Mining Geographic-Temporal-Semantic Patterns in Trajectories for Location Prediction

JOSH JIA-CHING YING, National Cheng Kung University
WANG-CHIEN LEE, Pennsylvania State University
VINCENT S. TSENG, National Cheng Kung University

In recent years, researches on location predictions by mining trajectories of users have attracted a lot of attention. Existing studies on this topic mostly treat such predictions as just a type of location recommendation, i.e., they predict the next location of a user using location recommenders. However, an user usually visits somewhere for reasons other than interestingness. In this paper, we propose a novel mining-based location prediction approach called *Geographic-Temporal-Semantic based Location Prediction (GTS-LP),* which takes into account a user's Geographic-triggered Intentions, Temporal-triggered Intentions, and Semantic-triggered Intentions, to estimate the probability of the user in visiting a location. The core idea underlying our proposal is the discovery of trajectory patterns of users, namely *GTS Patterns*, to capture frequent movements triggered by the three kinds of intentions. To achieve this goal, we define a new trajectory pattern to capture the key properties of the behaviors that are motivated by the three kinds of intentions from trajectories of users. In our *GTS-LP* approach, we propose a series of novel matching strategies to calculate the similarity between current movement of a user and discovered GTS Patterns based on various moving intentions. On the basis of similitude, we make an online prediction as to the location the user intends to visit. To the best of our knowledge, this is the first work on location prediction based on trajectory pattern mining that explores the geographic, temporal, and semantic properties simultaneously. By means of a comprehensive evaluation using various real trajectory datasets, we show that our proposed *GTS-LP* approach delivers excellent performance and significantly outperforms existing state-of-the-art location prediction methods.

Categories and Subject Descriptors: **H.2.8 [Database Management]**: Database Applications—*Data Mining*

General Terms: Design, Algorithms, Experimentation

Additional Key Words and Phrases: Trajectory mining, frequent movement patterns, semantic trajectory, location prediction.

ACM Reference Format:

Josh Jia-Ching Ying, Wang-Chien Lee, and Vincent S. Tseng, 2013. Mining Geographic-Temporal-Semantic Patterns in Trajectories for Location Prediction. *ACM Trans. Intell. Syst. Technol.* xx, y, Article zz (Month YYYY), 34 pages.
DOI:http://dx.doi.org/xx.xxxx/0000000.0000000

1. INTRODUCTION

Many applications, such as navigational planning services, traffic management, and location-based advertisement, have been developed for the rapidly growing location-based services market. Due to the various requirements of these applications, e.g.,

This research was supported by National Science Council, Taiwan, R.O.C. under grant no. NSC101-2221-E-006-255-MY3 and NSC100-2631-H-006-001.

Author's addresses: Josh Jia-Ching Ying and Vincent S. Tseng (correspondence), Department of Computer Science and Information Engineering, National Cheng Kung University, Taiwan, ROC, email to: tsengsm@mail.ncku.edu.tw; Wang-Chien Lee, Department of Computer Science and Engineering, Pennsylvania State University, University Park, PA 16802, USA.

DOI:http://dx.doi.org/xx.xxxx/0000000.0000000

system efficiency and marketing efficacy, accurately predicting the next location to which a mobile user may move is essential. Given a set of locations, which may be application-dependent and pre-determined, the location prediction technique identifies the next location a user is most likely to visit. Intuitively, the process of location prediction is very similar to location recommendation. Consequently, many existing works [Ge *et al.*, 2010; Liu *et al.*, 2010; Ge *et al.*, 2011; Zhuang *et al.*, 2011] directly adopt a location recommender as their location prediction model. However, these recommenders recommend locations using a non-real-time estimation process, i.e., current movements of users are NOT taken into consideration in making the recommendations. We argue that, apart from location recommendation, the problem of next location prediction focuses on inferring the next location that a user will visit. In fact, people do not solely visit locations because these locations are interesting to them. They also go to places because they have to do something, e.g., for work, transportation, etc. However, conventional location recommendation methods aim to suggest a *new* location (or more precisely a place) that a user may be interested in [Zheng *et al.*, 2011]. In other words, the location recommendation method only cares about interests of users, while the next location prediction method cares about intentions of users. On the basis of our observations, we categorize users' intentions into three classes:

- *Geographic-triggered Intentions* (*GI*), which refer to the specific geographical locations visited by users. These kinds of intentions reflect the reasons why a user travels from one location to another location. For example, as shown in Fig. 1, given two consecutive metro stations X and Y, we can predict metro station Y as the next location for those users who are currently at metro station X (see Trajectory_1).
- *Temporal-triggered Intentions* (*TI*), which refer to the relationship between locations and temporal information. These kinds of intentions reflect the reasons why a user visits and leaves a location at a certain time. As shown in Fig. 1, the user tends to go back home in the evenings and leave home in the morning based on Temporal-triggered Intentions (see Trajectory_3).
- *Semantic-triggered Intentions* (*SI*), which refer to the "general" geographical consequence of locations visited by users. These kinds of intentions reflect the reasons why users travel from some locations to other locations. For example, if users always go for a meal after they leave their workplaces, we can predict locations that contain many restaurants as the next location for those users who are currently leaving school (see Trajectory_2 in Fig. 1).

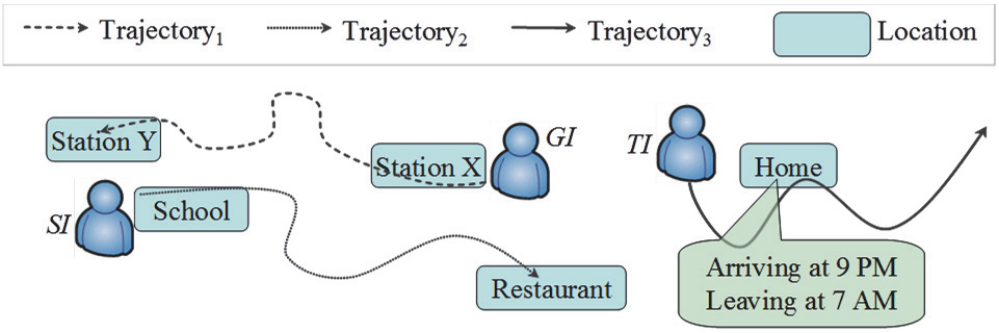

Fig. 1. User movement scenario based on three kinds of intentions.

Corresponding to the three kinds of intentions, movements of users can be viewed as a contexture of the behaviors that are motivated by the aforementioned three kinds of intentions. In other words, the movement of people from one location to another one may be triggered by multiple intentions, inclusive of *GI*, *TI,* and *SI*. However, existing techniques for predicting the next location of a user primarily focus on analyzing *Geographic-triggered Intentions* only. To extract the significant movements motivated by *Geographic-triggered Intentions*, the existing methods usually mine the frequent sequences of locations (i.e., geographical points/regions) from user trajectories. Due to the geographical binding of exact locations, these prediction techniques are only applicable to movements that are motivated by *Geographic-triggered Intentions*. For example, let us suppose that $Trajectory_1$ and $Trajectory_2$ in Fig. 2(a) are historical trajectories and $Trajectory_3$ depicts the current movement of a mobile user. Assume that $Trajectory_3$ is new with respect to the historical trajectory database. As there is no movement pattern similar to the current movement, the traditional prediction techniques will suffer from the low applicability problem (i.e., no pattern is applicable for predicting the current movement of the user).

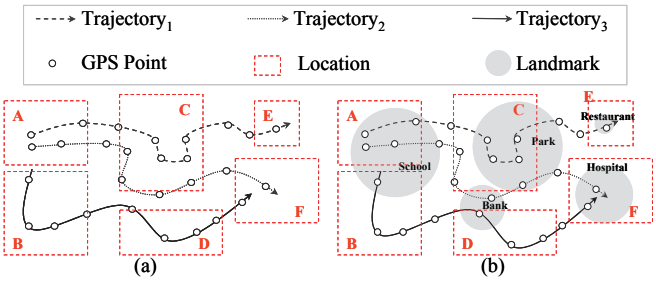

Fig. 2. An example of trajectories and landmarks.

Several works addressed the frequent movements motivated by *Temporal-triggered Intentions* [Giannotti *et al.*, 2006; Monereale *et al.*, 2009]. Unfortunately, these works only roughly calculate the transition time while disregarding the "stay time" and the "arrival time". Moreover, the frequent movements motivated by *Semantic-triggered Intentions* are still not taken into consideration in the representation of users behaviors. This leads to the problem of low precision (i.e., the making of inaccurate location predictions). In Fig. 2(b), it can be seen that both $Trajectory_1$ and $Trajectory_2$ consequently visit Location A and Location C. Suppose the transition time from A to C of the two trajectories are approximated, it is clear that the destinations of $Trajectory_1$ and $Trajectory_2$ are quite different. $Trajectory_3$ and $Trajectory_2$ share the same semantic tags, i.e., from School to Bank. Thus, we observe that $Trajectory_2$ and $Trajectory_3$ can be denoted by the same sequence, i.e., School→Bank→Hospital. As the semantic annotations of user behaviors exhibited in $Trajectory_2$ and $Trajectory_3$ are quite the same, it is therefore more reasonable to consult $Trajectory_3$ in order to make predictions about $Trajectory_2$.

On the basis of the observations prompted by the above examples, we propose a novel approach called *Geographic-Temporal-Semantic based Location Prediction (GTS-LP)* that predicts next locations of users based on the three kinds of intentions. As shown in Equation (1), given a set of users U and a set of locations L, the problem of location prediction can be formulated as an estimation of the probability of a given user visiting a given location based on his/her current movement.

$$f(l \mid u,t) \rightarrow [0,1], \text{where } u \in U, l \in L \text{ and } t \text{ is } u\text{'s current movement} \quad (1)$$

Hence, location prediction can be addressed as a historical movement matching problem. In addition, the question of how to extract significant movements to support the prediction based on heterogeneous moving intentions is also a critical and challenging issue. To support location prediction based on historical movements of users triggered by *GI*, *TI,* and *SI*, we design a novel frequent pattern, called GTS Pattern, to match current movements of users. In pattern mining, the fundamental issue is to identify and efficiently extract representative patterns from the heterogeneous trajectories of users. The extraction of representative patterns is important because those patterns have a direct impact on the efficacy of the prediction task. As mentioned earlier, considering only movement that is triggered by GI does not work well. However, there is no existing work on the mining of frequent patterns that deals with all behaviors triggered by *GI*, *TI,* and *SI*. Moreover, there is no location prediction model that makes predictions based on historical movements of users triggered by *GI*, *TI,* and *SI*.

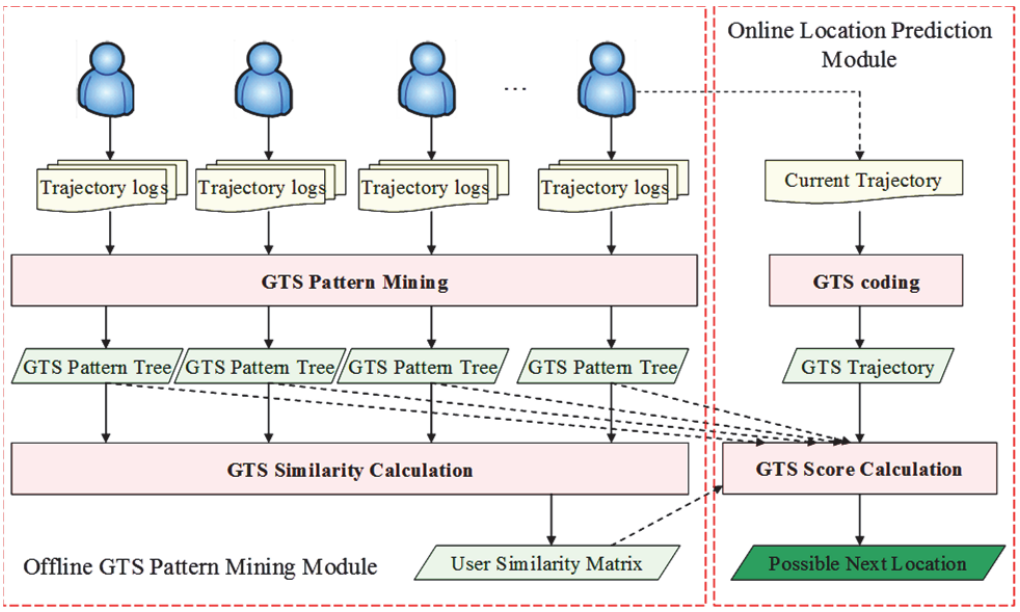

Fig. 3. Flow of data processing in GTS-LP.

To address the above-mentioned problem, we explore movements of users triggered by *GI*, *TI,* and *SI*, and seek representative trajectory patterns from moving intentions of the users. In contrast to conventional location prediction techniques that are based solely on the trajectories motivated by *Geographic-triggered Intentions*, we also utilize trajectories motivated by *Temporal-triggered Intentions* and *Semantic-triggered Intentions* to predict the next location. Our *GTS-LP* approach follows the user-based collaborative filtering framework that comprises 1) an offline frequent pattern mining module (called *GTS Pattern Mining*), and 2) an online location prediction module (called *GTS-based Location Prediction*). Fig. 3 shows the flow of processing in *GTS-LP*. To discover GTS Patterns for prediction of the next location, GTS Pattern Mining explores all the semantic, geographical, and temporal aspects of the mobile user activities being captured in the trajectories. The

GTS Pattern Mining Module includes two main steps: (i) GTS Pattern Discovery—the extraction of individual frequent behaviors of users in the forms of GI, TI, and SI trajectories; and (ii) GTS Similarity Calculation—the calculation of similarity based on their individual GTS Patterns. Since the main idea underlying the user-based collaborative filtering framework is the prediction of behavior of a user in accordance with similar behaviors of users, user similarity plays a crucial role in the prediction model. As shown in Fig. 3, discovered GTS Patterns of each user are utilized in the construction of a pattern tree in order to support efficient GTS-based Location Prediction. In the online module, we propose a scoring function to evaluate the probability of a location that may be the next location. Again, we consider not only geographic property but also semantic and temporal properties for the scoring function. To make a prediction, the location with the highest score is predicted as the next location the user will visit.

This research work makes a number of significant contributions, which are summarized as follows:

- We propose a novel approach named *GTS-LP* for mining and prediction of mobile users' movement behavior. The problems and ideas resolved and proposed, respectively, in *GTS-LP* have not been explored previously in the research community.
- We define a new frequent pattern, called GTS Pattern, to represent mobile users' frequent movements in terms of Geographic-triggered, Temporal-triggered, and Semantic-triggered Intentions.
- We develop a data mining algorithm, called GTSP-Miner, to discover GTS Patterns. It comprises two modules—a parameter-less hierarchical clustering algorithm for mining frequent semantic locations and a probabilistic model for discovering the temporal interval of a location.
- We develop a new index structure based on prefix tree to represent GTS Patterns in a compact form in order to facilitate efficient prediction computation.
- To sustain user-based collaborative filtering, we utilize the discovered individual GTS Patterns for each pair of users to obtain similitude of the users.
- On the basis of the GTS Patterns, we propose a novel location prediction strategy that takes into account all Geographic-triggered, Temporal-triggered, and Semantic-triggered Intentions to predict next location of a user.
- We use various real datasets to evaluate the performance of our proposal in a series of experiments. The results show that our proposed framework significantly outperforms other location prediction techniques in terms of precision and coverage.

The remainder of this paper is organized as follows. We briefly review related work in Section 2 and describe our proposed GTS Pattern Mining and GTS Pattern Tree construction approach in Section 3. Next, our proposed GTS Similarity Calculation and GTS Score Calculation methods are detailed in Sections 4 and 5, respectively. The performance of our proposal in an empirical evaluation study is discussed in Section 6. Finally, conclusions and directions for future work are given in Section 7.

2. RELATED WORK

The problem of predicting the next location to which a mobile user will move has received much research interest in recent years. Prediction techniques developed for this problem domain can be broken down into two steps: (i) user behavior mining and (ii) prediction model building.

2.1 User Behavior Mining

In the user behavior mining step, there are two viewpoints for illustrating movements of users—namely, random walk and frequent pattern. Researchers who utilize random walk to model movements of users believe that most movements of users follow some random regulation. Jiang *et al.* [2008] studied the trajectories of 50 taxicabs and found that the trajectories move according to the Lévy flight behavior. This finding can be used to model and predict the movement of taxicabs. However, it is obvious that the movement of taxicabs can reflect only a tiny part of human mobility. Accordingly, González *et al.* [2008] studied 100,000 trajectories provided by 206 mobile users. They found that the Lévy flight behavior could not explain mobility of users well but the radius of gyration for each user could be considered for user mobility modeling. Song *et al.* [2010] discussed several random walk behaviors for modeling and classifying movements of users. They concluded that only 93% of short-term mobility of users can be predicted. In other words, random walk-based predictors do NOT work well for long-term trajectory prediction.

Conversely, several data mining researchers believe that human mobility is not always random. They believe that regular pathing can be used in the prediction of next movement of a user. To extract the regular part from movements of users, three kinds of frequent patterns can be utilized: (i) mobile sequential pattern [Yavas *et al.*, 2005; Morzy, 2006; Morzy, 2007; Jeung *et al.*, 2008], (ii) spatial-temporal sequential pattern [Monereale *et al.*, 2009; Li *et al.*, 2011], and (iii) semantic-geographic pattern [Alvares *et al.*, 2007; Eagle *et al.*, 2007; Bogorny *et al.*, 2009; Noulas *et al.*, 2011]. The mobile sequential pattern considers a trajectory as a sequence of locations and thus uses the sequential frequent patterns mined from a historical set of trajectories. Jeung *et al.* [2008] proposed an innovative approach to forecast the future locations of a user that consists of combining predefined motion functions with the movement patterns of the user. The motion functions capture object movements as sophisticated mathematical formulas of linear or nonlinear models. To discover mobile sequential patterns, the movement patterns are extracted by means of a modified version of the Apriori algorithm. Yavas *et al.* [2005] and Morzy [2006] proposed several methods in which modified versions of the Apriori algorithm are used to generate association rules for an individual user. Such rules can reflect frequent co-occurrences of locations in the movement of an individual user, i.e., the places users always visit in one movement. In order to select the rule used for the prediction, they take into consideration the notions of support and confidence. The support of each candidate is computed by a distance based on the notion of string alignment. Morzy [2007] subsequently used a modified version of the *PrefixSpan* algorithm [Pei *et al.*, 2001] to discover frequent movements of users. Such a pattern can not only reflect co-occurrence of locations but also the consequence of location, i.e., the place users always visit after visiting somewhere else.

To improve mobile sequential pattern, it is argued that the temporal information in the spatial-temporal sequential pattern reflects crucial human mobility and thus exploits the frequent temporal behavior of mobile users. Giannotti *et al.* [2006] proposed a kind of spatial-temporal sequential pattern, called T-pattern, which contains two kinds of moving intentions—Geographic-triggered Intentions and Temporal-triggered Intentions. A T-pattern is a sequential pattern extended with a temporal property (i.e., travel time). To discover the temporal patterns, Giannotti *et al.* [2006] proposed an effective strategy to map temporal information in the space R^n and to discover the dense hypercube from the R^n space. Li *et al.* [2011] proposed two kinds of trajectory patterns—a periodic behavior pattern and a swarm pattern—and

developed a reference location-based method for mining periodic behavior patterns. The reference location-based method detects the reference locations, discovers the periods in complex movements, and then finds periodic patterns using hierarchical clustering. They also developed an efficient method for mining swarm patterns in which flexible moving object clusters are uncovered by relaxing the popularly-enforced collective movement constraints.

As mentioned in the introduction, many behaviors of users are semantic-triggered. This means that the above user behavior mining methods can only partially reflect movements of users. With regards to the reflection of semantic-triggered movement, Alvares *et al.* [2007] and Bogorny *et al.* [2009] are some of the studies on semantic trajectory data mining that have appeared in the literature. Alvares *et al.* [2007] proposed to explore the geographic and semantic properties by mining semantic trajectory patterns from location histories of mobile users. First, they discover the *stops* of each trajectory and map these stops to semantic landmarks. They then apply a sequential pattern mining algorithm to this sequence dataset to obtain frequent patterns, namely, semantic trajectory patterns, to represent the frequent semantic behaviors of mobile users. Bogorny *et al.* [2009] take hierarchical geographic and semantic properties into consideration in order to discover patterns that are more interesting. However, because the notion of stops in these works only takes the aspect of "stay" into account rather than the positions of these stops in geographic space, many unknown stops are generated. For example, as shown in Fig. 4, $stop_{1c}$, $stop_{2c}$, and $stop_{3b}$ are not associated with any semantic landmark and are thus marked as Unknown. Hence, $Trajectory_1$ is transformed to the sequence <School, Park, Unknown, Restaurant>. From the Fig. 4, it is clear that $stop_{1c}$ is near a Restaurant. Thus, by taking into account the geometric distribution of these stops, $stop_{1c}$ and $stop_{1d}$ are grouped together such that $Trajectory_1$ is transformed to the sequence <School, Park, Restaurant>.

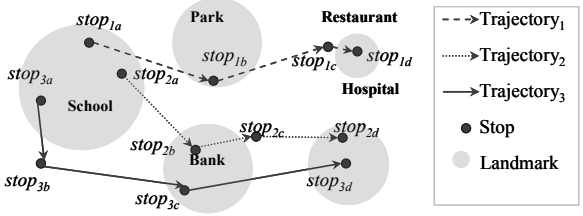

Fig. 4. An example of semantic trajectory.

2.2 Prediction Model Building

Existing studies on user location prediction could be classified into three categories: 1) Those using only a user's own data, 2) Those using the data generated by crowds and 3) hybrid methods using both kinds of data. In the first category of studies, they utilize only a user's own data to predict the next location and focus only on historical movements of users, such as some trajectory simulations [Jiang *et al.*, 2008; González *et al.*, 2008; Eagle *et al.*, 2009; Song *et al.*, 2010]. In [Eagle *et al.*, 2009], the prediction model is based on an eigenvector space to model regular movement of users for predicting next location modeling. However, such a prediction model does not consider historical movements of users. It always leads to low coverage of prediction. For example, even though the user has visited many locations, there must be some places the user has never been to. As a result, the prediction model is not applicable for predicting the locations the user and his friends have never been to

before. Hence, the methods using only a user's own data usually do not work well in dealing with the location prediction problem.

The second category of studies consider only the datasets generated by crowds for next location prediction modeling, based on approaches such as some probability distribution models [Backstrom *et al.*, 2010; Noulas *et al.*, 2011] or location recommenders [Ge *et al.*, 2010; Liu *et al.*, 2010; Zhuang *et al.*, 2011; Ge *et al.*, 2011]. In [Backstrom *et al.*, 2010], the prediction model is based on a social-spatial approximation which utilizes current GPS coordinates of user's friends to estimate GPS coordinate of the user. In [Zhuang *et al.*, 2011], the recommender leverages the rich context signals on the mobile device (i.e., user and sensory context, such as user click-through, geo-location, and time) to rank the location tailored to user's preference. However, these kinds of recommenders do NOT consider current movement of users. It leads to low precision in prediction. For example, even though the user frequently visits a gym, the probability for him to visit the gym after visiting the swimming pool must be very low. However, if we use these recommenders for predicting next location of users, the gym is usually predicted as next location of the users after he visits a swimming pool. As the result, those historical-oriented methods usually do NOT work well for dealing with location prediction problem.

The third category of studies precisely predicts next location of users using hybrid approaches [Monereale *et al.*, 2009; Ying *et al.*, 2011; Wei *et al.*, 2012; Xue *et al.*, 2013]. Monereale *et al.* [2009] have proposed a hybrid method which not only considers a user's own data (his/her current movement) but also utilizes the data generated by crowds. The prediction extracts T-patterns [Giannotti *et al.*, 2006] from trajectory of users to match current movement of a user. As mentioned earlier, the T-patterns are mined for representing Geographic-triggered Intentions and Temporal-triggered Intentions. In other words, the prediction can NOT deal with the next location motivated by Semantic-triggered Intentions. To address this issue, Ying *et al.* [2011] have proposed a novel prediction model by matching current movement of a user to discovered semantic trajectory patterns [Bogorny *et al.*, 2009]. However, the prediction model focuses only on movements of users motivated by Semantic-triggered Intentions and Geographic-triggered Intentions. Consequently, the prediction model can NOT deal with the problem of predicting the next location motivated by Temporal-triggered Intentions.

3. GEOGRAPHIC-TEMPORAL-SEMANTIC PATTERNS MINING

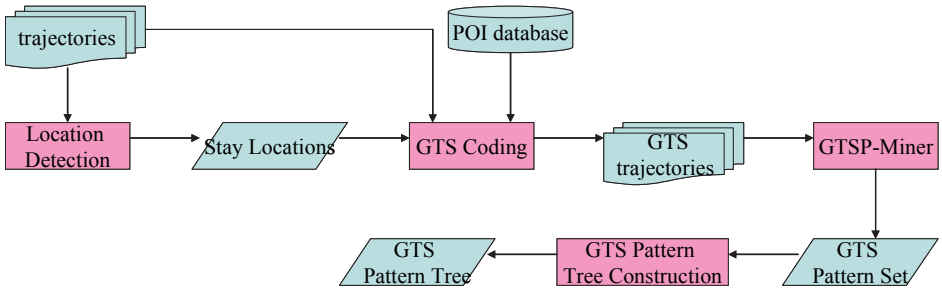

Fig. 5. The Geographic-Temporal-Semantic Patterns mining framework.

In this section, we propose a new type of pattern, called *GTS (Geographic-Temporal-Semantic) Pattern*, to represent users' frequent movement behaviors by considering all the three kinds of user intentions mentioned previously, i.e., geographic-triggered, temporal-triggered and semantic-triggered intentions. In contrast to the conventional

frequent pattern, which can only represent one part of the three kinds of intentions, we take into account the semantic and temporal properties in trajectories to illustrate movements of users. As shown in Fig. 5, we first detect stay locations from trajectories of users; then, we transform each trajectory to a GTS Trajectory. We call this step GTS Coding. We also developed a data mining method, called *GTSP-Miner*, to discover GTS Patterns from transformed GTS Trajectories and, to make the prediction phase efficient, we adopted a prefix tree, called GTS Pattern Tree, to compactly represent a collection of GTS Patterns.

3.1 Location Detection

In this subsection, we explain how each trajectory is transformed to a GTS Trajectory. Unlike some traditional location prediction models that predict pre-determined locations, we prefer to predict application-dependent locations because pre-determined locations are always independent from the distribution of the trajectories. In the illustration shown in Fig. 6, pre-determined locations cover only one trajectory. Consequently, no location prediction model can deal with such a situation. In addition, we argue that most of the activities carried out by a mobile user are usually performed when the user stops. For example, a user may stop at a café to have coffee. As mentioned earlier, the goal of this study is to support location-based services. Therefore, in this paper, we focus only on predicting where a user will go to and stay. Thus, to detect application-dependent locations, we first detect the stay points of individual users, after which we apply a clustering algorithm on stay points of all users and group them as locations.

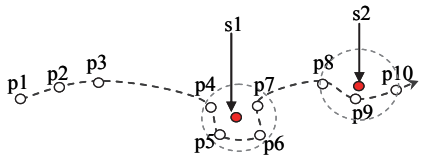

Fig. 6. An example of stay point detection.

3.1.1 Stay Point Detection. Similar to the approach taken by Zheng *et al.* [2011], we consider a stay location to be a region where many users stay. Thus, before carrying out stay location detection, we first detect the regions, called stay points, where a user has stayed. In Zheng *et al.*'s approach, only velocity (i.e., time and distance) is considered as a factor of "stay". However, we argue that direction change is also an important factor for detecting where the user stays. The extraction of a stay point is controlled by three parameters—a time threshold (τ), a distance threshold (ε), and a direction change threshold (δ). Formally, a single stay point s can be regarded as a virtual location characterized by a group of consecutive GPS points $G = \{p_m, p_{m+1}, \ldots, p_n\}$ such that

1) $Distance(p_m, p_i) \leq \varepsilon$,
2) $Time\ Difference(p_m, p_n) \geq \tau$, and
3) $Direction\ Difference(p_i, p_{i+1}) \geq \delta$,

where $\forall m < i \leq n$. A stay point s is conditioned by G, ε, τ, and δ, respectively representing the average latitude and longitude of the collection G, and the timestamp of p_m and p_n represents arrival and leaving times of a user on s. As illustrated in Fig. 6, p1→ p2 →…→ p10 forms a GPS trajectory and a stay point can be constructed by points {p4, p5, p6, p7}. Clearly, if we avoid the direction change factor, another stay point may be constructed by points {p1, p2, p3}. However, such

"stay" behavior would be obtained due to traffic events, such as a traffic jam or traffic signals. In other words, such a stay point is not a semantic-related region. Therefore, it is necessary to take into account direction change at the stay point.

3.1.2 Grouping Stay Points. To discover the region where most people will stay, we make use of these stay points to form stay locations. We could perform a density-based clustering algorithm on the stay points to detect the stay locations. However, the number of stay points may not reflect the popularity of a region where most people will stay as it will be affected by the duration of the trajectory log and the liveliness of the user. However, we expect that the location we find will be where many users have stayed. Thus, to deal with this problem, we use the P-DBSCAN algorithm [Joshi *et al.*, 2009] with these stay points to form stay locations. Unlike traditional density-based algorithms, P-DBSCAN determines the density by the number of people instead of by the number of points. For example, in Fig. 7 (in which the different shapes represent different users), the traditional density-based algorithms would determine both the density of Fig. 7(a) and that of Fig. 7(b) to be 14; but P-DBSCAN determines the density of Fig. 7(a) to be three and that of Fig. 7(b) to be one.

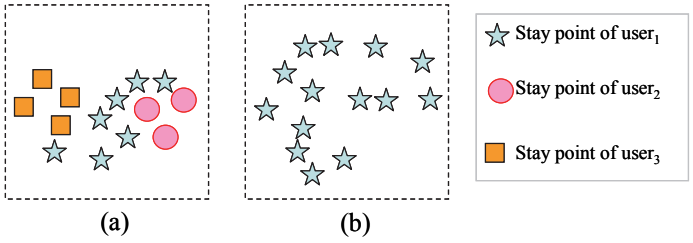

Fig. 7. An example of P-DBSCAN in action.

3.2 Location Detection

Intuitively, after stay points are detected from a trajectory, the trajectory can be transformed to a sequence of stay points. Thus, we transform each trajectory into a stay point sequence after stay point detection. The trajectory p1→ p2 →…→ p10, in Fig. 6, for example, is transformed to s1→ s2. To represent the moving behaviors—GI, TI, and SI—we perform GTS Coding on each stay point sequence to transform it into a GTS Trajectory. The GTS Coding comprises three parts: *Location tagging, Semantic tagging,* and *Temporal Tagging.*

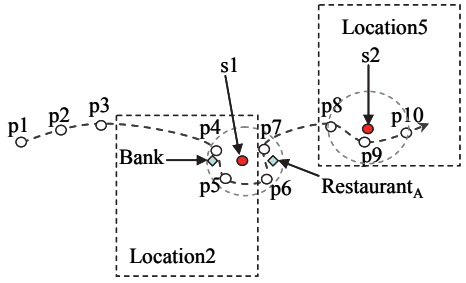

Fig. 8. An example of Temporal Semantic Coding

3.2.1 Location Tagging. As mentioned in the Location Detection section, a stay location is determined by clustering stay points. This means that each stay point corresponds to exactly one stay location. Generally, the location represents the region

in which the stay point lies. Therefore, to represent the moving behaviors that are motivated by GI, we tag each stay point by its location ID. Thus, we transform each stay point sequence into a stay location sequence. For example, the stay point sequence s1→ s2 in Fig. 8, is transformed to location sequence location2→ location5.

3.2.2 Semantic Tagging. We use a POI database and the activity label of the trajectory to calculate the probability of each semantic tag to each stay point. The POI database is a customized spatial database that stores the semantic category of landmarks collected from Google Maps (alternatively, a gazetteer can be used as a general-purpose POI database for this operation.) In our POI database, we store landmarks, their geographic scopes, and the associated semantic tag(s). In this paper, we use some general categories of the landmarks as their semantic tags. Since we need to precisely represent the possibility of each semantic tag for a user in the location, we still use the stay point of a trajectory to determine the semantic tags of the location that is passed by the trajectory. Thus, we construct a semantic vector for each stay point according to the landmarks falling in the stay point. As mentioned earlier, the semantic tag of a landmark for a user is always related to the activity (or purpose) of his/her trajectory. We utilize the co-occurrences of the activity label of the trajectory and the semantic tags of the landmarks to design the semantic vector for a stay point as follows:

Definition 3.1 (Semantic vector). Given a LBSN website and a trajectory with label *tl*, the probability of the semantic tag *st* in a stay point of the trajectory is

$$\Pr(st) = \begin{cases} \dfrac{|T_{st} \cap T_{tl}|}{|T_{tl}|}, & \text{if } \exists \text{ a the landmarks fallen in the stay point is labeled as } st \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

where T_{tl} is the collection of trajectories with activity label *tl*, which are collected from the LBSN website, and T_{st} is the collection of trajectories, which are retrieved by the query term *tl* from the LBSN website.

Example 3.2 In Fig. 8, the semantic tag of the landmark Bank is "Bank" and that of the landmark Restaurant$_A$ is "Restaurant". Suppose that the activity label of the trajectory is "Shopping" and there are four unique landmark categories, "Restaurant", "School", "Park", and "Bank", in a POI database. This means that the semantic vector would be represented as <*Pr(Restaurant)*, *Pr(School)*, *Pr(Park)*, *Pr(Bank)*>. Next, suppose that the number of query results is as shown in Table I, and the number of trajectories with the label "Shopping" is 12000. Since s1 overlaps the landmarks Restaurant$_A$ and Bank, the semantic vector of s1 is calculated as <3000/12000, 0, 0, 1000/12000>.

Table I. Query results for trajectories with the label "Shopping".

Semantic tag	Number of Trajectories
Restaurant	3000
School	1
Park	10
Bank	1000

It is possible that a stay point overlaps none of the landmarks. For example, in Fig. 8, there is no landmark overlapping s2. In this case, we consider all possible

reasons why the user may stay. Thus, we assign the vector <3000/12000, 1/12000, 10/12000, 1000/12000> to s2. After assigning semantic tags to the stay points, a stay point sequence can be transformed to a sequence of semantic-wise stay points. For example, the location sequence location2→ location5 is transformed to (location2, <3000/12000, 0, 0, 1000/12000>)→ (location5, <3000/12000, 1/12000, 10/12000, 1000/12000>).

3.2.3 Temporal Tagging. Finally, we use the stay location to which the stay point belongs to represent the geographic and semantic properties. In addition, the arrival and departure times on the stay point are used to represent the temporal information. This representation is called GTS Trajectory. In Fig. 8, for example, the sequence (location2, <3000/12000, 0, 0, 1000/12000>)→ (location5, <3000/12000, 1/12000, 10/12000, 1000/12000>) can be transformed to (Location2, [p4.T p7.T], <3000/12000, 0, 0, 1000/12000>)→ (Location5, [p8.T p10.T], <3000/12000, 1/12000, 10/12000, 1000/12000>), where pi.T denotes the timestamp of pi $\forall 1 \le i \le 10$. Here, for the sake of readability, we represent each GTS Trajectory as (Location ID, [Stay time], <semantic vector>) [transition time] (Location ID, [Stay time], <semantic vector>)…. For example, the GTS Trajectory (Location2, [p4.T p7.T], <3000/12000, 0, 0, 1000/12000>)→ (Location5, [p8.T p10.T], <3000/12000, 1/12000, 10/12000, 1000/12000>) can be represented as (Location2, [p7.T- p4.T], <3000/12000, 0, 0, 1000/12000>) [p8.T- p7.T] (Location5, [p10.T- p8.T], <3000/12000, 1/12000, 10/12000, 1000/12000>).

3.3 Definition of GTS Pattern

After GTS Coding, each trajectory is transformed to a GTS Trajectory, which contains geographic, temporal, and semantic properties. On the basis of these GTS Trajectories, the GTS Pattern is abstracted using aggregating support, as formally stated by the following definition:

Definition 3.3 (s-containment ($\subseteq_s$)). Given a GTS Trajectory T1 with length n, a GTS Trajectory T2 with length m (where $n \le m$), a semantic vector threshold θ_v, a stay time threshold θ_s, and transition time threshold θ_t, we say that T1 is s-contained in T2, denoted $T1 \subseteq_s T2$, if and only if there exists a sequence of integers $0 \le i_0 < \ldots < i_n \le m$ such that

1) T1(k).Location ID = T2(i_k). Location ID,
2) *Cosine*(T1(k).Semantic Vector, T2(i_k).Semantic Vector) $\ge \theta_v$
3) | T1(k).Stay Time – T2(i_k).Stay Time | $\le \theta_s$,
4) |(T1(k), T1(k+1)).Transition Time – (T2(i_k), T2(i_{k+1})).Transition Time | $\le \theta_t$,

where $\forall 1 \le k \le n$.

Example 3.4 Given a GTS Trajectory T1 = (Loc1, [5], <0.1, 0.0, 0.9>) [15] (Loc9, [7], < 0.1, 0.5, 0.4>) and a GTS Trajectory T2 = (Loc1, [5], <0.0, 0.2, 0.8>) [6] (Loc7, [5], <0.1, 0.9, 0.0>) [10] (Loc9, [8], <0.1, 0.4, 0.5>), and semantic vector threshold θ_v set at 0.5, stay time threshold θ_s set at one, and transition time threshold θ_t set at 10, we get

1) T1(1).Location_ID = T2(1).Location_ID = Loc1 and T1(2).Location_ID = T2(3).Location_ID = Loc9
2) *Cosine*(T1(1).Semantic Vector, T2(1).Semantic Vector) = *Cosine*(<0.1, 0.0, 0.9>, <0.0, 0.2, 0.8>) = 0.59 $\ge \theta_v = 0.5$ and *Cosine*(T1(2).Semantic Vector, T2(3).Semantic Vector)

= *Cosine*(<0.1, 0.5, 0.4>, <0.1, 0.4, 0.5>) = 0.98 ≥ θ_v = 0.5

3) | T1(1).Stay_Time – T2(1).Stay_Time | = | 5 – 5 | ≤ θ_s
= 1 and | T1(2).Stay_Time – T2(3).Stay_Time | = | 7 – 8 | ≤ θ_s = 1

4) | (T1(1), T1(2)).Transition_Time – (T2(1), T2(3)).Transition_Time |
= | 15 – (6+5+10) | ≤ θ_t=10

Hence, T1 $\subseteq_s$ T2 holds.

Definition 3.5 (s-support, Frequent GTS Trajectory). Given a GTS Trajectory set D, a stay time threshold θ_s, transition time threshold θ_t, and a minimum support θ_m, we define the s-support of a GTS Trajectory T as

$$\text{s-support}(T) = \frac{\left|\{T^* \in D \mid T \subseteq_s T^*\}\right|}{|D|} \tag{3}$$

and say that T is frequent in D if s-support(T) ≥ θ_m.

Example 3.6 Given a GTS Trajectory set D consisting of two GTS Trajectories, T1 = (Loc1, [5], <0.1, 0.0, 0.9>) [15] (Loc9, [7], < 0.1, 0.5, 0.4>)
T2 = (Loc1, [5], <0.0, 0.2, 0.8>) [6] (Loc7, [5], <0.1, 0.9, 0.0>) [10] (Loc9, [8], <0.1, 0.4, 0.5>), and semantic vector threshold θ_v set at 0.5, stay time threshold θ_s set at one, transition time threshold θ_t set at 10, and minimum support θ_m set at 0.3, with T' = (Loc1, [5], <0.0, 0.2, 0.8>) [6] (Loc7, [5], <0.1, 0.9, 0.0>), we get

1) T $\not\subset_s$ T1
2) T $\subseteq_s$ T2

Hence, the s-support(T') is 0.5 and T' is frequent.

However, the GTS Trajectory set can have highly dispersed temporal information and semantic vector values, resulting in all the frequent GTS Trajectory being highly redundant. For example, given the GTS Trajectory (Loc1, [5], <0.1, 0.0, 0.9>) [15] (Loc9, [7], <0.1, 0.5, 0.4>) and the GTS Trajectory (Loc1, [5], <0.0, 0.2, 0.8>) [16] (Loc9, [8] , <0.1, 0.4, 0.5>), it can be seen that the stay time of Loc1 in these two trajectories are both [5]. Similarly, the stay time of Loc9 in these two trajectories are very close. Thus, we can use an interval to aggregate and to represent the stay time and transition time. Moreover, the semantic vectors of Loc1 for the two trajectories are very similar and so are the semantic vectors of Loc9 for the two trajectories. Thus, we can use the mean of the semantic vectors to represent the semantic vectors. Accordingly, the two GTS Trajectories may be aggregated and represented as (Loc1, [5 5], <0.05, 0.1, 0.85>) [15 20] (Loc9, [7 8], <0.1, 0.45, 0.45>).

Definition 3.7 (GTS Sequence, Semantic-Temporally Belong ($\in_{ST}$)). Given a GTS Trajectory T1 and a sequence T that uses the time interval instead of the temporal part of GTS Trajectory, we say that T1 semantic-temporally belongs to T, denoted T1 $\in_{ST}$ T, if and only if

1) T1(k).Location ID = T(k). Location ID
2) *Cosine*(T1(k).Semantic Vector, T2(k).Semantic Vector) ≥ θ_v
3) T1(k).Stay Time ∈ T(k).Stay Time Interval
4) (T1(k), T1(k+1)).Transition Time ∈ (T2(k), T2(k+1)).Transition Time Interval,

where ∀1 ≤ k ≤ n.

Example 3.8 Given a GTS Trajectory T1 = (Loc1, [5], <0.1, 0.0, 0.9>) [15] (Loc9, [7], <0.1, 0.5, 0.4>) and a GTS Sequence T = (Loc1, [5 5], <0.1, 0.0, 0.9>) [15 20] (Loc9, [6 8], <0.0, 0.55, 0.45>), we get

1) T1(1).Location ID = T(1). Location ID = Loc1 and T1(2).Location ID = T(2). Location ID = Loc9
2) *Cosine*(T1(1).Semantic Vector, T(1).Semantic Vector) = *Cosine*(<0.1, 0.0, 0.9>, <0.1, 0.0, 0.9>) = 1.0 ≥ θ_v = 0.5 and *Cosine*(T1(2).Semantic Vector, T(2).Semantic Vector) = *Cosine*(<0.1, 0.5, 0.4>, <0.0, 0.55, 0.45>) = 0.99 ≥ θ_v = 0.5
3) T1(1).Stay Time = 5 ∈ T(1).Stay Time Interval = [5 5] and T1(2).Stay Time = 7 ∈ T(2).Stay Time Interval = [6 8]
4) (T1(1), T1(2)).Transition Time = 15 ∈ (T(1), T(2)).Transition Time Interval = [15 20]

Hence, T1 $\in_{\mathbf{T}}$ T holds.

Definition 3.9 (s-aggregating-support). Given a GTS Trajectory set D, a stay time threshold θ_s, transition time threshold θ_t, and a minimum support θ_m, we define the s-aggregating-support of a GTS Sequence S as

$$\text{s - aggregating - support}(S) = \frac{\left| \bigcup_{T^* \in_{ST} S} \{T^* \in D \mid T \subseteq_S T^*\} \right|}{|D|} \tag{4}$$

and say that S is a frequent in D if s-aggregating-support(S) ≥ θ_m.

Example 3.10 Given a GTS Trajectory set D consisting of two GTS Trajectories, T1 = (Loc1, [5], <0.1, 0.0, 0.9>) [15] (Loc9, [7], < 0.1, 0.5, 0.4>)
T2 = (Loc1, [5], <0.0, 0.2, 0.8>) [6] (Loc7, [5], <0.1, 0.9, 0.0>) [10] (Loc9, [8], <0.1, 0.4, 0.5>), and semantic vector threshold θ_v set at 0.5, stay time threshold θ_s set at one, transition time threshold θ_t set at 10, and minimum support θ_m set at 0.3, with S = (Loc1, [5 6], <0.0, 0.2, 0.8>) [14 22] (Loc9, [7 8], < 0.1, 0.5, 0.4>), we get

1) $S \subseteq_s$ T1
2) $S \subseteq_s$ T2

Hence, the s-aggregating-support(S) is 1.0 and S is frequent.

From the above definition, it is easy to see that the value of s-aggregating-support is proportional to the length of time interval in the discovered patterns. In particular, setting s-aggregating-support with a large value will result in a long and imprecise time interval for most discovered patterns. Thus, how to determine the time interval is an important issue in temporal pattern mining. Fortunately, additional information, such as semantic and geographic information, can be utilized for time interval detection. We argue that the stay time and the transition time are related to the semantic information of the locations because the semantic information can be used to ascertain the "purpose" of the stay or transit between the locations. Moreover, in Probability Theory, the exponential distribution [Papoulis *et al.*, 2002; Ross, 2004] describes the time between events in a Poisson process, i.e., a process in which events occur continuously and independently at a constant average rate. Therefore, we can treat the "leave" and "arrive" events as Poisson processes, and the stay time and the transition time can be modeled by exponential distribution. In statistics, a confidence interval [Papoulis *et al.*, 2002; Ross, 2004] is used to indicate the reliability of an

estimate, which is determined by the significance level [Papoulis *et al.*, 2002; Ross, 2004].

LEMMA 3.11. If the assumption that the stay time and the transition time can be modeled by exponential distribution holds, we can use the confidence interval of the mean of the exponential distribution to represent the possible interval of stay time (or transition time). Given a significance level α and stay time (or transition time) X_1, X_2, …, X_n, which corresponds to a user staying in a location (or transiting between two locations), the stay time interval of the stay time corresponding to the user staying in the location (or transiting between the two locations) is given as

$$\left[\frac{2\sum_{i=1}^{n} X_i}{\chi^2_{\alpha/2,2n}}, \frac{2\sum_{i=1}^{n} X_i}{\chi^2_{1-\alpha/2,2n}}\right] \tag{5}$$

where $\chi^2_{p,v}$ is the $100(1-p)$ percentile of the chi-squared distribution with v degrees of freedom.

PROOF. Since the "leave" and "arrive" events are Poisson processes, the stay time and the transition time can be modeled by exponential distribution. If X_1, X_2, …, X_n are independent exponential random variables, each having mean θ, then the MLE of θ is the simple mean $\sum_{i=1}^{n} X_i/n$.

Since $\sum_{i=1}^{n} X_i$ is a gamma random variable with parameter (n, $1/\theta$), this in turn implies that

$$\frac{2}{\theta}\sum_{i=1}^{n} X_i \sim \chi^2_{2n}$$

Hence, for any $\alpha \in (0, 1)$

$$p\left\{\chi^2_{1-\alpha/2,2n} < \frac{2}{\theta}\sum_{i=1}^{n} X_i < \chi^2_{\alpha/2,2n}\right\} = 1-\alpha$$

That is equivalent to

$$p\left\{\frac{2\sum_{i=1}^{n} X_i}{\chi^2_{\alpha/2,2n}} < \theta < \frac{2\sum_{i=1}^{n} X_i}{\chi^2_{1-\alpha/2,2n}}\right\} = 1-\alpha$$

Hence, a $100(1-\alpha)$ percent confidence interval for θ is

$$\theta \in \left[\frac{2\sum_{i=1}^{n} X_i}{\chi^2_{\alpha/2,2n}}, \frac{2\sum_{i=1}^{n} X_i}{\chi^2_{1-\alpha/2,2n}}\right]$$

Thus, we can say that there is a $100(1-\alpha)$ percent confidence to support the stay time corresponding to the user staying in the location (or transiting between the two locations) in the interval. The lemma thus holds. □

Table II. An example of a GTS Trajectory dataset

	GTS Trajectory
T1	(Loc1, [6], <0.0, 0.2, 0.8>) [9] (Loc7, [3], < 0.1, 0.9, 0.0>) [2] (Loc9, [8], < 0.1, 0.4, 0.5 >)
T2	(Loc1, [9], <0.5, 0.0, 0.5>) [10] (Loc7, [1], < 0.4, 0.5, 0.1>) [1] (Loc5, [6], < 0.0, 0.9, 0.1>)
T3	(Loc1, [6], <0.1, 0.2, 0.7>) [9] (Loc7, [3], < 0.1, 0.8, 0.1 >) [4] (Loc9, [7], < 0.1, 0.4, 0.5 >)
T4	(Loc1, [10], <0.5, 0.2, 0.3>) [10] (Loc5, [8], < 0.1, 0.8, 0.1>) [2] (Loc7, [1], < 0.5, 0.5, 0.0>)
T5	(Loc7, [3], < 0.1, 0.8, 0.1 >) [4] (Loc9, [7], < 0.1, 0.4, 0.5 >)

We use Table II as an example dataset to explain how to calculate the confidence interval. There are five GTS Trajectories; let us calculate the stay time interval for Loc7. Two trajectories contain Loc7 (i.e., $n = 5$), and we suppose that their semantic vectors are similar. If we let the significance level be 10% (i.e., $\alpha = 0.1$), then we have $\chi^2_{0.05,10} = 18.31$ and $\chi^2_{0.95,10} = 3.94$. Thus the interval is [2(3+1+3+1+3)/18.31, 2(3+1+3+1+3)/3.94] = [1.20, 5.58].

*Definition 3.12 (**GTS Pattern**).* Given a GTS Trajectory set D, we say that a GTS Sequence P is a GTS Pattern if and only if

1) s-aggregating-support(P) $\geq \theta_m$
2) P(k).Stay Time Interval is the confidence interval of the stay time of the locations that support P(k)
3) (P(k), P(k+1)).Transition Time Interval is the confidence interval of the transition time of the two locations that respectively support P(k) and P(k+1) .

where $\forall 1 \leq k \leq n$.

3.4 GTSP-Miner

As shown in the above definitions, GTS Patterns must be mined from the GTS Trajectory set. To the best our knowledge, there is no existing pattern discovery algorithm that can be directly applied to the GTS Trajectory set to discover GTS Patterns. Therefore, we propose a novel algorithm called *GTSP-Miner* (Fig. 9) that discovers GTS Patterns from GTS Trajectory sets. The design of this algorithm follows the pattern growth strategy [Pei *et al.*, 2001].

```
Input: A GTS Trajectory set D,
       a stay time significance level α_s,
       a transition time significance level α_t,
       a minimum support θ_m,
       a GTS Pattern P with length k
Output: A GTS Pattern Set GTSP
1:     GTSP ← ∅
2:     F1 ← Find Single Frequent Location (D, θ_m)
3:     foreach Location L ∈ F1 do
4:         {(D*, SemanticVector)} ← ProjBySemantic(L, D, θ_m)
5:         foreach (D*, SemanticVector) ∈ {(D*, SemanticVector)} do
6:             {(D**, StayTimeInterval)} ← ProjByStayTime(D*, θ_m, α_s)
7:             foreach (D**, StayTimeInterval) ∈ {(D**, StayTimeInterval)} do
8:                 if k=0 then
9:                     P*←(L[ StayTimeInterval]<SemanticVector>)
10:                    GTSP← GTSP ∪ { P*}
11:                    GTSP← GTSP ∪ GTSP-Miner(D**, α_s, α_t, θ_m, P*)
12:                else
13:                    {(D***, TransitionTimeInterval)} ← ProjByTransitionTime(D*, θ_m, α_t)
14:                    foreach ( D***, TransitionTimeInterval ) ∈ {( D***, TransitionTimeInterval )} do
15:                        P*←P[TransitionTimeInterval](L[ StayTimeInterval]{Semantic})
16:                        GTSP← GTSP ∪ { P*}
17:                        GTSP← GTSP ∪ GTSP-Miner(D***, α_s, α_t, θ_m, P*)
18:                    end
19:                end
20:            end
21:        end
22:    end
23:    return GTSP
```

Fig. 9. The GTSP-Miner algorithm

First, single locations are discovered (Fig. 9, Line 2). For each single location discovered, we use the semantic property to divide the input GTS Trajectory set into

several small GTS Trajectory sets, denoted D^*, and each D^* is given a representative semantic vector (Fig. 9, Lines 3 and 4). For each small GTS Trajectory set obtained by considering the semantic property, we use the stay time to divide it into several smaller GTS Trajectory sets, denoted D^{**}, and each D^{**} is given a representative stay time interval (Fig. 9, Lines 5 and 6). For each small GTS Trajectory set obtained by considering stay time information, there are two cases. One case is the initial case (k = 0), for which we use the frequent single location, representative semantic vector, and representative stay time interval to obtain the GTS Pattern P^* with length 1, and further consider P^* and D^{**} as inputs to the *GTSP-Miner* algorithm (Fig. 9, Lines 7–11). Another is the recursive case, for which we use the transition time to divide the GTS Trajectory set into several smaller GTS Trajectory sets, denoted D^{***}. Each D^{***} has a representative transition time interval (Fig. 9, Lines 12 and 13). For each small GTS Trajectory set obtained by considering the stay time information, we use the input GTS Pattern P with length k, the frequent single location, the representative semantic vector, the representative stay time interval, and the representative transition time interval to obtain the GTS Pattern P^* with length k+1, and further consider P^* and D^{***} as inputs to the *GTSP-Miner* algorithm (Fig. 9, Lines 14–17).

The three subroutines, *ProjBySemantic*, *ProjByStayTime,* and *ProjByTransitionTime* are the core of *GTSP-Miner*. Thus, we describe them in detail below:

3.4.1 ProjBySemantic: This subroutine is used to discover the frequent semantic vectors of a single frequent location in a GTS Trajectory set and to divide that GTS Trajectory set using the frequent semantic vectors discovered. The problem can be viewed as a problem of clustering GTS Trajectories by frequent semantic vector clusters. Thus, we propose a new clustering algorithm to cluster GTS Trajectories using frequent semantic vector clusters. Using the dataset in TABLE II as an example, if we set minimum support θ_m as 0.4, we find that locations Loc1, Loc5, Loc7, and Loc9 are frequent. Then, we perform *ProjBySemantic* subroutine on D for location Loc1 to extract the semantic information as a vector set. Further, we consider the vector set as input and the cosine similarity as the similarity between two vectors to obtain the clusters by applying the hierarchical clustering.

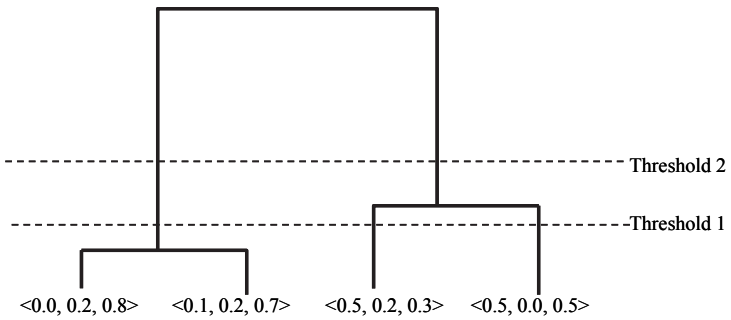

Fig. 10. Example of clustering by Frequent Itemsets

The traditional hierarchical clustering algorithm needs a user specified threshold to split clusters. As shown in Fig. 10, if we use threshold one to obtain clusters, the resulting cluster will be {<0.0, 0.2, 0.8>, <0.1, 0.2, 0.7>}, {<0.5, 0.2, 0.3>} and {<0.5, 0.0, 0.5>}. Since we set the minimum support θ_m as 0.4, we only consider clusters in which the number of vectors is greater than or equal to two. This means that only the

cluster {<0.0, 0.2, 0.8>, <0.1, 0.2, 0.7>} will be considered. However, if we use a threshold of two to obtain clusters, the cluster will be {<0.0, 0.2, 0.8>, <0.1, 0.2, 0.7>} and {<0.5, 0.2, 0.3>, <0.5, 0.0, 0.5>}. To preserve all possible frequent semantic meanings, our algorithm automatically determines the cluster threshold such that the number of frequent clusters is maximized.

We use the top-down strategy (i.e., we initially treat all vectors as a cluster) to split a cluster into several smaller clusters by increasing the similarity threshold while simultaneously counting the number of vectors in each cluster. Thus, we also can determine the number of frequent clusters. Using linear search, we can easily determine the optimum threshold and frequent clusters. The central of a frequent cluster is called the reprehensive semantic vector. For each frequent cluster of location *l* with central *v*, we obtain the (*l, v*)-projected GTS Trajectory dataset. This dataset consists of three parts—Suffix GTS Trajectory, stay time sequence, and transition time sequence. For example, the dataset in Table II becomes the (Loc1, <0.05, 0.2, 0.75>)-projected GTS Trajectory dataset shown in Table III.

Table III. (Loc1, <0.05, 0.2, 0.75>)-projected GTS Trajectory dataset

	Suffix GTS Trajectory	Stay time	Transition time
T1	(Loc7, [3], < 0.1, 0.9, 0.0>) [2] (Loc9, [8], < 0.1, 0.4, 0.5 >)	[6]	[9]
T3	(Loc7, [3], < 0.1, 0.8, 0.1 >) [4] (Loc9, [7], < 0.1, 0.4, 0.5 >)	[6]	[9]

3.4.2 ProjByStayTime: This subroutine is used to obtain the frequent stay time interval of a single frequent semantic vector from the (*l, v*)-projected GTS Trajectory set and to divide the projected GTS Trajectory set by the stay time interval. In fact, both stay time and transition time are only parts of the temporal property. As mentioned earlier, an effective strategy is to use probabilistic exponential distribution to model it. On the basis of the probabilistic exponential distribution, we can easily extract the confidence interval of the stay time. If we use the data in Table III as an example and set the stay time significance level α_s as 0.2, we get the 0.1 percentile $\chi^2_{0.1,4}$ = 1.064 and the (1-0.1) percentile $\chi^2_{0.9,4}$ = 7.779. The confidence interval is calculated as [2(6+6) / 7.779, 2(6+6) / 1.064] = [3.09, 22.56]. It can be seen that the interval contains two points, [6] and [6], and the minimum support θ_m is set at 0.4 (0.4 × 5 = 2). Hence, it is a frequent interval. Therefore, the pattern (Loc1, [3.09, 22.56], <0.05, 0.2, 0.75>) will be mined and the (*l, s, v*)-projected GTS Trajectory dataset obtained as shown in Table IV. Since the length of the pattern is one, there is no prefix pattern. The *ProjByTransitionTime* step will not be performed and further the (Loc1, [3.09, 22.56], <0.05, 0.2, 0.75>)-projected GTS Trajectory dataset is considered as input to the next iteration. Note that it is trivial to see that the interval size is inversely proportional to the significance level. To represent the stay time and transition time precisely, we may set the significance level a higher value.

Table IV. (Loc1, [3.09, 22.56], <0.05, 0.2, 0.75>)-projected GTS Trajectory dataset

	Suffix GTS Trajectory	Transition time
T1	(Loc7, [3], < 0.1, 0.9, 0.0>) [2] (Loc9, [8], < 0.1, 0.4, 0.5 >)	[9]
T3	(Loc7, [3], < 0.1, 0.8, 0.1 >) [4] (Loc9, [7], < 0.1, 0.4, 0.5 >)	[9]

3.4.3 ProjByTransitionTime: This subroutine is used to obtain the frequent transition time interval between two frequent locations from the (*l, s, v*)-projected GTS Trajectory set and to divide the projected GTS Trajectory set by the stay time

interval. Similarly, suppose we execute the *GTSP-Miner* algorithm on the (Loc1, [3.09, 22.56], <0.05, 0.2, 0.75>)-projected GTS Trajectory dataset, as shown in Table IV. After the *ProjBySemantic* and *ProjByStayTime* step, the (Loc1, [3.09, 22.56], <0.05, 0.2, 0.75>) (Loc9, [3.86, 28.20], <0.1, 0.4, 0.5>)-projected GTS Trajectory dataset is obtained, as shown in Table V. To determine the frequent transition time interval from locations Loc1 to Loc9 in T1 and T3, we first aggregate all the transition times from locations Loc1 to Loc9 in T1 and T3, respectively. We then extract the confidence interval of the transition time. For example,. if we set the transition time significance level α_t as 0.4 for Table V, we get the 0.2 percentile $\chi^2_{0.2,4}$ = 1.65 and the (1-0.2) percentile $\chi^2_{0.8,4}$ = 5.99. The confidence interval is calculated as [2(11+13) / 5.99, 2(11+13) / 1.65] = [8.01, 29.09]. The interval contains two points, [11] and [13], and the minimum support θ_m is set at 0.4 (0.4 × 5 = 2). Hence, it is a frequent interval. Thus, the GTS Pattern (Loc1, [3.09, 22.56], <0.05, 0.2, 0.75>) [8.01, 29.09] (Loc9, [3.86, 28.20], <0.1, 0.4, 0.5>) will be discovered. Note that here we consider the "transition" to be a different type of Poisson process from "stay"; therefore, we give another significance level for the transition time interval. It can be seen that there is no suffix GTS Trajectory in the projected GTS Trajectory dataset. Thus, the recursion will stop in this iteration.

Table V. An example of the (l, s, v)-projected GTS Trajectory dataset

	Suffix GTS Trajectory	Transition time
T1	null	[9] + [2] = [11]
T3	null	[9] + [4] = [13]

3.5 Construction of GTS Pattern Tree

After discovering all of the GTS Patterns, to make the prediction phase efficient, we adopted a prefix tree, called the GTS Pattern Tree, to compactly represent a collection of GTS Patterns. (Hereafter the path of a GTS Pattern Tree indicates a decision rule.) The *GTS-Tree* is a kind of decision tree in which each node v consists of six elements—location, stay time interval, transition time interval, semantic vector, support, and child links. Moreover, as mentioned earlier, the matching strategy in our prediction model is the most recent location matching strategy. Thus, we build the tree by reversing the order of the patterns and inserting the reversed patterns into the tree as a path. To simplify the structure of the tree, we merge redundant paths. The patterns mined for the dataset in TABLE II are displayed in TABLE VI. There are four redundant paths: (root)--(Loc9, [4.14, 20.00], <0.1, 0.4, 0.5>), (root)--(Loc9, [3.86, 28.20], <0.1, 0.4, 0.5>), (root)--(Loc7, [1.54, 11.28], <0.1, 0.85, 0.05>), and (root)--(Loc7, [1.69, 8.18], <0.1, 0.83, 0.07>). Since [4.14, 20.00] ⊂ [3.86, 28.20] and Cosine(<0.1, 0.4, 0.5>, <0.1, 0.4, 0.5>) ≥ θ_v, they will be merged as (root)--(Loc9, [3.86, 28.20], <0.1, 0.4, 0.5>). Similarly, the paths (root)--(Loc7, [1.54, 11.28], <0.1, 0.85, 0.05>), and (root)--(Loc7, [1.69, 8.18], <0.1, 0.83, 0.07>) can be merged as (root)--(Loc7, [1.54, 11.28], <0.1, 0.85, 0.05>). Formally, two nodes in the tree can be merged as a single node if and only if

1) they have a common prefix,
2) their locations are the same,
3) their semantic vectors are similar ($\geq\theta_v$),
4) their temporal intervals (both stay time interval and transition time interval) match each other.

Since the tree is used in location prediction, a path with a single element is meaningless. Therefore, we remove all paths that have a single element. The *GTS-Tree* is constructed as illustrated in Fig. 11.

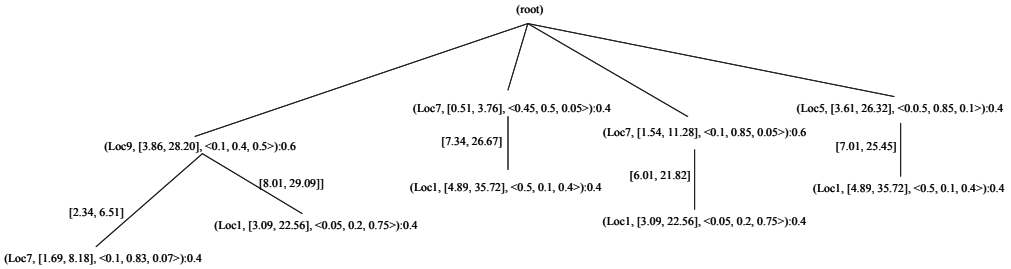

Fig. 11. Example of a GTS Pattern Tree

Table VI. Example of a GTS Pattern

GTS Pattern	Support
(Loc1, [3.09, 22.56], <0.05, 0.2, 0.75>)	0.4
(Loc1, [4.89, 35.72], <0.5, 0.1, 0.4>)	0.4
(Loc5, [3.61, 26.32], <0.05, 0.85, 0.1>)	0.4
(Loc7, [1.69, 8.18], <0.1, 0.83, 0.07>)	0.6
(Loc7, [0.51, 3.76], <0.45, 0.5, 0.05>)	0.4
(Loc9, [4.14, 20.00], <0.1, 0.4, 0.5>)	0.6
(Loc1, [3.09, 22.56], <0.05, 0.2, 0.75>) [8.01, 29.09] (Loc9, [3.86, 28.20], <0.1, 0.4, 0.5>)	0.4
(Loc1, [3.09, 22.56], <0.05, 0.2, 0.75>) [6.01, 21.82] (Loc7, [1.54, 11.28], <0.1, 0.85, 0.05>)	0.4
(Loc7, [1.69, 8.18], <0.1, 0.83, 0.07>)[2.34, 6.51] (Loc9, [4.14, 20.00], <0.1, 0.4, 0.5>)	0.4
(Loc1, [4.89, 35.72], <0.5, 0.1, 0.4>) [7.01, 25.45] (Loc5, [3.61, 26.32], <0.0.5, 0.85, 0.1>)	0.4
(Loc1, [4.89, 35.72], <0.5, 0.1, 0.4>) [7.34, 26.67] (Loc7, [0.51, 3.76], <0.45, 0.5, 0.05>)	0.4

4. GTS SIMILARITY

As mentioned earlier, the main idea underlying a user-based collaborative filtering framework is the prediction of behavior of a user from similar behaviors of users. Therefore, similarity plays a crucial role in the prediction model. We argue that each user's pattern tree represents his/her behavior, which is his/her frequent activity. Intuitively, the next location of a mobile user can be predicted not only from his/her own past movement behavior but also from that of other mobile users that exhibit similar semantic behaviors. In this section, we describe the similarity between two mobile users based on their GTS Pattern Trees. We first propose GTS Similarity to measure the similarity between two paths of two different GTS Pattern Trees. We then extend the GTS Similarity to measure the similarity between two users.

Given two paths for two different GTS Pattern Trees, we argue that they are more similar when they have more common parts. Thus, we first detect the longest common location path to represent the geographical common part. (For example, in Fig. 12, the longest common location path is (Loc1)--(Loc9).) We then calculate the semantic similarity between two paths corresponding to the longest common location path. Next, we define the semantic similarity of two nodes according to the cosine similarity of their semantic vectors. For example, the semantic similarity in Fig. 12 is given by Cosine(<0.05, 0.2, 0.75>, <0.05, 0.2, 0.75>) + Cosine(<0.0, 0.5, 0.5>, <1.0, 0.0, 0.0>). Thus, the semantic similarity is 1.0 + 0.0 = 1.0. Finally, we calculate the temporal similarity between two paths corresponding to the longest common location path. We then define the semantic similarity of two nodes according to the proportion of the intersection of their stay time intervals to the union of their stay time intervals. In Fig. 12, the temporal similarity is ([3.09, 22.56]∩[3.09, 22.56])/([3.09, 22.56]∪[3.09,

22.56]) + ([3.86, 28.20]∩[4.14, 20.00])/([3.86, 28.20] ∪ [4.14, 20.00]). Thus, the semantic similarity is 1.0 + 0.65 = 1.65. Accordingly, the similarity of the two paths is 1.0 + 1.65 = 2.65.

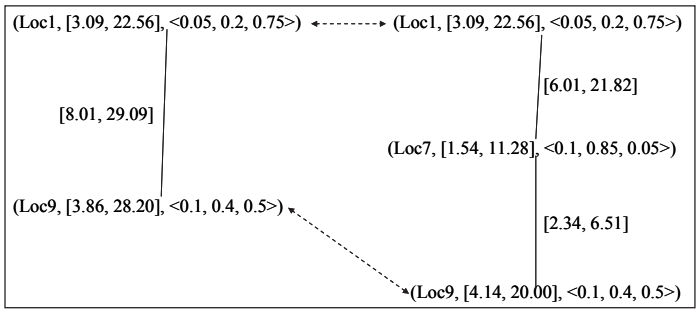

Fig. 12. An example of GTS Similarity

Since a path in the GTS Pattern Tree represents one of real-world moving behaviors of a user, we consider the similarity between two users in terms of the similarity of paths in their GTS Pattern Trees. When there is a strong similarity between the paths in the GTS Pattern Trees of two users, the location prediction model predicts their next locations as the same location. Since a GTS Pattern Tree may possibly possess several paths, we extend GTS Similarity to measure two GTS Pattern Trees. Let $S_U = \{M_1, M_2, ..., M_m\}$ and $S_V = \{M'_1, M'_2, ..., M'_n\}$ be the sets of paths in the GTS Pattern Trees corresponding to users U and V, respectively. The user similarity between U and V is defined by Equation (6):

$$Similarity(U,V) = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} \text{GTS } Similarity(M_i, M'_j)}{m \times n} \tag{6}$$

5. GTS-BASED LOCATION PREDICTION

In this section, we explain how to predict next stay location of a mobile user based on his/her current movement and his/her and similar GTS Pattern Trees of users. We argue that a GTS Pattern Tree provides the geographic, semantic, and temporal properties of most mobile users. Thus, for a given trajectory, we compute the best matching score of all the admissible paths in the pattern tree. For a mobile user who follows a trajectory T, we can detect all of the stay points and try to match them to a path in the pattern tree. Moreover, since our matching strategy is the most recent location matching strategy and the tree is constructed in a reverse pattern, we match the older stay point to the deeper node.

As shown in Fig. 13, for a mobile user who follows a trajectory T, his/her most recent stay point *s2* has already reached node (Loc7, [1.69, 8.18], <0.1, 0.83, 0.07>), i.e., the GPS coordinate stay point is close to stay location Loc1. Since the transition time interval between node (Loc7, [1.69, 8.18], <0.1, 0.83, 0.07>) and node (Loc1, [4.89, 35.72], <0.5, 0.1, 0.4>) is [7.34, 26.67], we seek the previous stay point from the 7.34 to the 26.67 time unit before GPS log. Based on a part of the log, we seek the stay point and match the stay point to child node (Loc7, [1.69, 8.18], <0.1, 0.83, 0.07>). In this way, we can match most of the recent stay points of trajectory T to a path in the tree. We can then compute the matching score between these stay points and the path by summing all of the local scores in the path. For example, in Fig. 13, the matching score between trajectory T and the leftmost path in the GTS Pattern Tree

is the summation of the local scores of node (Loc7, [1.69, 8.18], <0.1, 0.83, 0.07>) and node (Loc1, [4.89, 35.72], <0.5, 0.1, 0.4>). This local score is called *nodeScore,* and it indicates the goodness of a node w.r.t. a stay point. It measures the level of possibility of a node containing the stay point in a trajectory. We define the *nodeScore* to compute the average score of geographic behavior, semantic behavior, and temporal behavior as follows:

$$nodeScore(n,s) = \omega_1 \times GeographicScore(n,s) + \omega_2 \times SemanticScore(n,s) + \omega_3 \times TemporalScore(n,s), \quad (7)$$

where n is the node of the path in the GTS Pattern Tree, s is the stay point of current movement of the user, and $\omega_1 + \omega_2 + \omega_3 = 1$

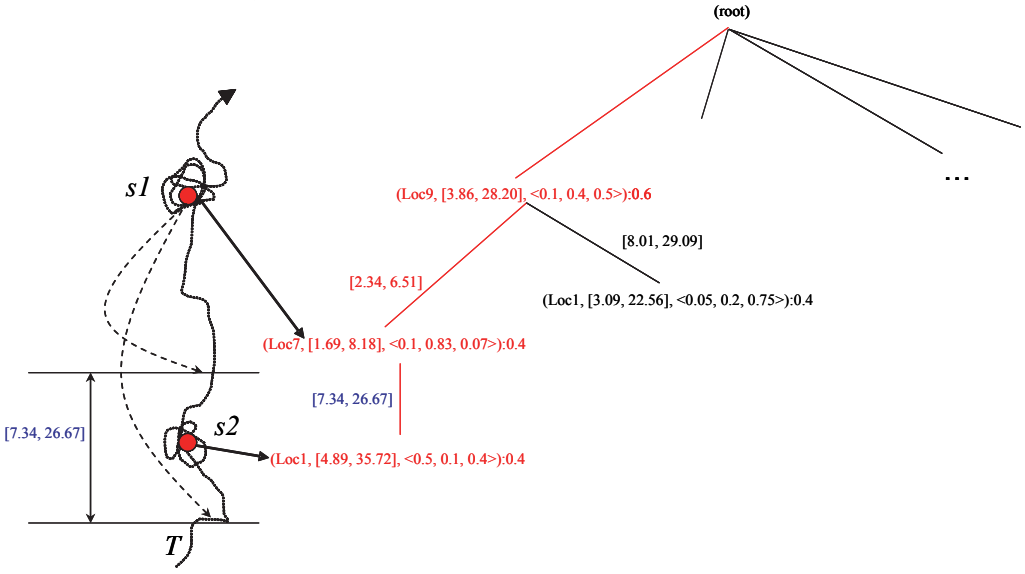

Fig. 13. Example of Path Matching

5.1 GeographicScore

We define the *GeographicScore* of a node according to the spatial distance between the stay point and the spatial region of the location of the node specified by the following three different possible cases:

- The stay point is located in the region: In this, the optimal case, the *GeographicScore* is equal to the support value of the node.
- The stay point is near to the region: In this case, we define the distance δ between the stay point and the region to be smaller than the radius r of the region. The *GeographicScore* is defined as follows:

$$GeographicScore = \text{support} \times \left(1 - \frac{\delta}{r}\right) \quad (8)$$

- The stay point is far away from the region: In this case, we define the distance between the stay point and the region to be greater than the radius of the region. The *GeographicScore* is thus equal to 0.

5.2 SemanticScore

We define the *SemanticScore* of a node according to the cosine similarity between the semantic vector V_1 of the stay point and the semantic vector V_2 of the node as follows:

$$SemanticScore = \text{support} \times Cos(V_1, V_2) \tag{9}$$

Here, the value of each dimension of the semantic vector of the stay point is considered to be the proportion of the representing POI category, and the similarity between the semantic vector of the stay point and the semantic vector of the node is considered to be the cosine similarity of two semantic vectors. For example, in Fig. 14, there are four POIs with the semantic B and C, respectively, and they are close to the stay point. The semantic vector of the stay point can be assigned as <0/4, 3/4, 1/4>. The similarity is Cosine(<0/4, 3/4, 1/4>, <0.1, 0.83, 0.07>) = 0.96. Thus, the *SemanticScore* is 0.4 × 0.96 = 0.384.

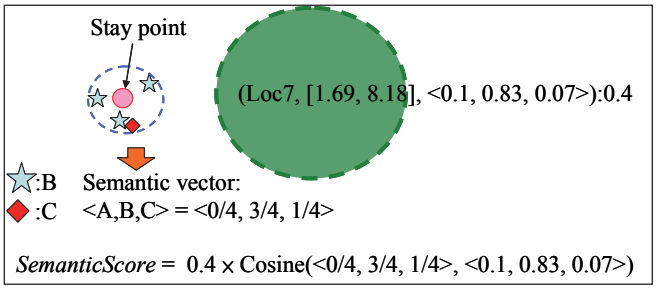

Fig. 14. Example of *SemanticScore*

5.3 SemanticScore

We define the *TemporalScore* of a node according to the temporal distance between the stay point and the node specified by the following three different possible cases:

- The stay time of the stay point belongs to the same stay time interval as the node: In this case, the *TemporalScore* is equal to the support value of the node.
- The stay time of the stay point is approximated to the stay time interval of the node: In this case, we define the difference Δ between the stay time of the stay point and the stay time interval of the node to be smaller than the range of the stay time interval σ of the region. The *TemporalScore* is defined as follows:

$$TemporalScore = \text{support} \times \left(1 - \frac{\Delta}{\sigma}\right) \tag{10}$$

- The stay time of the stay point is NOT approximated to the stay time interval of the node: In this case, we define the difference Δ between the stay time of the stay point and the stay time interval of the node to be greater than the range of the stay time interval σ of the region. The *TemporalScore* is thus equal to zero.

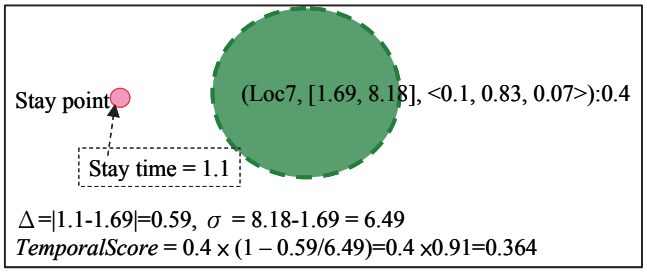

Fig. 15. Example of *TemporalScore*

For example, in Fig. 15, the user stays around the stay point for 1.1 time units. The range of the stay time interval σ of the region is 8.18 - 1.69 = 6.49 and the difference between the stay time of the stay point and the stay time interval of the node is |1.1 - 1.69| = 0.59 ≤ 6.49; hence, the *TemporalScore* is calculated using Equation (10). The *TemporalScore* is 0.4 × (1 – 0.59/6.49) = 0.364.

By traversing all paths in the GTS Pattern Tree of all users, the score of each path can be obtained as the summation of all *nodeScores* in the path. Each path obtained represents the most likely route of the user according to his/her own GTS Pattern Tree. Since a GTS Pattern Tree is built by reversing the order of patterns and inserting the reversed patterns into the tree as a path, the first node (i.e., the stump node of the tree) in each path can be used in the making of a prediction. Accordingly, we can use the average user similarity to evaluate the possibility score that a particular location will be the location that the user will move to. Given a user u and a location l, we define the possibility score of l as follows:

$$\text{Possibility}(u,l) = \frac{\sum_{v \in U(l)} similarity(u,v)}{|U(l)|} \tag{11}$$

where $U(l)$ is the set of users for which the first node in the path with the highest score is l. Accordingly, given a user-location pair, we can compute the possibility score of a location that the user will move to. Thus, we can make a next location prediction for current movement of a user by predicting the location as the one with the highest score.

6. RESULTS OF EXPERIMENTAL EVALUATION

In this section, we report on the results of a series of experiments conducted to evaluate the performance of our proposed next location prediction system using two real trajectory datasets crawled from two well-known trip-sharing websites—EveryTrail and Bikely. All the experiments were conducted using Java JDK 1.6 on an Intel Quad Core CPU Q6600 2.40 GHz computer with 1 GB of memory running Microsoft Windows XP. We first present the data preparation for the two crawled real datasets and then introduce the evaluation methodology. Finally, we present our results, followed by discussions.

6.1 Real Dataset

In this subsection, we detail the two real datasets used for examining our proposed method. Although there exist some widely-used trajectory datasets such as GeoLife Trajectory Dataset (http://research.microsoft.com/en-us/downloads/b16d359d-d164-469e-9fd4-daa38f2b2e13/default.aspx) or T-Drive Taxi Trajectroies (http://research.microsoft.com/apps/pubs/?id=152883), these datasets lack of semantic-related information. Consequently, these trajectory datasets do not fit for our experimental evaluation. Fortunately, some GPS sharing websites, like EveryTrail (http://www.everytrail.com/) and Bikely (http://www.bikely.com/), not only provide GPS sharing service for users but also allow users to tag some semantic terms on their uploaded trajectory. Accordingly, we crawled form EveryTrail and Bikely to make up two real datasets for our experimental evaluation.

6.1.1 EveryTrail Dataset. EveryTrail (http://www.everytrail.com/) is a trip-sharing and social networking website on which users can upload, share, and find trips. EveryTrail allows users to upload GPS logs and photos within a trip. Users can also

label an activity on a trip. While EveryTrail provides a public API that enables other applications to integrate with its service, some functionalities in the API are broken. For this reason, we used the API to support web page crawling in order to get all the data we needed. We consider the "stay" to be an important fact in location prediction. However, there are many trips on EveryTrail, such as biking, flying, and hiking, in which the activities involve no stay. The users that follow such trips do not "stay" anywhere. Therefore, we only selected trips comprising the kinds of activities from which we could obtain stay points. Using this rationale, we obtained 116,179 trips from 11 selected activities—specifically, Backpacking, Driving, Geocaching, Ice skating, Motorcycling, Relaxation, Romantic Getaway, Sightseeing, Snowshoeing, Walking, and Other. These trips were provided by 35,153 mobile users. For each mobile user, we randomly selected 70% of the trips from his/her trips dataset as our training dataset, and used the remaining trips to form our testing dataset.

6.1.2 Bikely Dataset. Bikely (http://www.bikely.com/) is also a trip-sharing website on which users can upload and share trip information. Bikely allows users to upload GPS logs within a trip and tag a trip using semantic terms. Bikely does not provide a public API for other applications to integrate with its service. For this reason, we had to crawl the sites web pages to get all the data we needed. As with EveryTrail, there are many trips on Bikely, such as training and offroad, in which no stay is involved. The users that follow such trips do not "stay" anywhere that has semantic meaning. Therefore, we selected only those trips with several kinds of tags for which we could obtain stay points. Using this rationale, we obtained 89,578 trips for 10 selected tags—specifically, Commute, Recreational, Onroad, Smooth, Not Bike, Low traffic, Urban, Rural, Scenic, and Touring. These trips were provided by 4,196 mobile users. For each mobile user, we randomly selected 70% of the trips from his/her trips dataset as our training dataset, and used the remaining trips to form our testing dataset.

6.2 Evaluation Methodology

The following are the main measurements used in our experimental evaluation. The Precision, Coverage, and F-measure are defined, respectively, by Equations (12), (13), and (14), where p^+ and p^- indicate the number of correct predictions and incorrect predictions, respectively, and $|R|$ indicates the total number of trajectories. In addition, we used the average improvement rate to measure the percentage by which our proposed method outperforms other methods. The average improvement rate is defined by Equation (15), in which m_{ours} and $m_{baseline}$ are the measured results of our proposed method and that of the compared baseline method, respectively.

$$\text{Precision} = \frac{p^+}{p^+ + p^-} \tag{12}$$

$$\text{Coverage} = \frac{p^+ + p^-}{|R|} \tag{13}$$

$$\text{F-measure} = \frac{2 \times \text{Precision} \times \text{Coverage}}{\text{Precision} + \text{Coverage}} \tag{14}$$

$$\text{Average improvement rate} = \frac{m_{ours} - m_{baseline}}{m_{baseline}} \tag{15}$$

The experiments were divided into two groups: i) internal evaluation; and ii) external comparison. The internal evaluation group focused on evaluation of various

parameter settings for our *GTS-LP* framework. Table VII summarizes the major parameters in our prediction model and their default values. We first evaluated the precision of our proposed GTS-based Location Prediction under various settings for parameters related to GTSP-Miner (i.e., α_s, α_t, and θ_m). We then compared the proposed *nodeScore* with various parameter settings (i.e., ω_1, ω_2, and ω_3) in terms of precision. For external comparison, we evaluated our proposed *GTS-LP* against the collaborative location recommendations model [Zheng *et al.*, 2010], spatial-social approximation [Backstrom *et al.*, 2010], and *WhereNext* [Monereale *et al.*, 2009] in terms of precision, coverage, and F-measure by varying parameters such as minimum support threshold and the training data proportion.

Table VII. Major parameters of GTS-based Location Prediction

Parameters	Description	Default value
θ_v	Semantic similarity threshold for GTS Pattern Tree construction	0.5
α_s	Significance level of the stay time interval	0.2
α_t	Significance level of the transition time interval	0.4
ω_1	Weight of *GeographicScore* in *nodeScore*	0.5
ω_2	Weight of *SemanticScore* in *nodeScore*	0.3
ω_3	Weight of *TemporalScore* in *nodeScore*	0.2
θ_m	Minimum support	0.005

6.3 Efficiency Evaluation of GTS-LP

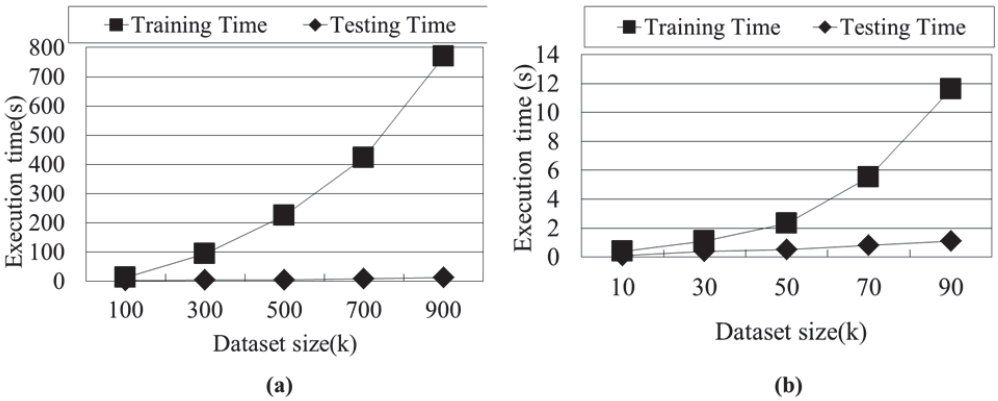

Fig. 16. Efficiency Evaluation of *GTS-LP*

In this experiment, we evaluated the efficiency of GTSP-Miner for various dataset sizes. We used a semi-simulation strategy to generate bigger datasets. First, we generated a synthetic user and randomly selected between two and 20 trips of real users as the synthetic user's trips. Fig. 16(a) shows the resulting dataset simulated from the EveryTrail dataset. As can be seen in Fig. 16(a), the training time increased exponentially as the size of the dataset increased. This is reasonable for training high quality models to make precise predictions. Fig. 16(a) shows that, unlike the training time, the testing time, i.e., the time for online prediction, linearly increased as the size of the dataset increased. This is also reasonable as more requests cost more in terms of time. In fact, in the real application, this step was panelized and sped up to constant time. Fig. 16 shows that the resulting dataset simulated from the Bikely dataset was similar to that simulated from the EveryTrail dataset. Since the Bikely dataset was smaller than the EveryTrail dataset, the dataset simulated from the

Bikely dataset was also small, as shown in Fig. 16(b). It can also be seen that the training time increased exponentially as the size of the dataset increased, while the testing time increased linearly as the size of the dataset increased.

6.4 Comparison of Various Parameters in GTSP-Miner

6.4.1 EveryTrail Dataset. In this experiment, we evaluated the precision of our approach under various parameter settings for GTSP-Miner based on the EveryTrail dataset. In the experiment, the default values of weights for *nodeScore* (ω_1, ω_2, and ω_3) were set at 0.5, 0.3, and 0.2, respectively. In Fig. 17(a), the highest precision occurs for parameter settings θ_v,= 0.5 and α_s = 0.3. However, we also observed that the resulting precision for parameter settings θ_v,= 0.5 and α_s = 0.3 were very close to the highest precision achieved. In Fig. 17(b), the highest precision occurs for parameter settings θ_v = 0.5 and α_t = 0.4. Both Fig. 17(a) and Fig. 17(b) show that the highest precision occurred when the semantic similarity threshold was set at 0.5. The optimal setting for the significance level of the stay time interval was smaller than that of the transition time interval. This signifies that the stay time interval should be more centralized than the transition time interval and also that they are always determined by different probabilistic distributions. This result strongly proves that movements of users are partially motivated by Geographic-triggered, Temporal-triggered, and Semantic-triggered Intentions.

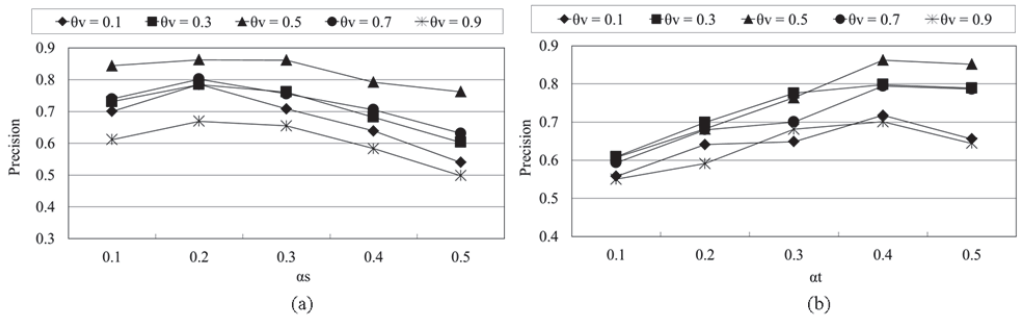

Fig. 17. Comparison of various parameter settings for GTSP-Miner based on the EveryTrail dataset

6.4.2 Bikely Dataset. In this experiment, we evaluated the precision of our approach under various parameter settings for GTSP-Miner based on the Bikely dataset. The default values for the weights of *nodeScore* (ω_1, ω_2, and ω_3) were set at 0.5, 0.3, and 0.2, respectively. In Fig. 18(a), the highest precision occurs for parameter settings θ_v,= 0.5 and α_s = 0.2. On the basis of this observation and the result shown in Fig. 17(a), the default values for θ_v and α_s were set at 0.5 and 0.2, respectively. In Fig. 18(b), the highest precision occurs for parameter settings θ_v= 0.5 and α_t = 0.4. Both Fig. 18(a) and Fig. 18(b) show that the highest precision occurred when the semantic similarity threshold was set at 0.5. This observation along with the result shown in Fig. 17(b) strongly supports our default settings. Similarly, as also shown by Fig. 17, the optimal significance level of the stay time interval was smaller than that of the transition time interval. This result strongly proves that it is necessary to consider the temporal property as the transition time and the stay time, respectively.

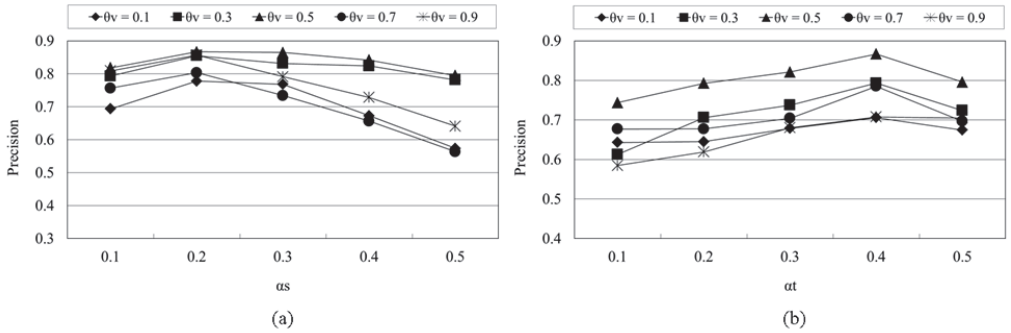

Fig. 18. Comparison of various parameter settings for GTSP-Miner based on the Bikely dataset

6.5 Comparison of Various Path Matching Scores

6.5.1 EveryTrail Dataset. In this experiment, we evaluated our approach under various parameter settings in terms of Precision based on the EveryTrail dataset. As can be seen in Fig. 19(a), the Precision of our method improved when ω_1 was increased, i.e., higher precision was achieved when we gave more weight to geographic behaviors, with the highest precision occurring for parameter settings ω_1 = 0.5 and ω_2 = 0.3 (i.e., ω_3 = 0.2). Even though it appears that the Geographic-triggered behaviors are the most essential property, the semantic- and temporal-triggered behaviors still potentially have an effect on predicting the next movement. We also observed that the Precision does not increase monotonically as ω_2 increases, i.e., as more weight is assigned to *SemanticScore*, the precision does not necessarily increase. This contradicts our assumption that considering semantic-triggered behaviors will improve the precision of prediction. We believe that this is because the temporal-triggered behaviors still potentially have an effect on predicting the next movement. On the basis of this observation, we set the default values for ω_1, ω_2, and ω_3 at 0.5, 0.3, and 0.2, respectively. As shown in Fig. 19(b), the Precision of our method improved when the minimum support increased, i.e., higher precision was achieved when we focused on more frequent behaviors, and the highest precision occurred for the following parameter settings: minimum support = 3%, ω_1, = 0.5, ω_2 = 0.3, and ω_3 = 0.2 (i.e., when all the Geographic-triggered, temporal-triggered, and semantic-triggered behaviors were taken into consideration). We also observed that the Precision was worst, on average, when we set ω_1 = 1 (i.e., when only the geographic property was taken into consideration).

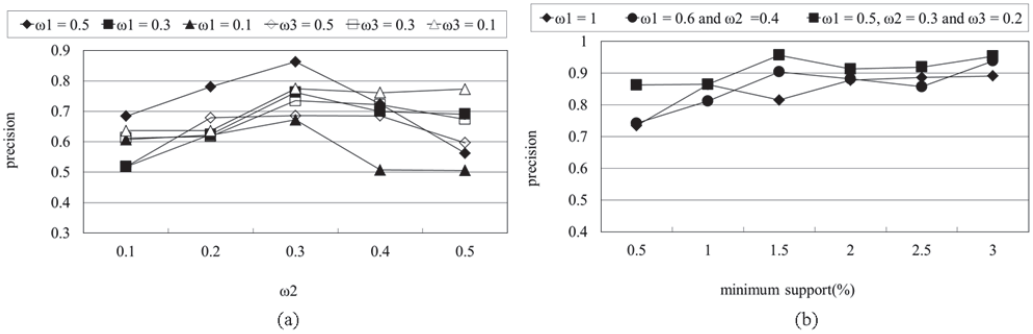

Fig. 19. Comparison of various parameter settings based on the EveryTrail dataset

6.5.2 Bikely Dataset. In this experiment, we evaluated our approach under various parameter settings in terms of Precision based on the Bikely dataset. Fig. 20(a) shows that a similar result to that of Fig. 19(a) was obtained—the Precision of our method improved when ω_1 was increased, i.e., higher precision was achieved when more weight was given to geographic behaviors, with the highest precision occurring for parameter settings ω_1 = 0.5 and ω_2 = 0.3 (i.e., ω_3 = 0.2). This result strongly supports the default value of ω_1, ω_2, and ω_3 being set at 0.5, 0.3, and 0.2, respectively. However, from Fig. 20(b), it can be seen that the Precision of our method decreased when the minimum support increased, i.e., higher precision was achieved when we mined more patterns, and the highest precision occurred for the following parameter settings: minimum support = 0.5%, ω_1, = 0.5, ω_2 = 0.3, and ω_3= 0.2 (i.e., when all the Geographic-triggered, temporal-triggered, and semantic-triggered behaviors were taken into consideration). This is because the average number of trips per user in the Bikely dataset is larger than that of the EveryTrail dataset. When minimum support is decreased, many significant patterns can still be mined from the Bikely dataset. We also observed that the Precision was worst, on average, for ω_1 = 1 (i.e., when only the geographic property was taken into consideration).

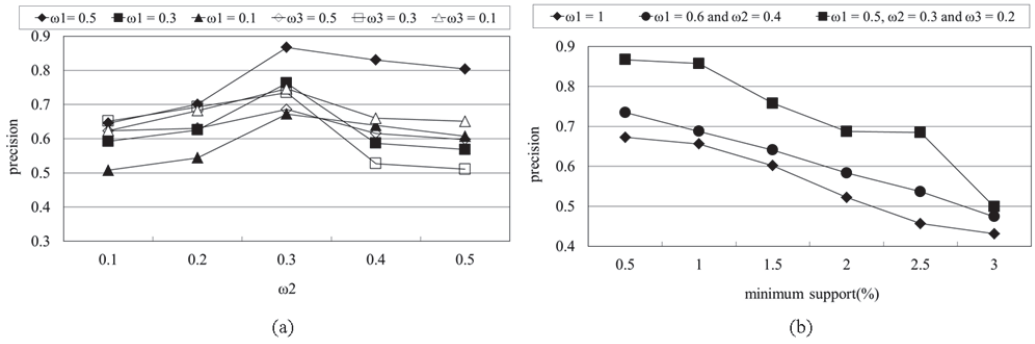

Fig. 20. Comparison of various parameter settings based on the Bikely dataset

6.6 Impact of the Minimum Support Threshold

6.6.1 EveryTrail Dataset. In this experiment, we analyzed the precision, coverage, and F-measure of the prediction techniques—specifically, the collaborative location recommendations model [Zheng *et al.*, 2010], spatial-social approximation [Backstrom *et al.*, 2010], *WhereNext* [Monereale *et al.*, 2009], and our *GTS-LP* approach—for various minimum support thresholds. Since collaborative location recommendation and spatial-social approximation are not pattern based prediction models, they do not need the minimum support parameter to mine frequent behavior of users. Therefore, the two models give a constant result under various minimum support settings. Fig. 21 shows that *GTS-LP* performed marginally better than collaborative location recommendation in terms of precision, coverage, and F-measure, but significantly outperformed *WhereNext* and spatial-social approximation for those measures. This is because *GTS-LP* and collaborative location recommendation both utilize a user-based collaborative filtering framework for their prediction model. The user-based collaborative filtering framework makes the prediction more precise and be able to deal with trajectories that cannot be predicted. The difference between our method and collaborative location recommendation is that our method considers all the Geographic-triggered, Temporal-triggered, and

Semantic-triggered behaviors. Therefore, our method is more precise. For these two reasons, the precision, coverage, and F-measure improved. The average improvement rates of *GTS-LP* over *WhereNext* were 63.17% for precision, 555.03% for coverage, and 383.62% for F-measure. The average improvement rates of *GTS-LP* over collaborative location recommendation were 29.40% for precision, 0% for coverage, and 7% for F-measure. The average improvement rates of *GTS-LP* over spatial-social approximation were 3% for precision, 366.88% for coverage, and 198.68% for F-measure.

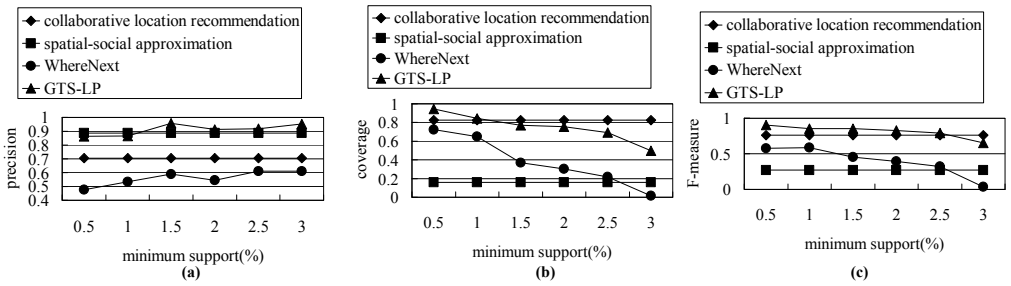

Fig. 21. Impact of the Minimum Support Threshold based on the EveryTrail dataset

6.6.2 Bikely Dataset. In this experiment, we analyzed the precision, coverage, and F-measure of the prediction techniques examined—specifically, the collaborative location recommendations model [Zheng *et al.*, 2010], *WhereNext* [Monereale *et al.*, 2009], and our *GTS-LP* approach—for various minimum support thresholds. Since Bikely (http://www.bikely.com/) does not provide a social networking service, spatial-social approximation cannot work on its dataset. In addition, because collaborative location recommendation is not a pattern based prediction model, it does not need the minimum support parameter to mine frequent behaviors of users. Therefore, collaborative location recommendation gives a constant result under various minimum support settings. Fig. 22 shows that *GTS-LP* was marginally better than collaborative location recommendation in terms of precision, coverage, and F-measure, but significantly outperformed *WhereNext* for those measures.

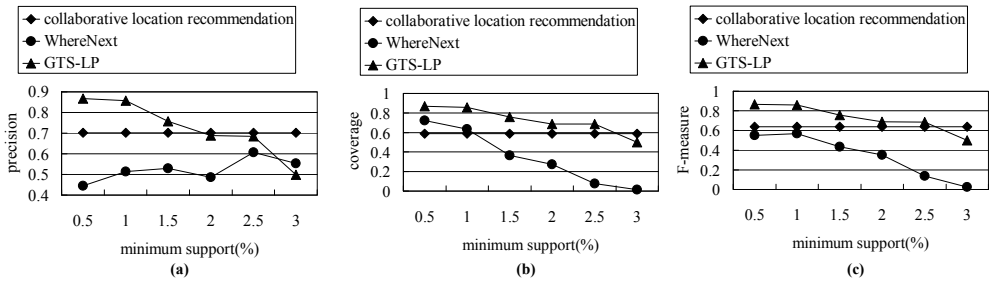

Fig. 22. Impact of the Minimum Support Threshold based on the Bikely dataset

As in the previous experiment, this occurred because *GTS-LP* and collaborative location recommendation both utilize a user-based collaborative filtering framework in their prediction model. The user-based collaborative filtering framework makes the prediction more precise and able to deal with trajectories that cannot be predicted. The difference between our method and collaborative location recommendation is that our method takes into consideration all the Geographic-triggered, temporal-triggered, and semantic-triggered behaviors in the prediction of

next location of a user. This results in our method being able to comprehensively deal with all types of movement of a user. Thus, the precision, coverage, and F-measure are improved. The average improvement rates of *GTS-LP* over *WhereNext* were 41.48% for precision, 788.90% for coverage, and 415.19% for F-measure. The average improvement rates of *GTS-LP* over collaborative location recommendation were 3.3% for precision, 23.58% for coverage, and 13.44% for F-measure.

6.7 Impact of the Training Dataset Size

6.7.1 EveryTrail Dataset. In this experiment, we analyzed the precision, coverage, and F-measure of the prediction techniques examined—that is, the collaborative location recommendations model [Zheng *et al.*, 2010], spatial-social approximation [Backstrom *et al.*, 2010], *WhereNext* [Monereale *et al.*, 2009], and our *GTS-LP* approach for various training data proportions. The training data proportion refers to the proportion of the dataset we used to train the prediction model. For example, when we set the training proportion at 70%, 70% of the trajectories were treated as training data, and the remaining 30% were treated as testing data. Fig. 23 shows that the precision of *GTS-LP* was sensitive to training proportion. This is because *GTS-LP* takes into consideration all the Geographic-triggered, temporal-triggered, and semantic-triggered behaviors in its computation of *nodeScore*. When the training data is deficient, it not only damages the geographic property, but also lacks semantic and temporal properties. We also observed that *GTS-LP* marginally outperformed collaborative location recommendation and spatial-social approximation in terms of coverage and F-measure, but significantly outperformed *WhereNext* for those measures. On average, the improvement rate was still reasonable. The average improvement rates for *GTS-LP* over *WhereNext* were 60.17% for precision, 19.78% for coverage, and 40.78% for F-measure. The average improvement rates for *GTS-LP* over collaborative location recommendation were 12.67% for precision, 4.97% for coverage, and 8.9% for F-measure. The average improvement rates for *GTS-LP* over spatial-social approximation were 47.34% for precision, 544.22% for coverage, and 284.24% for F-measure.

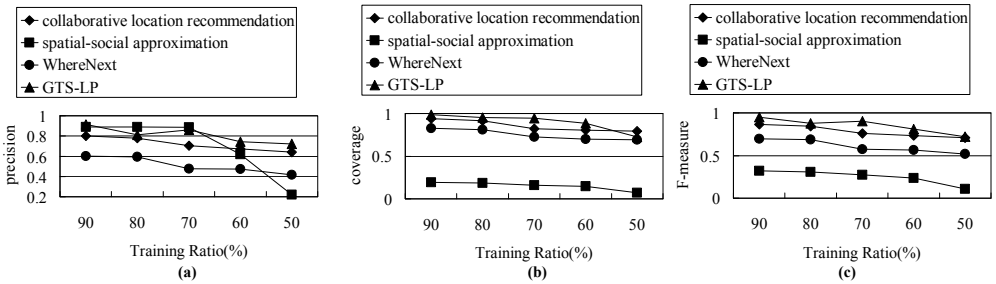

Fig. 23. Impact of size of Training Dataset based on the EveryTrail dataset

6.7.2 Bikely Dataset. In this experiment, we analyzed the precision, coverage, and F-measure of the prediction techniques examined—that is, the collaborative location recommendations model [Zheng *et al.*, 2010], *WhereNext* [Monereale *et al.*, 2009], and our *GTS-LP* approach—for various training data proportions. Since Bikely (http://www.bikely.com/) does not provide a social networking service, spatial-social approximation could work on its dataset. Fig. 24 shows that the precision of *GTS-LP* was not sensitive to training proportion. This result contradicts the observation made

in the case of the EveryTrail dataset. This is because the average number of trips per user in the Bikely dataset is greater than that of the EveryTrail dataset. When the size of the training dataset decreases, there are still many significant patterns to be mined from the Bikely dataset. It can also be seen that *GTS-LP* marginally outperformed collaborative location recommendation and *WhereNext* in terms of precision, coverage, and F-measure. On average, the improvement rate was quite reasonable. The average improvement rates for *GTS-LP* over *WhereNext* were 53.58% for precision, 26.46% for coverage, and 39.31% for F-measure. The average improvement rates for *GTS-LP* over collaborative location recommendation were 24.5% for precision, 82.65% for coverage, and 48.4% for F-measure.

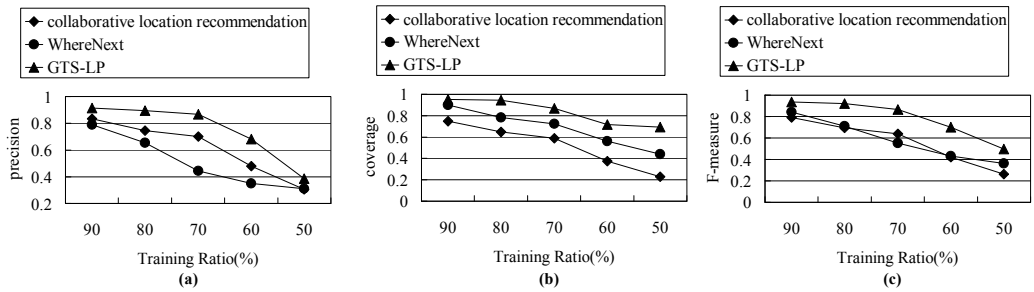

Fig. 24. Impact of size of Training Dataset based on the Bikely dataset

7. CONCLUSIONS

In this paper, we defined a new kind of frequent pattern, namely GTS Pattern, which takes into account moving behaviors of users motivated by *Geographic-triggered*, *Temporal-triggered,* and *Semantic-triggered Intentions*. On the basis of GTS Pattern, we proposed a novel user-based collaborative filtering framework called *GTS-LP* to predict the next location of a mobile user for applications such as location-based services. The core of our prediction module is a novel prediction strategy that evaluates the score of the next stay location for a given mobile user by mining the moving behaviors of users in terms of the geographic, temporal, and semantic properties. To the best of our knowledge, this is the first work that focuses on next location prediction by mining trajectory data that takes into consideration all the geographic, temporal, and semantic moving behaviors of users. Through a series of experiments, we validated our proposal and showed that our approach gives excellent performance under various conditions and also outperforms state-of-the-art approaches such as collaborative location recommendation [Zheng *et al.*, 2010], spatial-social approximation [Backstrom *et al.*, 2010], and *WhereNext* [Monereale *et al.*, 2009] in terms of precision, coverage, and F-measure. For future work, we will explore to design more advanced methods to further enhance the quality of location prediction systems for various location-based service applications. Besides, since our prediction model considers all of the geographic, temporal and semantic properties of users' moving behaviors, a number of parameters corresponding to these properties are used in the model. As the next step, we will also try to simplify the parameters to reduce the complexity of the model.

REFERENCES

Alvares, L.O., Bogorny, V., Palma, A., Kuijpers, B., Moelans, B., AND Macedo., J.A.F. 2007. Towards semantic trajectory knowledge discovery. *Technical Report*, Hasselt University, Belgium.

Bogorny, V., Kuijpers, B., AND Alvares, L.O. 2009. ST-DMQL: A semantic trajectory data mining query language. *International Journal of Geographical Information Science* 23(10), 1245-1276.

BACKSTROM, L., SUN, E., AND MARLOW, C. 2010. Find me if you can: Improving geographical prediction with social and spatial proximity. In *Proceedings of the 19th International Conference on World Wide Web*, Raleigh, North Carolina, USA, April 26-30, 2010.

EAGLE, N., AND PENTLAND, A. 2009. Eigenbehaviors: Identifying structure in routine. *Behavioral Ecology and Sociobiology*, 63(7),1057–1066.

GE, Y., LIU, Q., XIONG, H., TUZHILIN, A. 2011. Cost-aware Travel Tour Recommendation, *In proceeding of KDD,* 2011

GE, Y., XIONG, H., TUZHILIN, A., XIAO, K., GRUTESER, M., PAZZANI, M. J. 2010. An Energy-Efficient Mobile Recommender System. *In proceeding of KDD,* 2010

GIANNOTTI, F., NANNI, M., AND PEDRESCHI, D. 2006. Efficient mining of temporally annotated sequences. In *Proceedings of the 6th SIAM International Conference on Data Mining*, Bethesda, MD, April, 2006.

GONZALEZ, M.C., HIDALGO, C.A., AND BARABASI, A.-L. 2008. Understanding individual human mobility patterns. *Nature*, 453(7196),779–782.

JEUNG, H., LIU, Q., SHEN, H.T., AND ZHOU, X. 2008. A hybrid prediction model for moving objects. In *Proceedings of the 22nd IEEE International Conference on Data Engineering*, 70-79.

JIANG, B., YIN, J., AND ZHAO, S. 2009. Characterizing the human mobility pattern in a large street network. *Physical Review E*, 80(2):021136,.

JOSHI, D., SAMAL, A.K., AND SOH., L.K. 2009. Density-based clustering of polygons. In *Proceedings of the IEEE Symposium Series on Computational Intelligence and Data Mining*, 2009, 171-178.

LI, Z., HAN, J., JI, M., TANG, L.-A., YU, Y., DING, B., AND LEE, J.-G. 2011. Roland Kays: MoveMine: Mining moving object data for discovery of animal movement patterns. *ACM TIST,* 2(4), 37.

LIU, Q., GE, Y., LI, Z., CHEN, E., XIONG, H. 2011. Personalized Travel Package Recommendation. *In proceeding of ICDM,* 2011.

MORZY, M. 2006. Prediction of moving object location based on frequent trajectories. *Springer ISCIS,* 4263,583–592.

MORZY, M. 2007. Mining frequent trajectories of moving objects for location prediction. *MLDM 4571*, 667–680.

MONREALE, A., PINELLI, F., TRASARTI, R., AND GIANNOTTI. F. 2009. WhereNext: A location predictor on trajectory pattern mining. *In Proceedings of KDD,* 2009, 637-646.

NOULAS, A., SCELLATO, S., MASCOLO, C., AND PONTIL, M. 2011. An empirical study of geographic user activity patterns in foursquare. In *Proceedings of the International Conference on Weblogs and Social Media (ICWSM '11)*, 2011.

PAPOULIS, A., AND PILLAI, S.U. 2002. *Probability, Random Variables and Stochastic Processes* . McGraw-Hill, New York, NY.

PARK, M., HONG, J., AND CHO, S., 2007. Location-based recommendation system using Bayesian users preference model in mobile devices. *Ubiquitous Intelligence and Computing*, 2007.

PEI, J., HAN, J., MORTAZAVI-ASL, B., PINTO, H., CHEN, Q., DAYAL, U., AND HSU., M.C. 2001. PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *Proceedings of the 17th International Conference on Data Engineering* , 2001, 215-224.

ROSS, S.M. 2004. *Introduction to Probability and Statistics for Engineers and Scientists*. Wiley, New York, NY, 2004.

SONG, C., QU, Z., BLUMM, N., AND BARABASI, A. 2010. Limits of predictability in human mobility. *Science*.

WEI, L.-Y., ZHENG, Y., AND PENG, W.-C. 2012. Constructing popular routes from uncertain trajectories, In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, August 12-16, 2012, Beijing, China

XUE, A.Y., ZHANG, R., ZHENG, Y., XIE, X., HUANG, AND XU, Z. 2013. Destination Prediction by Sub-Trajectory Synthesis and Privacy Protection Against Such Prediction. In *Proceedings of IEEE ICDE*, 2013.

YAVAS, G., KATSAROS, D., ULUSOY, Ö., AND MANOLOPOULOS., Y. 2005. A data mining approach for location prediction in mobile environments. *D.K.E., 54(2)*, 121–146.

YE, Y., ZHENG, Y., CHEN, Y., FENG, J., AND XIE, X. 2009. Mining individual life pattern based on location history. *In Proceedings of the International Conference on Mobile Data Management (MDM 2009),* 2009.

YING, J.J.C., LU, E.H.C., LEE, W.C., WENG, T.C., AND TSENG, V.S. 2010. Mining user similarity from semantic trajectories. In *Proceedings of ACM SIGSPATIAL International Workshop on Location based Social Networks*, San Jose, California, USA, November 2010.

YING, J.J.C., LEE, W.C., WENG, T C., AND TSENG, V.S. 2011. Semantic trajectory mining for location prediction. In *Proceedings of The 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM GIS' 11)*, Chicago, IL, November 2011.

ZHENG, V.W., CAO, B., ZHENG, Y., XIE, X., AND YANG, Q. 2010. Collaborative filtering meets mobile recommendation: A user-centered approach. In *Proceedings of AAAI 2010*.

ZHENG, Y., WANG, L., ZHANG, R., XIE, X., AND MA., W.Y., 2009. GeoLife: Managing and understanding your past life over maps. In *Proceedings of IEEE MDM*, 2009,

ZHENG, Y., ZHANG, L., XIE., X., AND MA, W.Y. 2009. Mining interesting locations and travel sequences from GPS trajectories. In *Proceedings of ACM WWW*, 2009.

ZHENG, Y., ZHANG, L., XIE., X., AND MA, W.Y. 2009. Correlation between locations using human location history. *In Proceedings of The 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, , Seattle, Washington, USA, November, 2009.

ZHENG, Y., ZHANG, L., XIE., X., AND MA, W.Y. 2010. Finding similar users using category-based location histories. *In Proceedings of The 18th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems* , San Jose, California, USA, November, 2010.

ZHENG, Y., ZHANG, L., AND XIE., X. 2011. Recommending friends and locations based on individual location history. *ACM Trans. on the Web 5(1),* Article 5, pages 44, 2011

ZHENG, V.W., ZHENG, Y., XIE., X., AND YANG, Q. 2010. Collaborative location and activity recommendations with GPS history data. In *Proceedings of the 19th International Conference on World Wide Web*, April 26-30, 2010, Raleigh, North Carolina, USA.

ZHENG, Y., AND ZHOU, X. 2011. Computing with Spatial Trajectories, *Springer Publishing Company*, Incorporated, 2011

ZHUANG, J., MEI, T., HOI, S., XU, Y., AND LI, S. 2011 When recommendation meets mobile: Contextual and personalized recommendation on the go. In *Proceedings of UbiComp 2011*.