r.t. sup_{min} (b) Scalability ($sup_{min} = 10$)

Figure 10: Runtime & Scalability (uber)

Scalability. This experiment is to evaluate the mining algorithms' scalability. Figure 10b shows the runtime when we replicate uber by a scale ratio from 1 to 16, where a scale ratio 2 means that the data is replicated once. The minimum support and the feature number per sub-trajectory are set to 10 and 3, respectively. As we can see, both TbD and SbD scale well with the size of the dataset, while TbD scales better. We also observe that TbD can finish in less than one minute on the dataset with the size of 16 times of the original dataset, while SbD takes more than 1,000 seconds. The Two-Step approach could not even finish on the original dataset (with scale ratio of 1) after 1 day.

7 RELATED WORK

There exist works on predicting an object's future movement by the historical trajectories of all objects, the problem we address in this paper. The work [1], the most related to ours, presents the approach WhereNext, which predicts next location using all the frequent T-patterns mined from a set of past trajectories. Our method differs

from WhereNext in three aspects: 1) We define DD-patterns (and TT-patterns) and mine top-k discriminative DD-patterns while WhereNext mines all frequent T-patterns, which is much larger. 2) We propose an algorithm for mining top-k discriminative DD-patterns while WhereNext uses a method like our baseline approach, which fails to mine patterns on large dataset. 3) Our prediction method is totally different from WhereNext, which is based on a constructed pattern tree from all T-patterns. Each testing trajectory is predicted by choosing child nodes of the best matching paths of the trajectory on the tree. Another state-of-the-art work [7] predicts users' next traveling location based on a model combining the Markov model and the topic model. This work also considers to predict future route utilizing the single-location prediction model. We compare with [1, 7] in our experiments.

There are studies on predicting an object's future movement based solely on *the historical trajectory of the object itself*. For example, the work [11] predicts a user's next movement in a mobile cell network using the user's transition probabilities from one cell to another calculated by the Markov model based on the user's historical data. Jeung et al. [2] propose a hybrid manner to forecast an object's future location, with predefined motion functions constrained by a road network and the trajectory patterns of the object captured by the Markov model. The two proposals [2, 11] are not applicable to our problem. Yavas et al. [12] predict user's next movement by the matched rule with the highest confidence among all rules mined from the user's previous trajectories. Yavas et al. [12] employ traditional sequential patterns without time/distance span, and it is shown in [1] that the traditional patterns perform much worse than T-patterns for trajectory prediction. Recently, Cho et al. [13] propose a novel approach by utilizing a user's periodic behavior to help predict his/her future movement. They further adopt the user's social network connections to help predict long distance movements. However, their approach cannot be used for our problem, as the prediction model assumes pre-assigned latent states on each user.

There are also studies on trajectory pattern mining without prediction. The work [14] focuses on mining trajectory patterns, which are defined as a sequence of (location, velocity) tuples. The trajectory patterns discovered there are very different from ours. Giannotti et al. [15] propose a type of trajectory pattern which is a specialized version of T-pattern. Instead of using traveling time span between each two locations in the pattern, they use a certain traveling time value derived from statistical methods.

Our work is also related to the existing work on using association rules for classification. CBA [16] and CMAR [17] classify item-set data using high confidence association rules. Cheng et al. [8] propose using information gain ratio instead of confidence as the discrimination measure for better classification accuracy. To avoid the time-consuming feature pattern selection procedure, Cong et al. [18] propose a method mining top-k best covering patterns directly from gene data, and the HARMONY algorithm [19] chooses a covering rule with the highest confidence for each training instance on item-set data. Some other proposals [20, 21] also focus on classifying item-set data, with different discrimination measures or different instance covering strategies. These proposals use discriminative frequent patterns to classify item-set data, and are not applicable to trajectory prediction.

Lee et al. [22] use discriminative patterns for classifying trajectories into a small set of pre-defined classes, but not future locations as in our trajectory prediction problem. They use traditional sequential patterns mined by sequence pattern mining algorithms like PrefixSpan [9], rather than TT-patterns or DD-patterns used in this paper. In their other work [23], a different and interesting approach based on hierarchical region-based and trajectory-based clustering is proposed to address the same problem.

8 CONCLUSION

We propose a novel framework to predict a trajectory's future location and route. The framework defines a new type of pattern D-patterns, and top-k discriminative D-patterns for each location of a training trajectory. We propose an efficient algorithm for mining such patterns directly, without high-cost feature selection or sub-trajectory extraction. In the framework, each discriminative D-pattern is transformed into a feature. The D-patterns, together with the last location patterns, are used to build a multi-class classifier for trajectory prediction. The empirical results on several real and synthetic datasets show that our framework outperforms two state-of-the-art baselines significantly. The experimental study also demonstrate the efficiency and scalability of the proposed pattern mining algorithm TbD.

REFERENCES

[1] A. Monreale, F. Pinelli, R. Trasarti, and F. Giannotti, "Wherenext: a location predictor on trajectory pattern mining," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Paris, France: ACM, 2009, pp. 637–646.

[2] H. Jeung, Q. Liu, H. T. Shen, and X. Zhou, "A hybrid prediction model for moving objects," in *Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 70–79.

[3] Y. Tao, C. Faloutsos, D. Papadias, and B. Liu, "Prediction and indexing of moving objects with unknown motion patterns," in *Proceedings of the ACM SIGMOD International Conference on Proceedings of the ACM SIGMOD International Conference on Management of Data*. Paris, France: ACM, 2004, pp. 611–622.

[4] D. Ashbrook and T. Starner, "Using gps to learn significant locations and predict movement across multiple users," *Personal and Ubiquitous Computing*, vol. 7, no. 5, pp. 275–286, 2003.

[5] R. Agrawal and R. Srikant, "Mining sequential patterns," in *Proceedings of the Eleventh International Conference on Data Engineering*. Taipei, Taiwan: IEEE Computer Society, 1995, pp. 3–14.

[6] T. M. Mitchell, *Machine learning*. McGraw-Hill, 1997.

[7] T. Kurashima, T. Iwata, G. Irie, and K. Fujimura, "Travel route recommendation using geotags in photo sharing sites," in *Proceedings of the 19th ACM Conference on Information and Knowledge Management, CIKM 2010*. Toronto, Ontario, Canada: ACM, 2010, pp. 579–588.

[8] H. Cheng, X. Yan, J. Han, and C.-W. Hsu, "Discriminative frequent pattern analysis for effective classification," in *Proceedings of the 23rd International Conference on Data Engineering*. Istanbul, Turkey: IEEE, 2007, pp. 716–725.

[9] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M. Hsu, "Prefixspan: Mining sequential patterns by prefix-projected growth," in *Proceedings of the 17th International Conference on Data Engineering*. Heidelberg, Germany: IEEE Computer Society, 2001, pp. 215–224.

[10] "Levenshtein distance," http://en.wikipedia.org/wiki/Levenshtein_distance.

[11] A. Bhattacharya and S. K. Das, "Lezi-update: an information-theoretic approach to track mobile users in pcs networks," in *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*. New York, NY, USA: ACM, 1999, pp. 1–12.

[12] G. Yavas, D. Katsaros, O. Ulusoy, and Y. Manolopoulos, "A data mining approach for location prediction in mobile environments," *Data Knowl. Eng.*, vol. 54, pp. 121–146, 2005.

[13] E. Cho, S. A. Myers, and J. Leskovec, "Friendship and mobility: user movement in location-based social networks," in *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. San Diego, CA,

USA: ACM, 2011, pp. 1082–1090.

[14] J. Yang and M. Hu, "Trajpattern: Mining sequential patterns from imprecise trajectories of mobile objects," in *EDBT 2006, 10th International Conference on Extending Database Technology*. Munich, Germany: Springer, 2006, pp. 664–681.

[15] F. Giannotti, M. Nanni, F. Pinelli, and D. Pedreschi, "Trajectory pattern mining," in *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. San Jose, California, USA: ACM, 2007, pp. 330–339.

[16] B. Liu, W. Hsu, and Y. Ma, "Integrating classification and association rule mining," in *Proceedings of the Fourteen ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1998, pp. 80–86.

[17] W. Li, J. Han, and J. Pei, "Cmar: Accurate and efficient classification based on multiple class-association rules," in *Proceedings of the 2001 IEEE International Conference on Data Mining*. San Jose, California, USA: IEEE Computer Society, 2001, pp. 369–376.

[18] G. Cong, K.-L. Tan, A. K. H. Tung, and X. Xu, "Mining top-k covering rule groups for gene expression data," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*. Baltimore, Maryland, USA: ACM, 2005, pp. 670–681.

[19] J. Wang and G. Karypis, "On mining instance-centric classification rules," *IEEE Trans. Knowl. Data Eng.*, vol. 18, no. 11, pp. 1497–1511, 2006.

[20] H. Cheng, X. Yan, J. Han, and P. S. Yu, "Direct discriminative pattern mining for effective classification," in *Proceedings of the 24th International Conference on Data Engineering*. Cancún, México: IEEE, 2008, pp. 169–178.

[21] W. Fan, K. Zhang, H. Cheng, J. Gao, X. Yan, J. Han, P. S. Yu, and O. Verscheure, "Direct mining of discriminative and essential frequent patterns via model-based search tree," in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Las Vegas, Nevada, USA: ACM, 2008, pp. 230–238.

[22] J.-G. Lee, J. Han, X. Li, and H. Cheng, "Mining discriminative patterns for classifying trajectories on road networks," *IEEE Trans. Knowl. Data Eng.*, vol. 23, no. 5, pp. 713–726, 2011.

[23] J.-G. Lee, J. Han, X. Li, and H. Gonzalez, "*TraClass*: trajectory classification using hierarchical region-based and trajectory-based clustering," *PVLDB*, vol. 1, no. 1, pp. 1081–1094, 2008.