الجمهورية الجزائرية الديمقراطية الشعبية
LA REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET PUBLIQUE

وزارة التعليم العالي و البحث العلمي
MINISTERE DE L'EDUCATION SUPERIEURE ET DE LA RECHERCHE SCIENTIFIQUE
جامعة عمّار ثليجي بالأغواط
UNIVERSITE AMAR TELIDJI-LAGHOUAT

كلية العلوم
FACULTE DES SCIENCES
DEPARTEMENT D'INFORMATIQUE

L'UTILISATION DES TECHNIQUES D'APPRENTISSAGES AUTOMATIQUE POUR LA PRÉDICTION DE MOBILITÉ DANS LES RÉSEAUX VÉHICULAIRES

Thèse présentée pour obtenir le grade de Docteur en Informatique

Par

AMIRAT HANANE

Membres de jury

M.YAGOUBI	Professeur	Université de Laghouat	Président
A.MOUSSAOUI	Professeur	Université de Sétif 1	Examinateur
A.KORICHI	Professeur	Université de Ouargla	Examinateur
M.BOUAKKAZ	MCA	Université de Laghouat	Examinateur
N.LAGRAA	Professeur	Université de Laghouat	Encadreur
Y.OUINTEN	Professeur	Université de Laghouat	Co-Encadreur

2019/2020

الجمهورية الجزائرية الديمقراطية الشعبية
PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
وزارة التعليم العالي و البحث العلمي
MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH
جامعة عمّار ثليجي بالأغواط
UNIVERSITY AMAR TELIDJI OF LAGHOUAT

كلية العلوم
FACULTY OF SCIENCES
DEPARTMENT OF COMPUTER SCIENCE

THE USE OF MACHINE LEARNING TECHNIQUES FOR MOBILITY PREDICTION IN VEHICULAR NETWORKS

Thesis submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Science

Thesis defended on

AMIRAT HANANE

Jury members

M.YAGOUBI	Professor	University of Laghouat	President
A.MOUSSAOUI	Professor	University of Setif 1	Examiner
A.KORICHI	Professor	University of Ouargla	Examiner
M.BOUAKKAZ	MCA	University of Laghouat	Examiner
N.LAGRAA	Professor	University of Laghouat	Supervisor
Y.OUINTEN	Professor	University of Laghouat	Co-Supervisor

2019/2020

Abstract

The rapid development of location acquisition and mobile communication technologies have fostered a number of location-based services, such as location prediction and recommendation. While location prediction consists of forecasting the future location of a user or a moving object (e.g. vehicle), location recommendation aims at suggesting the next venues that a user may be interested to visit in the future. These services have several important applications in traffic congestion forecasting, location-based routing protocol designs, and targeting advertisements generation.

In literature, various techniques and models have been proposed for effective location prediction and recommendation, e.g., Markov model, collaborative filtering. Although these models were shown to perform well, they suffer from several limitations and drawbacks. Among these we can mention information loss in prediction, noise sensitivity toward mobility data and static modeling in the recommendation. Besides, and due to the specificity of human mobility, the nature of mobility data and the continuous evolution of these services, the application of these methods in predicting and recommending locations become in many cases irrelevant and obsolete.

To deal with these issues, we propose, in this thesis, three models for location prediction and recommendation. The first proposed model is named *NextRoute*. This predictor is a novel and accurate lossless model as it compresses location data in a prediction tree without information loss, and it is designed to use all the relevant information contained in the training data to perform prediction.

The second proposed model is *MyRoute*, a dependency-graph based predictor for real-time route prediction. *MyRoute* represents routes as a graph, which is then used to accurately match road network architecture with real-world vehicle movements. Unlike many prediction models, the designed model is noise-tolerant, and can thus provide high accuracy even with data that contains noise and inaccuracies such as GPS mobility data.

The third model we propose, in this thesis, is a rule based point of interest recommendation system we named *STS-Rec*. In addition to the sequential behavior of human mobility, this model takes also into account social and temporal influences. *STS-Rec* first transforms mobility data into location sequences. Then, it mines sequential recommendation rules from these sequences. The experimental evaluations we conducted on large-scale and realistic datasets show that our proposed models outperform several state-of-the-art models in terms of accuracy and coverage.

Keywords: *real-time, route prediction, dependency graph, Compact Prediction Tree, noise tolerance, lossless model, POI recommendation, Sequential rules mining.*

Résumé

Le développement rapide des techniques d'acquisition de localisation et de communication mobile a favorisé l'apparition d'un grand nombre de services basés sur la localisation, tels que la prédiction et la recommandation de localisation. Alors que la prédiction de mobilité consiste à prédire la localisation future d'un utilisateur ou d'un objet mobile (ex. Véhicule), la recommandation de localisation vise à suggérer les prochains sites qu'un utilisateur pourrait être intéressé à visiter dans l'avenir. Ces deux services ont plusieurs applications importantes telles que la prédiction de la congestion du trafic, la conception de protocoles de routage basés sur la localisation et le ciblage de la génération de publicité.

Dans la littérature, diverses techniques et modèles ont été proposés pour supporter ces services notamment les modèles de Markov et le filtrage collaboratif. Bien que ces modèles se soient révélés performants, ils souffrent de plusieurs limitations et inconvénients tels que la perte d'informations lors de la prédiction, la sensibilité au bruit dans les données de mobilité et la modélisation statique des systèmes de recommandation. Par ailleurs, et en raison de la spécificité de la mobilité humaine, la nature des données de mobilité ainsi que l'évolution continue de ces services, l'application des méthodes traditionnelles est devenue moins pertinente. Par conséquent, le développement de nouveaux modèles efficaces, remédiant à ces problèmes, s'avère nécessaire et indispensable.

A cet effet, nous proposons, dans cette thèse, trois modèles de prédiction et de recommandation de localisation. Initialement, un nouveau modèle de prédiction de route nommé *NextRoute* est proposé. Ce dernier est un modèle efficace sans perte car il compresse les données de localisation dans un arbre de prédiction sans perte d'informations. *NextRoute* est principalement conçu pour utiliser toutes les informations pertinentes contenues dans les données d'apprentissage pour effectuer la prédiction.

MyRoute est le deuxième modèle proposé dans cette thèse. *MyRoute* est un prédicteur basé sur les graphes de dépendances pour la prédiction temps réel des routes. Il représente les routes sous forme d'un graphe, qui est ensuite utilisé pour faire la correspondre entre l'architecture du réseau routier et les mouvements réels des véhicules. Contrairement à de nombreux modèles de prédiction, le modèle conçu est tolérant au bruit et il peut donc fournir une grande précision même avec des données contenant du bruit et des inexactitudes telles que les données de mobilité GPS.

Le troisième modèle proposé, dans cette thèse, est *STS-Rec*. Ce modèle est un système de recommandation de point d'intérêt basé sur la fouille des règles séquentielles. *STS-Rec* prend en compte non seulement le comportement séquentiel de la mobilité humaine, mais aussi les influences sociales et temporelles. *STS-Rec* transforme d'abord les données de mobilité en séquences de localisation qu'il utilise ensuite pour extraire les règles de recommandation séquentielles.

L'évaluation expérimentale que nous avons réalisée sur des ensembles de données réalistes et a grande échelle a montré que les modèles que nous

avons proposé réalisent une meilleur performance en terme de précision et de couverture, comparés aux modèles existants.

Mots-clés: *temps réel, prédiction de route, graphe de dépendance, arbre de prédiction compact, tolérance au bruit, modèle sans perte, recommandation des points d'intérêt, la fouille des règles sequentielles.*

الملخص

أدى التطور السريع في تقنيات اكتشاف الموقع وتقنيات الاتصالات المتنقلة إلى ظهور عدد كبير من الخدمات القائمة على الموقع ، مثل التنبؤ بالموقع و اقتراح المواقع. التنبؤ بالتنقل او الموقع يهدف اساسا للتنبؤ بالموقع المستقبلي لمستخدم أو كائن متنقل (مثل السيارة)، في حين تهدف توصية المواقع إلى اقتراح الاماكن التي قد يكون المستخدم مهتمًا بزيارتها في المستقبل.
تدعم هاتان الخدمتان العديد من التطبيقات المهمة، مثل توقع ازدحام المرور، وتصميم بروتوكولات التوجيه القائمة على الموقع واستهداف الإعلانات.
وفقا للدراسات السابقة، فقد تم اقتراح عدة تقنيات ونماذج مختلفة لدعم هذه الخدمات، لا سيما نماذج ماركوف، والتصفية التعاونية. على الرغم من أن هذه النماذج أثبتت كفاءتها، إلا أنها تعاني من العديد من العيوب مثل عدم اخذ جميع المعلومات في الحسبان أثناء التنبؤ، والحساسية للضوضاء في بيانات التنقل، والنمذجة الثابتة لأنظمة التوصية.

علاوة على ذلك، ونظراً لخصوصية تنقل البشر، وطبيعة بيانات التنقل وكذلك التطور المستمر لهذه الخدمات، أصبح تطبيق الأساليب التقليدية في تصميم انظمة التنبؤ والتوصية اقل فعالية ، ومن هنا فان اقتراح نماذج جديدة و فعالة، كعلاج لهذه المشاكل اضحى حاجة ملحة وجب اخذها بعين الاعتبار.
تحقيقا لهذه الغاية ،وبناءا على ما سبق ذكره، فإننا نقترح في هذه الاطروحة عدة انظمة جديدة للتنبؤ و توصية المواقع.

أول نظام نقترحه هو نظام NextRoute و الذي صمم أساسا لتنبؤ الطريق القادمة التي ستسلكها السيارة. هذا الأخير هو نموذج فعال يأخذ كل معلومات التنقل بعين الاعتبار لأنه يضغط بيانات المواقع في شجرة للتنبؤ دون فقدان للمعلومات. تم تصميم NextRoute في المقام الأول لاستخدام جميع المعلومات ذات الصلة الواردة في بيانات التدريب لجعل التنبؤ أكثر فعالية و دقة.

MyRoute هو النموذج الثاني المقترح في هذه الاطروحة. MyRoute عبارة عن متنبئ يعتمد على الرسوم البيانية التبعية للتنبؤ بالمسار آنيا وفي االوقت الفعلي والحقيقي للحركة. MyRoute يمثل الطرق في شكل رسم بياني ، والذي يستخدم بعد ذلك لمطابقة بنية شبكة الطرق وكذلك الحركات الفعلية للمركبات. على عكس العديد من نماذج التنبؤ، هذا النموذج مصمم خصيصا لمقاومة التشويش الموجود في البيانات ، وبالتالي يمكنه توفير دقة عالية حتى مع وجود بيانات غير دقيقة و تحتوي على ضوضاء مثل

بيانات GPS .

النموذج الثالث المقترح في هذه الاطروحة هو STS-Rec . هذا النموذج هو نظام توصية للمواقع و اماكن الاهتمام للمستخدمين و الذي يرتكز أساسا على استخراج البيانات من السلاسل. STS-Rec لا يأخذ في الاعتبار السلوك التسلسلي للتنقل البشري فحسب ، ولكن أيضا التأثيرات الاجتماعية والزمنية. يقوم STS-Rec أولاً بتحويل بيانات التنقل إلى سلاسل مكانية و التي تستعمل بعد ذلك لاستخراج قواعد التوصية.

بالنسبة لجميع النماذج المقترحة في هذا البحث، أظهر التقييم التجريبي، الذي أجري باستعمال مجموعات من البيانات الحقيقية، فعالية مقترحاتنا التي تجاوزت العديد من النماذج الحالية من حيث الدقة والقدرة على التنبؤ.

الكلمات المفتاحية: الوقت الحقيقي، التنبؤ بالمسار، الرسم البياني التبعي، شجرة التنبؤ المدمجة، التسامح مع الضوضاء، نموذج فاقد للمعلومات، توصية المواقع ، استخراج البيانات التسلسلية.

Acknowledgements

First of all, I would like to thank ALLAH for helping me through this journey of life. I have felt his guidance in every stages of this thesis. I would like to express my special thanks to my supervisors, Prof. Lagraa Nasreddine and Prof. Ouinten Youcef, for their generous support, and their valuable and in-depth guidance for my PhD study and research. In the past six years, I learned a lot from them. With their help, I learned how to discover fresh and intriguing research topics, how to peer-review publications, and how to write high-quality papers. The six-year research experience will also be of great benefit in my future career. I would also like to thank, Prof. Philippe Fournier-Viger. I appreciate all his contributions of time, support and ideas, to make my PhD. experience stimulating and rewarding. I would like also to thank the examining committee Pr.Yagoubi,Pr.Moussaoui, Pr.korichi and Dr.Bouakkaz for their time, kindness and supportive comments that have raised a number of interesting points of discussion.

"Dedicated to my beloved parents, family & Friends"
For their love, endless support, encouragement & sacrifices

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Research Background

In recent years, mobile devices have become essential tools in human life and work. Based on their embedded sensors, the advances in mobile technologies, and the availability of online social applications a large number of appealing location-based services (Figure 1.1) such as mobility prediction, location recommendation, trip planning have emerged providing crucial intelligence to business and governments.

More specifically, mobility prediction and location recommendation are key services that are used in several applications such as Foursquare [1], Google Places [2], Yelp [3], etc.

While mobility prediction concerns human mobility as well as other moving objects (vehicles, animals, etc.), location recommendation is mainly designed for location-based social networks (LBSN) that exploit the user location to perform recommendation by suggesting locations or POI (point-of-interest) that a user may find useful or interesting to visit.

In the research area, most recent studies for location prediction and recommendation assume that people's movements is highly regular and socially dependent as human mobility is influenced by others. Thus, human mobility could be predicted with a certain precision. In literature, several approaches have been proposed. Particularly, machine learning based models such as probabilistic models (Markov chains model and its variations, Bayesian networks), artificial intelligence (neural networks), data mining (sequential patterns mining). are the prominent methods that have been broadly applied. These models constitute powerful tools dealing with location prediction and recommendation tasks like or even better than a human does as they encompass intelligence and enable discovering hidden correlations among data.

In this thesis, we mainly study the problems of mobility prediction of vehicles for vehicular networks and location recommendation in location-based social networks. The core of our research is to consider the sequential nature of human displacement to develop efficient mobility predictors and recommenders using machine learning techniques (probabilistic models and sequential data mining).

For ease of presentation and reader convenience, we will introduce the studied issues related to route prediction and POI recommendation separately.

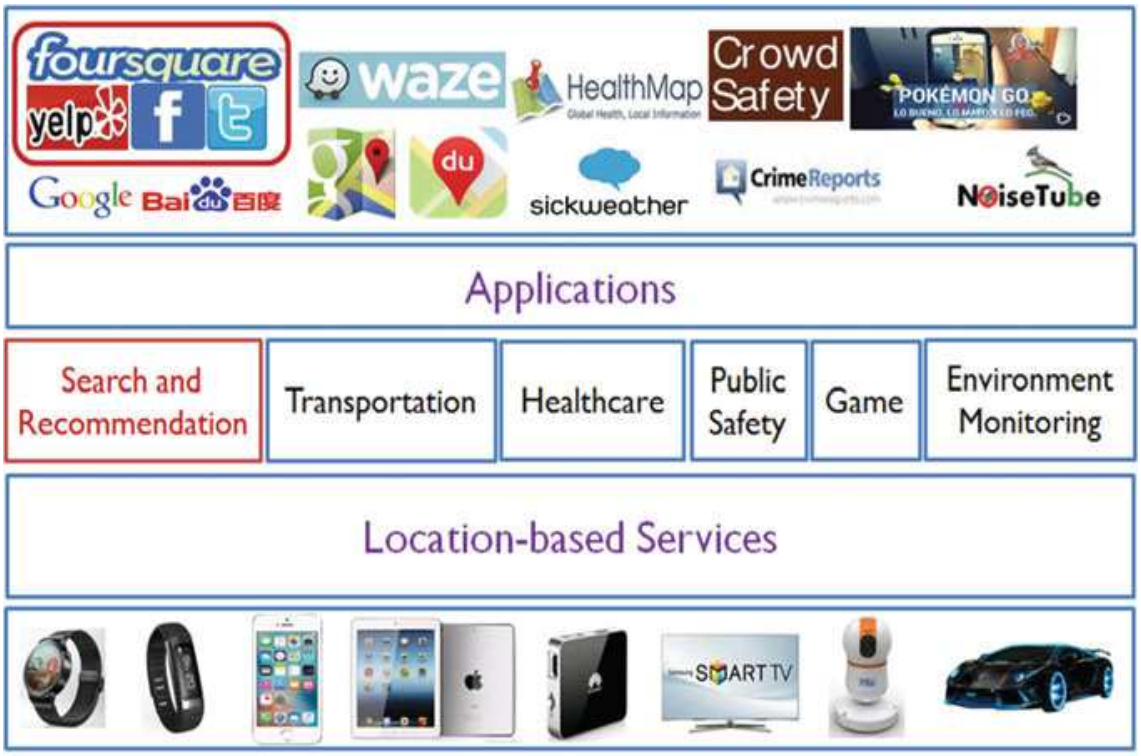


FIGURE 1.1: A sample of Location Based Services [4].

For each problem, a brief description, the problem formulation and the challenges to be faced are provided. Then follows the presentation of the major contributions we made to fulfill the challenges. In the end, the organization of the thesis is given.

1.2 Route prediction

With the increasing number of vehicles traveling in cities, mobility prediction of vehicles (aka route prediction) has become a key feature toward efficient mobility management in smart cities, Intelligent Transportation Systems (ITS), and mobile computing environments. For instance, route prediction can be used to display traffic warnings (e.g. road incidents), and to provide more accurate estimations of congestion levels in specific areas through long-term predictions. It was also shown that route prediction could provide major economic and environmental benefits by reducing fuel consumption by up to 7.8% [5]. Route prediction also plays an important role in many applications such as the display of targeted advertisements about points of interest and shops that a user is approaching. Besides, the prior knowledge of future movements of a vehicle could be also valuable in enhancing the quality of VANET as it is a core factor in the design of several location-based routing protocols.

Predicting the vehicle or the driver location is mainly based on the assumptions of spatial and temporal regularities of driving. The spatial regularity relies on the following two behaviors that determine two sub-driving behavior namely individual and global driving behavior. The individual or personalized behavior is based on the observation that a person always tends to take the same route from home to work or some other places (e.g. shopping malls, gardens). Besides, people also share similar movement patterns when they have common location preferences. For instance, people typically drive from A to B by almost following the same route. This phenomenon is called global spatial regularity.
Moreover, temporal regularity relies on the fact that human mobility follows temporal patterns where several trips occur (start/end) at approximately the same time (time of day) or period (workdays, weekends, months, seasons, etc.). For instance, a person may always tend to go to work in the morning (e.g. 8 AM) in workdays. Hence, considering the temporal context of mobility in prediction would improve the prediction. These two assumptions were deduced from observations and then validated empirically by many studies afterward (e.g. [6]).
Based on the above assumptions, a significant number of a person's trips are repeated. Which means that it is possible to predict, if he or she will take that route and at what time of the day.

1.2.1 Problem characterization

The problem of route prediction can be viewed as an instance of the problem of sequence prediction. This latter consists of predicting the next element(s) (in our context a road segment) of a sequence of symbols based on its preceding element(s). Sequence prediction has many applications in various domains such as biological sequence analysis [7], text analysis [8], and web page prefetching [9]. Most route prediction approaches are applied in four steps: (1) collecting mobility data of users, (2) preparing collected data (i.e. pre-processing), (3) training a sequence prediction model, and (4) applying the prediction model to predict future routes.
Numerous route prediction models have been proposed, based on various sequence prediction models such as probabilistic models (the Markov chain model and its variations), data mining techniques (decision tree, sequential patterns mining, etc.), and trip matching techniques.

1.2.2 Challenges

Although route predictors were shown to perform well, several key challenges that attract much attention of researchers remain open research axis.

- **Lossless model design**. Most of route prediction proposals are Markov chain based models. They mainly assume the Markovian hypothesis that the next location of a mobile user only depends on its current location. This hypothesis has been employed as a simplifying assumption

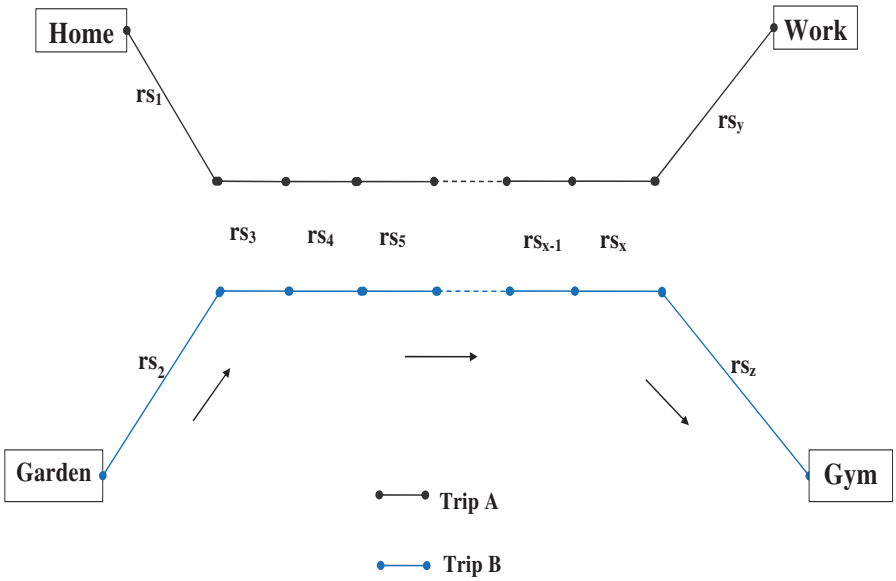


FIGURE 1.2: Two sample routes frequently visited by a driver.

because it allows building models with small sizes. However, this assumption is often unrealistic and thus greatly decreases the accuracy of route prediction.
For the sake of illustration, consider the example depicted in Figure 1.2, which depicts two similar routes (Trip A and Trip B) that a person often takes to respectively go from home to work and from a garden to the gym. Based on this data, if the driver is in road rs_x, a route prediction model could predict that the driver will next be in rs_y or rs_z. However, if only the previous location (road segment) is considered, it is impossible to distinguish between these two routes to determine whether the next location is more likely to be rs_y or rs_z.
An obvious solution to this problem is to consider more than one route segment to perform predictions. Some higher-order Markov models consider up to the last k segments of a route to predict the next segment (Markov models of order k). Nevertheless, a major limitation of these models is that their sizes exponentially increase as a function of the number of route segments that are considered. Thus, large values of k cannot be used in practice with these models, even though real-life trips often contain a large number of segments. Third, another important limitation of current route prediction models is that they are lossy in terms of information. In other words, these models are trained with summarized historical data. As a result, they induce a discard of information found in historical data. Therefore, the development of lossless predictors is necessary.

- **Noise tolerance and real-time implementation**. Noise sensitivity toward the mobility patterns learned is another drawback of existing

route prediction models. Relying on these models, the smallest deviation in trajectory data affects the prediction result, which the prediction accuracy. This problem worsens when dealing with noisy mobility datasets such as GPS trajectory data that are prone to many disturbances and inaccuracies.
Moreover, some prior proposals were designed without regarding any constraints on computational and memory limitations of mobile devices. This makes the real implementation and deployment of these solutions unpractical in an online environment. Hence, future route prediction models should tolerate noisy data and fit mobile devices' physical limitations.

- **Personalized and global predictions**. Many route prediction proposals have addressed the prediction problem considering only one type of human mobility behavior and there were only a few attempts to study the impact of both individual and collective behaviors on personalized and global mobility prediction. Thus, considering both behaviors is required.

1.3 POI recommendation

Social relationships among users have been also taken into account in human mobility leading to the development of location-based social networks (LBSNs). These networks (Figure 1.3) have introduced new research challenges in the area of location-based systems. To help in filling the gap between the online social networking services and the real world, LBSNs have extended the concept of social relationships in conventional social networks (SNs) to incorporate the human mobility context expressed as contents in SN, such as geo-tagged media, location, and trajectory-based information.

Nowadays, LBSNs such as Foursquare [1] have attracted millions of users, and have collected billions of check-in points-of-interests in their databases. For instance, in June 2016, Foursquare has recorded more than 8 billion check-ins counting more than 65 million locations around the world with a number of users that exceeds 55 million each month [10]. Each user in LBSN maintains a profile (e.g. name, age, address, etc) and reports his check-in locations to the LBSN service provider. A check-in made by a user indicates a visited venue, also called a Point of Interest (POI) (e.g., a restaurant), and user comments and appreciations of that location. A POI is a geographical area with a specific role (e.g., hotel, restaurant, etc.) that the user may find beneficial or interesting.
To further enhance the user experience in LBSN, the service of POI recommendation are proposed by service providers. It consists of suggesting new points-of-interest to a user relying on check-in records, friendship relations, and many other factors related to users mobility behavior such as geographical [11–16], temporal [16–18] and social [15, 19–21] influence factors.

The geographical influence factor is based on the assumption that a user is more likely to visit nearby locations than faraway ones. The temporal influence factor indicates that a user destination is generally time-dependent. For instance, a user may tend to go to his workplace in the morning and to eat at a restaurant at lunch-time, during weekdays, whereas he may visit shopping malls during weekends. Finally, the social influence factor is based on the idea that friends often share common interests, tastes, and preferences. In other words, users may follow their friends' recommendations and visit the same places. For example, friends may work out at the same gym and sometimes eat at the same restaurant.
Effective POI recommendation is beneficial to both users and LBSN service providers. It lets users benefit from the previous experiences of other users (in particular, friends) to decide whether a location is interesting and should be visited especially to users who are visiting unfamiliar areas (e.g. tourists). For LBSN service providers, an economic profit is made by advertising locations to only potential visitors instead of a large number of users.

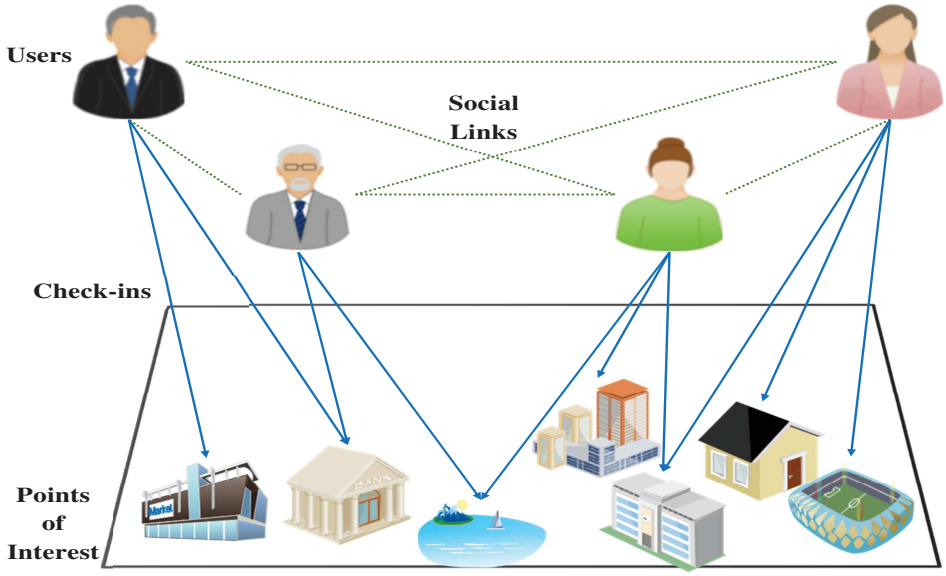


FIGURE 1.3: Location-based social networks.

1.3.1 Problem formulation

POI recommendation consists of recommending to users a set of unvisited location that might be of interest to them based on historical check-in records of other users. In the last decade, POI recommendation has become a popular research topic where numerous POI recommendation approaches have been developed. The most applied technique is collaborative filtering (CF)

[11, 12, 22–35] which exploits user ratings and preferences to perform recommendation. To enhance the quality of POI recommenders, the geographical [11–16], temporal [16–18] and social [15, 19–21] influence factors have been considered to further extend traditional CF approaches.

1.3.2 Challenges

Although conventional CF recommender systems are somewhat effective, they do not consider the current location of the user when recommending the next location. These recommenders ignore the sequential nature of human mobility behavior [36,37]. In other words, CF recommenders use check-in data but do not consider that the needs of users vary depending on their recent movements and locations. This sequential behavior of movement is called sequential influence. By considering the sequential influence, a novel recommendation task called successive POI recommendation has emerged.
In recent research, the task of successive POI recommendations (e.g. [36,38]) has been broadly studied. Recommenders for this task consider the hidden sequential correlation in the mobility behavior of users to perform recommendations. In particular, given the current location of a user and his historical movements (if available), a set of POI suggestions are generated to recommend locations that the user may be interested in visiting in the near future. For successive POI recommendation, an important challenge is to be faced.

- **Considering sequential influence with variable ordering tolerance**. Most of the successive recommenders use Markov chains [36,39–41] or sequential patterns mining [14,23,37,42–44] based models. Specifically, k-order Markov models consider the last k locations visited by a user to recommend new places. These proposals have the advantage of being simple and easy to implement. However, they could require a large amount of memory when the order, k, of a Markov Model is set to a large value. Moreover, a small variation in the order of previously visited locations implies different recommendations even when the same set of locations has been visited.
  Sequential patterns mining based approaches have been also used for successive POI recommendations [14, 23, 37, 42–44]. These proposals discover frequent subsequences of locations (i.e. patterns) visited by users and exploit these patterns to perform recommendations. A sequential pattern is said to be popular or frequent if it appears more than *minfreq* times in location data, where *minfreq* is a threshold set by the user. Although sequential pattern mining based systems allow considering sequential influence, they have three main drawbacks.
  First, locations in a sequential pattern are ordered by time but may still not represent locations that are consecutively visited by users (some locations may be ignored). Therefore, recommendations based on a sequential pattern may represent locations that are far from each other

and are often only visited after visiting many other places not appearing in the pattern. Thus, the recommendations may sometimes be irrelevant. Second, the process of mining sequential patterns is greatly influenced by the setting of the minimum frequency (*minfreq*) threshold which is made on the basis of trial and error. Therefore, it is difficult and time-consuming to find an appropriate value for this parameter. Finally, and similarly to Markov models, recommendations based on sequential patterns assume that the order of locations must be followed in the same order by the user. Because of this very strict ordering constraint, a small variation in a sequence may lead to completely different recommendations. Hence, the development of successive recommenders that tolerate order variation is mandatory.

1.4 Contributions

Our research for this thesis leaded us to three main contributions:

- **NextRoute**. To deal with the challenges of noise tolerance, personalized predictions, and lossless design of the prediction model, we propose in this research a route prediction model called *NextRoute*. This model adopts a novel, accurate, and lossless prediction model called Compact Prediction Tree (CPT) [45]. CPT is considered as a lossless model because it provides the benefit of preserving all the information contained in training sequences to perform predictions.
  Yet, and regardless of CPT's lossless nature that allows it to preserve all relevant data, a significant data reduction is attained through its prediction structures and its FSC(Frequent Subsequence Compression) and SBC(Simple Branches Compression) advanced compression strategies [46]. In particular, FSC and SBC strategies provide a significant space reduction especially when many trajectories share common routes, thus, greater compression can be achieved.

  Besides, CPT implements an indexing mechanism that permits a very fast vehicle's path searching and matching. In addition to its fast prediction process, and unlike several other prediction models that are noise sensitive, CPT is noise tolerant. It copes with noise found in data by adopting a similarity measure mechanism that tolerates variations in mobility sequences while performing predictions. This property of noise-tolerance, along with CPT's previously mentioned properties, justifies our choice of CPT as an efficient model to deal with GPS mobility data.
  Furthermore, to consider the individual and global driving behavior, *NextRoute* implements two CPT based models, named PM (Personal Model) and GM (Global Model), designed to perform personalized and global predictions. They consider individual and collective movement patterns and the similarity between trajectories to make predictions.

- **MyRoute**. To deal with the challenges of noise tolerance, real-time design, and personalized prediction, a real-time graph-based approach for route prediction called *MyRoute*, which adopts the Dependency Graph (DG) predictor was proposed. The central idea of our approach is that graphs can perfectly represent the structure of road networks, and therefore accurately model vehicle movements. Moreover, this work is also based on the idea that vehicle mobility is order dependent as a vehicle passes through a sequence of route segments to reach some location of interest, by following a specific order and traversing road segments in specific directions. To forecast the next location, *MyRoute* utilizes the current location of a vehicle and its historical data, if available.
  The proposed approach first builds a prediction graph, where the nodes or vertices represent the road segments and the arcs represent the traversal order of road segments by vehicles. Then, the prediction graph is utilized to predict the next route of a driver by attempting to match its current trajectory with graph paths. However, graph matching is difficult due to the presence of noise in data. To address this issue, the proposed approach creates graph arcs not only between consecutive road segments (nodes) in a path but also with the following road segments appearing within a user-defined *lookahead* window. By doing so, the noise found in the data is ignored which achieves an improvement in the prediction accuracy.
  To provide high prediction accuracy, the proposed model also offers great flexibility in terms of considering the road segments in trajectories when matching a current trajectory of a vehicle with its previous trajectories, unlike some prior models which match GPS trajectories.
  Moreover, to model both global and personal mobility behaviors, two prediction mechanisms are proposed called the GMG (Global Mobility Graph) and PMG (Personal Mobility Graph).

- **STS-Rec**. To incorporate the sequential influence in POI recommendation, we propose in this research STS-Rec, a sequential rule mining based recommender system. It addresses the main drawbacks of successive recommenders and considers sequential, temporal, and social influences between check-in records to perform recommendations. STS-Rec efficiently discovers patterns indicating the evolving periodic sequential behavior of human mobility by loosening the strict ordering constraint on location sequences and discovering POSR (Partially Ordered Sequential Rules) [47]. This kind of rules allows establishing relationships between locations where locations in both the antecedent and consequent of a rule are unordered. Besides, STS-Rec also extends the concept of sequential rules with the use of a sliding-window (a window-size constraint). This constraint allows extracting recommendation patterns of locations that appear within a maximum number of consecutive locations in location sequences. Through this technique,

the sequential relationship between locations is considered to avoid recommending distant locations. Besides, STS-Rec considers social influence by extracting patterns from users that are socially correlated.

1.5 Organization of the thesis

The thesis is organized in 7 chapters. In the second chapter, we review the field of route prediction. We introduce preliminaries related to this field and formulate the problem of route prediction. We, also, describe the most relevant prediction techniques, and how they were used in earlier works. At the end of this chapter, a comparative table of most significant works is presented emphasizing the main limitations of the existing models.
In **Chapter 3**, we present the NextRoute model, our first contribution of route prediction. A detailed description of the model is provided.
In **Chapter 4**, we describe MyRoute model, our second contribution which concerns real-time route prediction with noise tolerance ability by presenting its architecture and main features.
In **Chapter 5**, we present a literature review of the field of POI recommendation and describe the most relevant studies in this domain.
In **Chapter 6**, we expose the STS-Rec recommendation system, our third contribution which is a sequential rule mining based recommender system. We describe its architecture and its different modules.
Finally, a conclusion of our work and suggestions of possible future research directions are given in **Chapter 7**.

Chapter 2

Literature Review on Route Prediction

2.1 Introduction

In the field of ITS and LBS, designing techniques to predict routes and destinations of mobile users has become a very active research topic [48]. The proposed studies can be classified in terms of various factors such as 1) the types of predictions that are performed (route and/or destination predictions, personalized or global predictions), 2) the techniques used for carrying out predictions, and 3) the prediction range (short or long-term). This chapter begins with some general definitions and preliminaries related to route prediction. It then presents the related work structured in a taxonomy based on the technique used. Next, a comparison study and a discussion that reveal some limitations of the existing works are presented. These limitations defined the issues that lead us to our proposals in the present thesis.

2.2 Predictibility of human mobility

Nowadays, the rapid advancement of mobile technologies has led to the development of various handheld and wearable mobile devices that have become instrumental tools for most of our daily tasks. These devices, such as smartphones and other communication devices, have been steadily enriched with built-in positioning sensors and receivers, like GPS, for navigation and orientation. This positioning feature in mobile devices constitutes a rich source for human mobility data as they are generally held by users as they move. This has allowed collecting a large amount of high-resolution movement traces which, in turn, has stimulated the emergence of the mobility prediction task.

2.2.1 Mobility prediction

Mobility prediction consists of forecasting future locations and positions that a human being would visit or be interested in visiting in the future (that could be short or long-term prediction). A location L_i is abstracted by using absolute coordinates such as the GPS latitude and longitude coordinates or using

symbolic representation (e.g. cell IDs in GSM networks). In some situations, both may be available such as in check-in data in location-based social networks.
The underlying assumption hereby (of mobility prediction) is that a human being exhibits regular and predictable movement behavior that is well-noticed by the great degree of spatial and temporal regularity that appear in his mobility patterns.

Mobility prediction, thanks to the improved ability to track people's locations, has been implicitly or explicitly utilized in many areas of wireless system design and pervasive computing. It has also shown its utility in a diverse array of applications such as controlling the spread of infectious diseases, targeting advertisements, optimizing intelligent transportation systems. For wireless networks, mobility prediction mainly serves intelligent resources and cell allocation in cellular networks and designing location-based routing protocols in VANET.
In the field of intelligent transportation systems, active research has been conducted in predicting driving and walking routes and destination intents. Such predictions are useful for transportation routing and planning, traffic management, optimizing energy efficiency in hybrid vehicles, giving the driver warnings about upcoming traffic hazards, etc.

2.2.2 Regularity of human mobility

Temporal and spatial regularities are mainly based on the fact that people follow daily or weekly routines and have only a few frequently visited locations. Despite the apparently, a considerable number of works have captured significant regularity and predictability in both temporal and spatial spaces in human mobility patterns.
Using cellular tower data, the researchers in [49] have shown that a predictability of 93% could be achieved, which demonstrates that human mobility is highly regular and predictable. Using mobile phone logs of 100,000 users the authors in [50] have shown a high degree of mobility regularity among several highly visited haunts. These two researchers have also insisted that periodicity is an oft-occurring phenomenon in the movement and thus, the time factor plays an important role in the mobility prediction. The movement patterns of humans vary from one time period to another (e.g., weekdays vs. weekends) where people tend to follow similar mobility patterns when visiting workplaces or taking a child to school in workdays whereas, gardens and other entertainment locations are typically visited in weekends.
In the field of ITS, Krumm et al. [51] have found that, drivers observed for at least 40 days, duplicate nearly 60% of their trips. Thus, the driving of vehicles is also routine. Even though shortest and fastest routing algorithms are frequently used, drivers always tend to choose the routes they prefer when driving from the same source and destination. This finding facilitates the exploration of human mobility and its understanding. It is also behind the

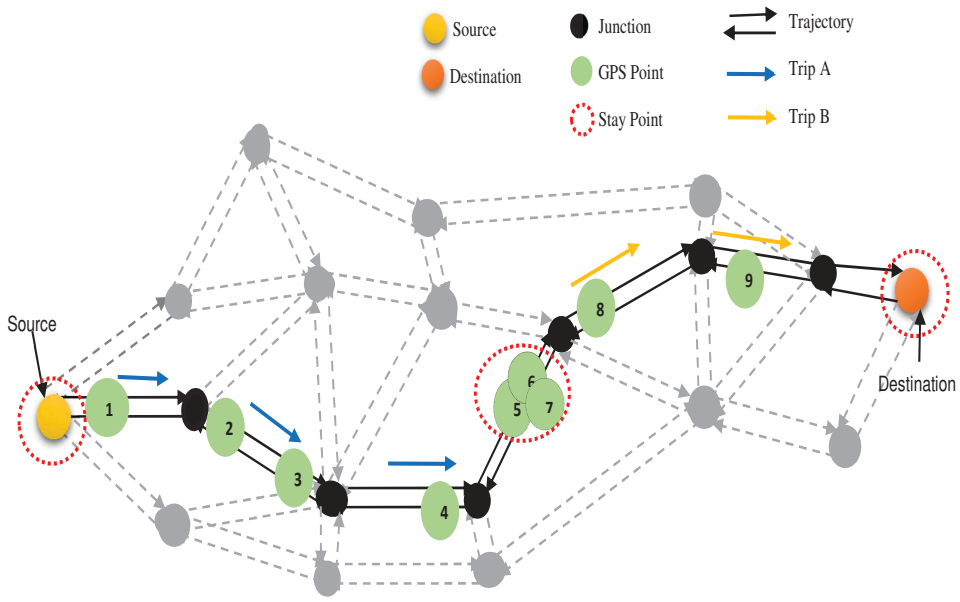


FIGURE 2.1: Illustration of a GPS trajectory.

emergence of the idea of route prediction which is one of the most prominent tasks in mobility prediction.

2.3 Route prediction

Predicting vehicle routes has attracted a lot of attention from researchers in recent years [48]. Current route prediction models mainly rely on human trajectory logs to perform prediction. In typical uses, people tend to leave digital records, in the form of GPS trajectories, obtained from GPS receivers located in the vehicle onboard navigation system or the driver's handheld mobile devices (e.g. smartphones). These trajectories are gathered thereafter to form the location history H of the driver or the vehicle.

Definition (Trajectory). A trajectory is a set of consecutive GPS points that forms the mobility of a person. Thus, $Traj =< p_1, p_2, \ldots, p_n >$, where $p_i\ (Longitude, Latitude, Timestamp) \in P$, and P is a set of GPS points. For instance, Figure 2.1 illustrates a trajectory composed of 9 GPS points.

Relying on the set of drivers' location histories, and likewise mobility prediction, route prediction consists of forecasting the future routes that the person will be driving on. In the following, we will give a formulation of the problem of route prediction. Prior to that, we will define two important terms in that formulation, which are road segment and road network.

Definition (Road Segment). A road segment is an abstraction of a vehicle location. A road segment r_i is a unidirectional edge between two nodes (road junctions) [52]. For instance, Figure 2.2 depicts road segments r_x and r_y connecting two junctions.

Definition (Road network). A road network is represented in a conventional way, as a directed graph, denoted as (N, RS), of $n = |N|$ nodes and

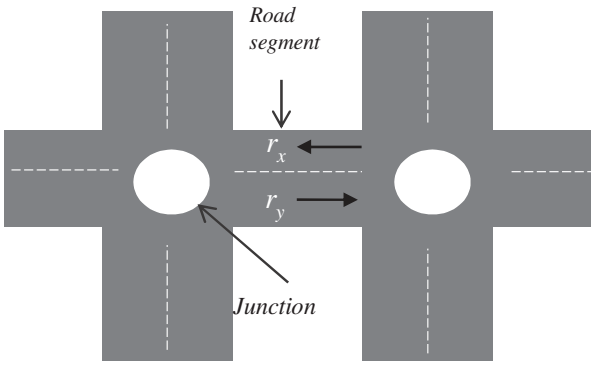


FIGURE 2.2: Illustration of a road segment.

$|RS|$ edges. The set of nodes N is a set of vertices representing the road junctions, intersections, dead ends, and road name changes in a given area. The set of edges RS is a set of arcs, representing all the road segments joining those junctions' nodes.

Definition (Route prediction). Given mobility history H containing all trajectories of a set of moving objects O, route prediction aims to predict the set FR of the future road segments to be traversed by each moving object $o \in O$ with $FR = \{r_k, r_{k+1}, \ldots, r_h\}$ and $r_i \in RS$. Thus, a moving object could be a person or a vehicle and prediction can be extended to reach the final intended destination.

2.3.1 Phases of route prediction

Most of route prediction approaches consist of a chained process of three stages: (1) collecting data about driver paths, (2) pre-processing or preparing the data, and (3) prediction to construct and apply the prediction model to predict routes (Figure 2.3).

2.3.1.1 Data collection

Using positioning tacking devices, this stage consists of periodically collecting mobility data of drivers comprising their locations, with other contextual data (weather conditions, traffic status, etc.). These positioning devices have made various types of mobility data available to the industry and researchers such as GPS trajectories [51], cellular tower data [53], Wi-Fi signals [54], smart card transactions [37], and location check-ins from online social networks [16, 18].

- **GPS data**. The most used data for prediction is collected using the satellite navigation system GPS (Global Positioning System). This type of data commonly includes the longitude, latitude, and time, and is obtained from GPS positions recorded by the vehicle built-in GPS navigation system (sensors) or driver's handheld mobile devices like smartphones, laptops, GPS receivers, smartcards, bus-trip records, etc. GPS

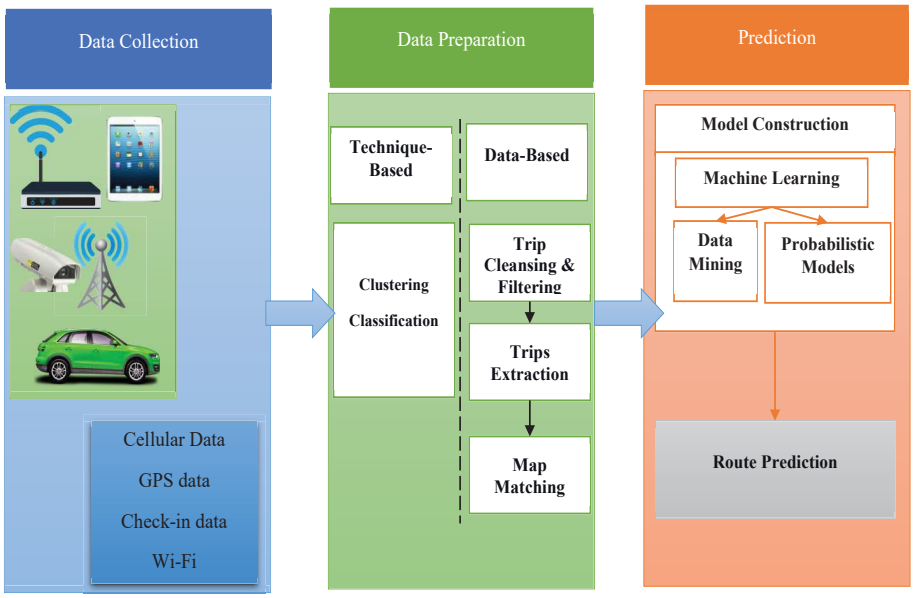


FIGURE 2.3: The main phases of route prediction.

datasets could be dense or sparse depending on the sampling rate setting and battery capacity of the GPS devices.
Although GPS data are widely used, the quality of this type of data is not always consistent as it is prone to several disturbances, noise, and inaccuracies.

- **Other data types**. Vehicle or driver position can be also acquired when it/he passes through or enters a coverage zone of a sampling location. The latter refers to a point or a region where infrastructure or mobile or fixed equipment such as surveillance cameras, cell towers in GSM networks, and access points in wireless networks, are installed. For instance, vehicles are photographed when they pass the surveillance cameras installed over highways and streets, and the vehicle passage records include the vehicle IDs, the locations of the cameras and the time.
  Check-in data, collected from the activity of users in online location-based networks such as Foursquare [1], constitutes another important source of mobility data. Typically, check-in data are composed of Location ID, user ID, time in addition to GPS coordinates of the location visited by the user. This kind of data is usually sparse and sporadic. Location prediction based on this type of data is more challenging than on high-frequency datasets like GPS data as most of the user mobility is missing, which impede the exploration and the understanding of mobility and affects the prediction thereafter.

Although it is possible to use the unprocessed mobility data directly in prediction; their continuous storage could exceed the capacity of current in-vehicle computing systems as more and more daily trips coming in. For instance, and due to GPS uncertainty, trips sharing exactly the same routes

may have a completely different set of recorded GPS coordinates, which consequently leads to the storage of several instances of the same trip. To accommodate this issue, the pre-processing of mobility data is required.

2.3.1.2 Data preparation

This stage aims to extract useful information from mobility data collected in order to be inputted in the prediction model (Figure 2.3). Given the raw of mobility data, the steps to be conducted are either technique-dependent or data-dependent or both.

A) Technique-dependent processing. In this category of data preparation, processing mobility data mainly depends on the final or intermediate purpose of data analysis and the technique used in processing. For the sake of illustration, let us consider that the purpose and the technique to be used is classifying or clustering mobility data. In this situation, the steps of feature selection and feature extraction commonly used in such cases are adopted. These steps that aim at measuring the similarity or dissimilarity among trips are utilized thus to segment data into distinct groups.

B) Data-dependent processing. On the other hand, data-dependent processing relies on the type of mobility data used: GSM, GPS Wi-Fi, check-in data, etc. In the context of location and route prediction, location is an important component. It is considered the ultimate input and goal at the same time as it is used to train the prediction model that forecasts locations at the end (afterward). The location ID is explicitly presented in GSM, Wi-Fi, check-in datasets, which eases the extraction of locations. While for GPS data, this information is not readily available which requires additional treatment to extract the set of locations from GPS trajectories. In what follows, a general description of the main steps required to prepare GPS data, considered in our research for prediction, is presented.

- **GPS data processing**. Processing GPS data is a challenging task that consists of extracting most of the useful information (location ID, speed, etc.) required for travel modeling purposes. The typical GPS processing aims to transform the raw of GPS measurements to a plausible set of discrete trips including the sequence of locations traversed by the driver and other information. To do so, a three-stage procedure is applied (Figure 2.3).

  1. Trip cleansing and filtering. As mentioned earlier, GPS coordinates are often noisy and uncertain is some cases. The GPS data points, therefore, need to be cleaned to discard invalid points such as outliers and filtered to identify and eliminate faulty GPS readings. Outlier points are mainly characterized by their far distance from their temporally nearby

points in the trip, these points thus could be easily discarded by computing local speed and acceleration from neighboring points. Faulty GPS readings are also removed from trip data in the filtering step.

2. Trips extraction. GPS trajectories comprise the coordinates of locations visited by users every time interval; yet, GPS data cannot automatically give an explicit indication of when a trip begins or ends.

Definition (Stay point). A stay point is a geographic area indicating where the driver stayed for a period greater than a time threshold T^{thre} and the distance between GPS trajectory points doesn't exceed a spatial threshold D^{thre}. More formally, for a given subset of GPS points $SP = \{p_m, p_{m+1}, \ldots, p_n\}$, a stay point is detected if, for $m \leq i \leq n$, $Distance(p_m, p_i) \leq D^{thre}$ and $p_m.t_m - p_i.t_i \geq T^{thre}$ with $p_i.t_i$ is the *timestamp* of the point p_i. As depicted in Figure 2.1, a stay point around p_5, p_6, p_7 is detected where: $p_7 - p_5 \leq D^{thre}$ and $p_7.t_7 - p_5.t_5 \geq T^{thre}$ for $Traj =< p_1, p_2, \ldots, p_9, p_n >$.

Definition (Trip). A trip is a GPS trajectory that starts and finishes with a stay point; hence, stay points are trip delimiters. Note that the driver's staring points are also considered as stay points.

To extract the set of trips presented by their GPS coordinates, the set of stay points should be first detected and then discarded from GPS trajectories. Stay points take place in several situations: when the vehicle is stationary or the GPS logger is turned off. Notice that stay points may also occur when a driver passes through a hot congested area leading to his stop or his low velocity. It is worth mentioning that selecting the best thresholds values is challenging as it is difficult to differentiate between brief stops that are valid ends of a trip (e.g., reaching the desired destination) and normal traffic stoppage due to traffic jams or long traffic lights. For that reason, these values were empirically set in many earlier proposals.

3. Map Matching. To convert the trips extracted from the previous step into a set of road segments, the *map-matching* process is applied. Map-matching or GPS snapping denotes a procedure that assigns geographical objects to locations on a digital map. In the context of route prediction, map-matching consists of finding a corresponding block (road positions or road segment) for a GPS observation (time-stamped latitude/longitude pair) in the road network.
This process is a critical step in location sequence generation predominantly in the case of poor quality of GPS data and/or an inaccurate map used. It is usually uncertain especially if some of the GPS records are in between two possible routes and where streets are very narrow

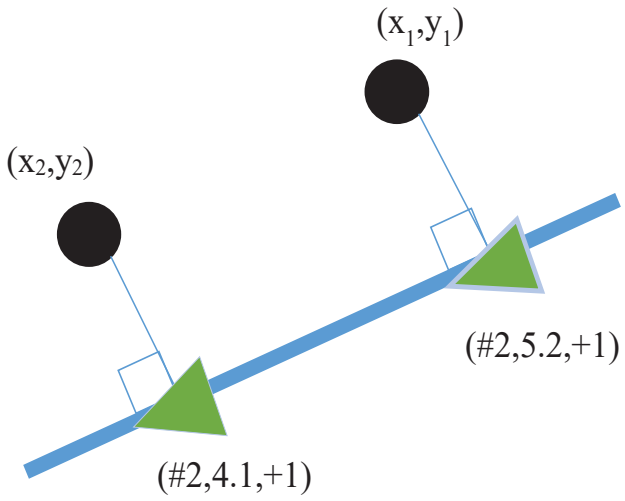


FIGURE 2.4: Point-to-point map matching.

and close to each other. Hence, the more accurate is map-matching the more exact is the location sequence generation and the more effective is the prediction. In literature, two main approaches can be distinguished namely point-to-point and geometric matching.

(a) ***Point-to-point matching***. Point-to-point matching simply consists of snapping each GPS record to the nearest road. For instance, in Figure 2.4, the dots represent the vehicle's GPS locations, and the triangles represent the corresponding positions map-matched onto the road network. In point-to-point matching, GPS records are typically mapped to the closest road in the digital map.

Nevertheless, point-to-point matching is not always effective. In many situations, affecting GPS observation to the closest road may generate a wrong matched path. In the illustration presented in Figure 2.5, the actual path can be easily recognized, however, the second and the third points would be mismatched (represented with red crosses) if they were associated with the nearest road.

(b) ***Geometric matching***. In the sight of the drawback of point-to-point matching, geometric matching methods have been proposed. It consists of finding the sub-polylines or possibly smoothed curves with similar geometry in the digital map that represent the path of the vehicle expressed with its sequence of GPS observations.
However, purely geometric approaches have their sensitivity to measurement noise and sampling rate. Clearly, connecting the dots of a set of noisy measurements sampled at a slow rate would not match well with the road geometry, especially direction information.

In our research, a geometric map-matching model [55], was used. This model consists of finding, with minimum computation, the polylines

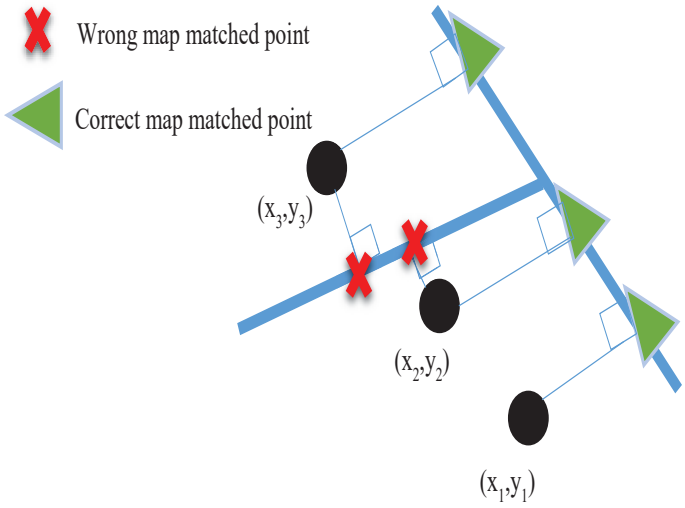


FIGURE 2.5: Wrong point-to-point matching.

describing the curvature of routes traversed by users for a stream of coordinates recorded by a GPS receiver for every second. The model is available as a cloud map-matching based API in [56].

- **Synthetic data processing**. This kind of data is considered as an alternative to real-world mobility data. Due to privacy issues and the lack of large and open real mobility datasets, many researchers were compelled to use simulation data rather than realistic ones. The processing of synthetic mobility data mainly aims to generate mobility sequences or trips from simulation data. In what follows, we explain how to pre-process SUMO traces, a well-known sample of synthetic traces, to retrieve mobility sequences. As depicted in Figure 2.6, discovering mobility sequences consists of first generating the vehicular mobility data from the SUMO scenario files. Second, a pruning step is applied to parse irrelevant data to get and only preserve the path of each vehicle. The output of this step is the set of trips or mobility sequence containing a list of *route segment ID* traversed by each vehicle.

2.3.1.3 Prediction

Data preparation described in the previous is necessary to retrieve information from mobility data and to prepare the data to be in a suitable form for the prediction model (Figure 2.3). The typical prediction stage can be divided into two principal steps. The first step is to construct the prediction model using the pre-processed data whereas the second step is to use this model to perform prediction. In the context of route prediction, most route prediction approaches consist of applying machine learning techniques such as neural networks [57, 58] or statistical models such as Markov models [59–63],Bayesian model [64] and probabilistic trees [65]. More details about these models is presented in the next section.

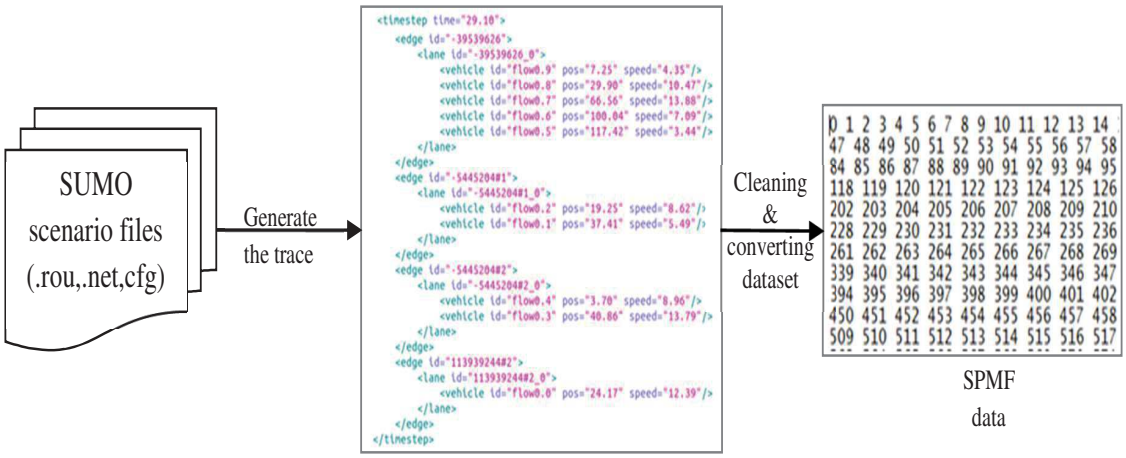


FIGURE 2.6: Generate vehicular trace from SUMO scenario.

2.4 A taxonomy of route and destination prediction models

To review existing models for route prediction, we provide in this chapter a taxonomical classification that categorizes the prediction models according to the technique employed to perform predictions (Figure 2.7). Overall, the proposed techniques can be classified as one of the two following groups: (1) domain-independent techniques and (2) domain-specific techniques. Domain-independent techniques do not explicitly take advantage of the characteristics of the route prediction and mobility data. These techniques are generally based on probabilistic models, data mining models or prediction abilities inherent in data compression algorithms. Domain-specific models comprise trip matching algorithms that have been designed for route prediction in ITS and do explicitly take benefit of their features. Note that some proposals combine more than one type of technique from the same group or from the other group.

2.4.1 Domain-independent techniques

We discuss three categories of domain-independent methods that have been used as the core of techniques for route and destination prediction namely: probabilistic models, data compression methods, and data mining.

2.4.1.1 Probabilistic models

Generally speaking, prediction is always uncertain. Making perfect prediction is not possible due to the nature of driving and future movements are predicted, thus, with a certain probability. For that reason, probabilistic models have been widely used and have shown their efficiency dealing with route prediction problem. Many studies have applied probabilistic models for prediction including: first order Markov [59, 60], k-order Markov [61], hidden Markov [62, 63], Probabilistic Suffix Trees [65], and Bayesian model [64].

Probabilistic models usually follow two main steps. The first step is to assign conditional probabilities to a set of roads in RS of a road network RN, given the user's mobility history H. The second step is to use these values to predict the next location in the mobility sequence.

In literature, most statistical or probabilistic proposals rely on the assumption that previous trajectories of a person may be used to perform route prediction, depending on the order of the Markov model used.

literature, most statistical or probabilistic In what follows, we will first briefly describe the formal definition of Markov model followed by a compilation of the main relevant works based on this model.

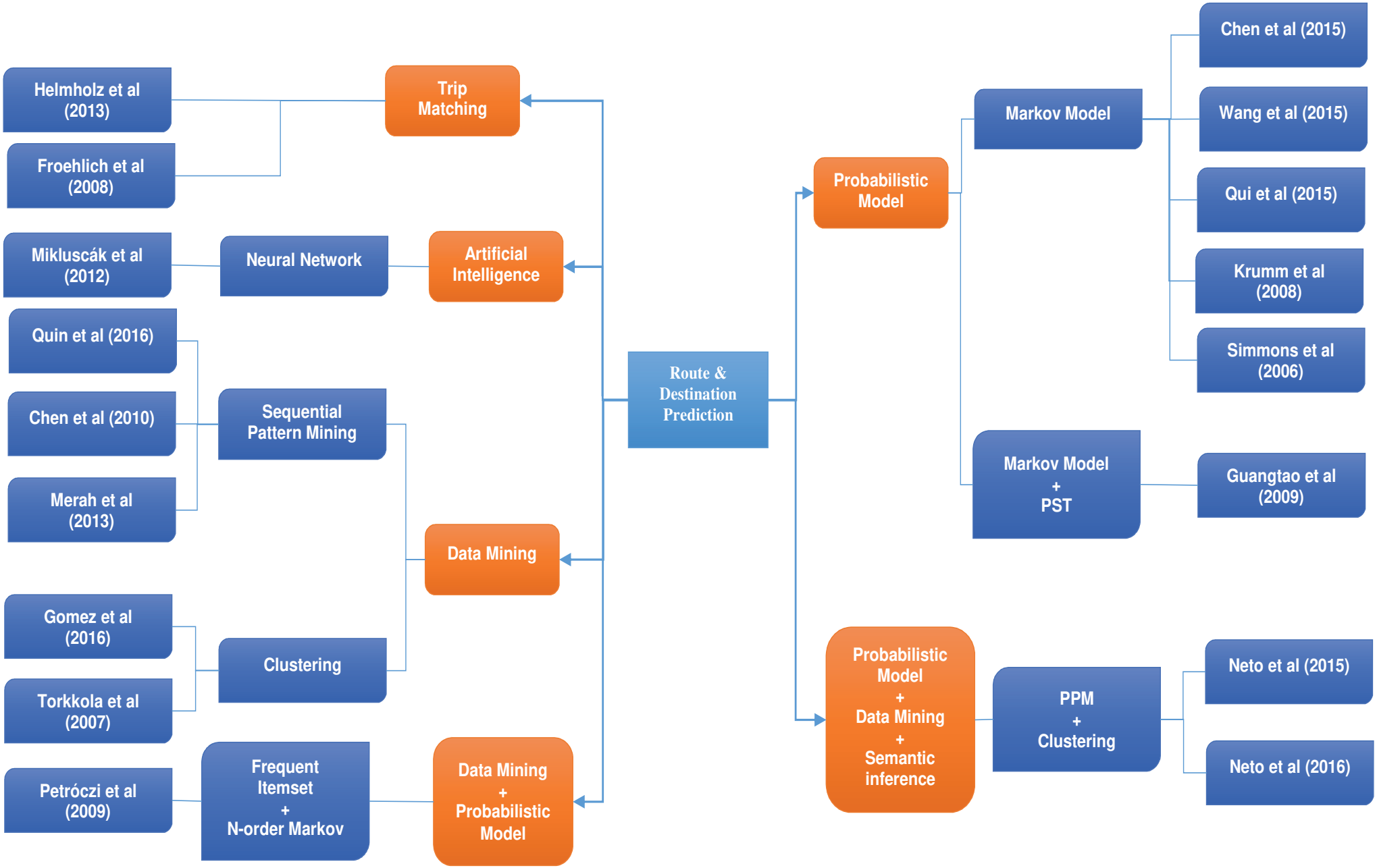


FIGURE 2.7: Taxonomy of route prediction models.

1. **Markov models**. Markov chain models have been widely applied for mobility prediction [51, 59, 60, 66–70]. Generally speaking, a Markov predictor can be formulated as follow.

   Let ms=L_n, L_{n+1}, L_{n+2},..., L_{c-2}, L_{c-1},L_c, L_{c+1}, L_{c+2} ,..., L_m be a mobility sequence of a moving object B where L_c is the current location of B, and L_{c+1}, L_{c+2},...,L_m are the unknown future road segments that we are trying to predict. L_n,...,L_{c-1}, so called *context of prediction* and denoted as CP, is the immediately preceding set of locations previously visited by B and that could go back to the beginning of the movement, if available.
   To perform prediction, the model estimates the conditional probabilities of the form P(L_X | L_s,L_{s+1},...,L_{y-2}, L_{y-1},L_y). This represents the distribution over all locations giving the probability that a location L_X will be visited after visiting the locations L_s,L_{s+1},...,L_{y-2}, L_{y-1},L_y. Specifically, P(L_{c+1} | L_n,L_{n+1}, L_{n+2},..., L_{c-2}, L_{c-1},L_c) represents the probability of visiting the location L_{c+1} after visiting the locations L_n,L_{n+1},L_{k+2},..., L_{c-2}, L_{c-1},L_c.
   With the standard first order Markov model, it is assumed that the probability $P(L_{c+1}|L_n, L_{n+1}, L_{n+2}, \ldots, L_{c-2}, L_{c-1}, L_c)$ for the next location is independent of all but L_c, the current location. Therefore, $P(L_{c+1}|L_n, L_{n+1}, L_{n+2}, \ldots, L_{c-2}, L_{c-1}, L_c) = P(L_{c+1}|L_c)$. Similarly, a second order Markov model is sensitive to the two most recent locations, i.e. $P(L_{c+1} \mid L_{c-1}, L_c)$. In general, we can build the k^{th} order Markov model ($k \geq 1$) to predict the next location to be visited using P(L_{c+1} | L_{c-k+1},...,L_{c-2},L_{c-1}, L_c).

   Markov based route predictor In literature, several works have been proposed using the above predictor in the context of route prediction or location prediction in general. Specifically, krumm et al. [51] have applied a simple Markov model for short-term upcoming road prediction. The proposed approach consists of training a Markov model for each vehicle using GPS data in its long-term trip history. The trained models are then used to predict the path of each vehicle for a few road segments ahead, based on its previous paths.
   To better represent driver's behavior, Chen et al. [66] have proposed an approach for next location prediction, which creates three Markov models called the Global (GMM), Personal (PMM) and Regional (RMM) Markov models. GMM and PMM are used to model, respectively, collective and individual mobility behaviors of moving objects. RMM considers geographic similarities between trajectories by clustering similar trajectories of a person into clusters and then train a sub-Markov model for each resulting cluster.
   Petróczi et al. [61] have proposed three prediction models: (1) a statistical model based on frequent itemset mining, (2) an k-order Markov model where $k < 4$, and (3) a Pattern Matching Model. The latter allows to consider multiple previous items for prediction using a modified Markov model based on k-order

Markov models with a flexible number of items to consider. The obtained results in [61] have shown that the second model based on k-order Markov models gave the best accuracy (70%). In another work, Xue et al. [65] have applied a combination of a Variable-order Markov Model (VMM) and Probabilistic Suffix Trees (PSTs). Their approach utilized multiple VMMs with different traffic conditions for daytime driving to mine mobility patterns from real GPS taxi traces. It provided about 40% prediction accuracy for each vehicle, with a 60% confidence.

2. **Hidden Markov Model**. Hidden Markov Model (HMM) is a Markov chain model that is dedicated to tracking processes with hidden states. In the context of route prediction, Simmons et al. [70] have proposed to use an HMM-based predictor to simultaneously forecast a driver's intended route and destination. The authors define the process as the sequence of driver actions used to follow a route and the hidden states as the driver's intended destination and route. Simmons et al. have then extended their initial model to consider temporal factors, such as time-of-day and day-of-week, for prediction.

   Performing accurate route predictions require precise mobility data. Yet, the process of collecting GPS data is prone to several disturbances and inaccuracies that effects data precision. To cope with uncertainty in GPS data, HMM has been also applied in [63] where Qiu et al. have addressed the problem of prediction using imprecise and noisy mobility data on the top of an HMM-based model. Their proposal consists of using mobility data collected from cell-towers instead of GPS devices as alternative to GPS data. Their proposal has been then optimized to reduce the execution time of the prediction process so that the designed approach could be deployed in real applications.

2.4.1.2 Data compression methods

Data compression algorithms have also shown their efficiency dealing with mobility prediction tasks. Apart from their traditional use in data compression areas, Markov predictors based on the families of these algorithms have attained a substantial predictability. More specifically, predictors based of compression algorithms are universal variable to the fixed-length coding methods that have been mainly proposed as tools to reduce the size of data while building the compression scheme and then carry out sequence prediction. Sequence prediction is performed thereafter by computing the conditional probabilities of a given symbol σ to appear after a context s by detecting the sequence in the compression scheme built. Actually, there is a strong correlation between sequence prediction and lossless compression algorithms, where, any lossless compression algorithm such as PPM [71], LZ [72] can be used for prediction and vice-versa (see, e.g., [73]) .

1. **Prediction by Partial Matching (PPM)**. Prediction by Partial Matching (PPM) [71] is a lossless, probabilistic and sophisticated algorithm for data compression. PPM, as well as many other compression algorithms, adopts a ranking system of symbols equivalent to probability mass function estimation. Each symbol (a letter, bit or any other amount of data) is ranked before it is compressed. The symbols are ranked not based on its frequency in the information source, but by their probabilities to appear after a context formed by the last n symbols. Then, the whole sequence is compressed into a single fraction that is computed according to these probabilities.
   To perform prediction, PPM model utilizes the Markov model principle where the number of previous symbols, n, determines the order of the PPM model which is denoted as $PPM(n)$. PPM uses the set of n previous letters (or a given context) in the uncompressed symbol stream to predict the next symbol in the stream. If no prediction can be made based on all n context symbols a prediction is attempted with $n-1$ symbols. This process is repeated until a match is found or no more symbols remain in context.
   In literature, several variants of PPM have been proposed (e.g. [74]). PPM implementations are among the best-performing lossless compression programs for texts. For instance, the variant $PPMd$ is used by 7-Zip [1] compression software for files in the $7z$ format.

   PPM based route predictor PPM algorithm has been used for personal route prediction in [67]. Their approach builds a probability transition matrix containing the probabilities of moving from each road segment to the others. To deal with the growing amount of personal mobility data and provide real-time route predictions, data reduction algorithms have been also applied afterward on the resulting probability matrices.
   Besides, PPM model has also been combined with semantic inference systems to enrich trajectories with information such as names and types of places. For example, and for the sake to better understand users' behavior, the researchers in [59] have extended movement trajectories semantically using contextual information. Their approach goes through three steps: (1) predicting future routes using PPM, (2) identifying stay points for each user using density-based clustering (DBSCAN) [75], and (3) building a semantic inference system to automatically define the type of places, using APIs services such as Google Places [2].

2. **LZ algorithm**. The LZ [72] algorithm is a popular and a lossless compression algorithm that has attracted massive attention in the field of sequence prediction. The algorithm has been used in several archiving software such as the compress utility of Unix operating systems.

[1] https://www.7-zip.org/

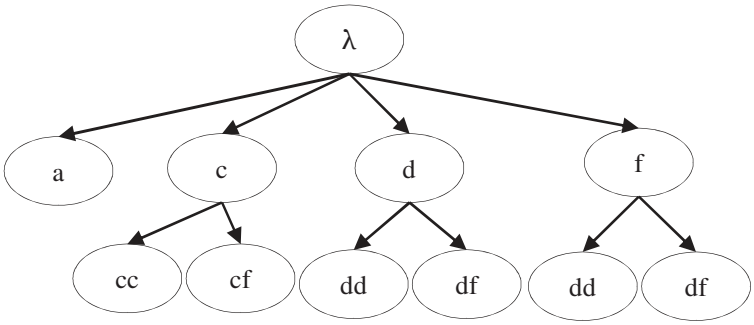


FIGURE 2.8: An example of LZ parsing tree.

The prediction component of LZ consists of a parsing process followed by tree construction and then prediction or probabilities estimation.
The LZ parsing algorithm can be formulated as follow: Let γ be the empty pattern and p is an input pattern. LZ parsing process consists of dividing the pattern p into a set of distinct sub-patterns p_0,p_i,...,p_m where $p_0 = \gamma$, and $\forall j \geq 1$, there exists a sub-pattern p_j such that p_j without its last item (location) is equal to some p_i where $i \in \{0, j\}$ and p_0,p_1,...,p_m=p.
This dividing process is iteratively successive, in other words, after identifying each p_i, the LZ algorithm only considers the remainder of the input pattern p. For instance, given the finite set of items as alphabets $A = a_1, a_2, \ldots, a_n$, the pattern p= acccfdddffffc is parsed as γ, a, c, cc, cf, d, dd, df , f , ff, fc.
Once the pattern is parsed, its resulting sub-patterns are then incrementally inserted in a tree structure to build the LZ tree. Let p_i,p_j be two sub-patterns of p. In the LZ tree, each node represents a sub-pattern and a node containing the pattern p_i is an ancestor of node comprising the pattern p_j if and only if p_i is a prefix of p_j. In each node, a counting value is stored to determine the number of times that the corresponding sub-pattern has been seen as a prefix of other patterns. In Figure 2.8, the LZ tree constructed from the pattern p= acccfdddffffc is depicted.
Finally, the process to perform prediction with LZ-based predictor is conducted as follow. Given a pattern p parsed into p_0, p_i,...,p_m, two subcases could be distinguished. If the node associated with p_m is a leaf in the LZ tree then LZ-based predictor assume that each element in A is equally likely to follow p_m and the probability is calculated as $P(X_{n+1} = a_k|p) = 1/|\mathrm{A}|$ where $a_k \in A$. Otherwise, LZ-based predictor estimate $\mathrm{P}(X_{n+1} = a_k|p)$ based on the items that have previously followed p_m when p was parsed.

In the context of mobility prediction, several LZ predictors have been proposed. To use the LZ algorithm for location prediction, the researchers have to map locations to items. For instance, to tackle the problem of location management in wireless mobile environment and predict the next cell that a user is more

likely to reside in, the authors in [53] have proposed *LeZi-Update* algorithm. This algorithm is a heuristic alteration of the original LZ algorithm. The authors have designed an adaptive online algorithm for tracking mobile users by learning his data with the best message exchange. *LeZi-Update* algorithm was used as the core of an update scheme that is used to decrease the paging cost and improve the paging mechanism. Though the main application of the authors lies in the mobile wireless networks , the central prediction algorithm itself is not specific to this domain.

2.4.1.3 Data mining models

Prediction is a key task of Data Mining (DM); thus, DM techniques have been widely applied for the task of route prediction. Several data mining techniques, such as: sequential pattern mining [76–78], clustering [48, 79], decision tree [77], etc, have been principally or partially applied in an isolated or combined manner with other techniques in the prediction process.

1. **Sequential pattern mining**. A sequential pattern is a type of behavioral pattern that describes the existence of events in a sequentially and/or timely ordered manner. A Sequential pattern can be defined as a subsequence of symbols (or items) that appears frequently in some sequences of symbols. In the context of route prediction, a sequential pattern represents an ordered list of road segments frequently traversed by vehicles during their trips from starting locations to some destinations.
   In [76], Merah et al. have proposed several schemes to collect historical vehicular paths. After collecting paths, a parameter called *the minimum support* threshold (or *minsup*) is set by the user to extract the most frequently travelled vehicular paths. A set of sequential mobility rules are then derived from these sequential patterns and used afterward to forecast routes of vehicles.
   In another work, the authors in [77] have developed the CRPM (Continuous Route Pattern Mining) model to predict the next road segment(s) of routes. The proposed approach is based on the well-known *PrefixSpan* algorithm [80] for sequential pattern mining. CRPM extracts route patterns from historical movement data. These patterns are stored in a tree structure, which is then used to perform predictions. CRPM has been extended thereafter in [78] to simultaneously predict the intended destinations and routes of drivers. Their proposal takes as input real GPS data. It then applies their proposed FBM (Forward–Backward Matching) clustering algorithm to automatically group important places that a user may depart from or go to. Trajectories are then abstracted and frequent movement patterns are extracted using an extended version of the CRPM algorithm.
   The use of sequential pattern mining algorithms has three main limitations. First, determining suitable values for the *minsup* parameter used for pattern

extraction, is often difficult because this parameter is dataset dependent. If we want to obtain optimal results with this approach, we need to spend time fine-tuning this parameters. If this parameter is set too high, few or no patterns are found, while if this parameter is set too low, too many patterns may be found, and the algorithms may become very slow or run out of storage space. Besides, an inherent limitation of these approaches is that they assume that frequent patterns are more interesting than rare patterns. As a result, these approaches ignore rare patterns, even if they have high confidence. Thus, these approaches may not perform well for predicting rare cases. Lastly, as Chen et al. [78] mentioned, it is often very time consuming and computationally expensive to extract sequential mobility patterns, and thus these approaches may have scalability issues.

2. **Clustering** Clustering algorithms consists of measuring and estimating the similarities among objects and group them into clusters according to their similarity degrees. This category of algorithms has been widely used in prediction as a core method in route predictors or partially combined with other techniques. In [79], the authors have presented a location-based traffic advisory system. The latter applies an adaptive clustering algorithm to determine locations where users spend most of their time. Once important locations have been identified, commuting routes between pairs of locations are learned. These routes are then taken as input by a mobile application that determines if congestion will occur along predicted routes. In [48], a novel probabilistic model was developed for personal route prediction. The underlying assumption of this approach is that a route can be uniquely identified using points called *Meaningful Velocity Areas* where user velocity (speed and direction) remarkably changes. These areas are detected with a slightly modified version of an online density-based clustering algorithm called *landmark discovery algorithm (LDM)* [81]. The developed prediction mechanism takes as input a *Multigraph model* where the nodes represent the stop and turn areas and the vertices represent the velocity areas. Each vertex is labeled with a unique identifier, its frequency, and a destination or stop area along a particular route.

2.4.1.4 Other techniques

Several other artificial intelligence methods such artificial neural networks have been used to perform effective predictions.

Artificial neural networks (ANN) are mathematical models inspired from the organization and functioning of biological neurons. ANNs are represented as weighted directed graphs where nodes symbolize the artificial neurons and directed edges (with weights) denoting the connections between neuron inputs and outputs. Relying on how neurons are modeled, two main classes of ANN could be distinguished namely: feed-forward networks, in which graphs have no buckle (self-loop), and

recurrent (or feedback) networks, in which loops occur because of feedback connections [82].
Since their first introduction in [83], ANN have been broadly used to solve a variety of problems in pattern recognition, prediction, optimization, etc. In the context of route prediction, several ANN-based prediction models have been proposed.
For instance, De Brébisson et al. [57] have developed an ANN-based framework to predict the destination of a taxi based on its starting location and associated meta-information such as departure time, driver identifier and client information. In this work, a recurrent bidirectional neural network was applied to encode each taxi's path with its relevant metadata. Moreover, Mikluščák et al. [58] have trained artificial neural networks using historical movement data to perform route prediction. They have compared various neural network architectures with various data representations and coding schemes. The authors have found that prediction accuracy depends largely on the selection of appropriate input weight values, and the choice of the architecture [58]. Ensuring optimal prediction accuracy may thus be time-consuming, as it requires training multiple networks with various architectures using a trial-and-error approach.

2.4.2 Domain dependent models

In this section, we discuss trip matching based models that have been proposed for specific route prediction application domain.

2.4.2.1 Trip matching models

Trip matching is a geographical-based process that relies on a geometric representation of mobility data (e.g. GPS records). This process aims to find the most similar trip for a driver by comparing the current driver trajectory with previously driven trips that belong to that driver and/or other drivers.
Trip matching is performed by comparing points in one trip to points [79] or line segments [84] in another trip using several similarity measures. To measure the similarity between pairs of trips, Hausdorff metric has been used in [84,85]. For instance, Helmholz et al. in [85] have applied a variation of the Hausdorff metric to compare the topological congruence of two trips as well as the driving directions. Helmholz et al. have also extended their model to take the temporal context (time-of-day and day-of-week) into account so the temporal regularity in each user's driving habits is considered. Moreover, in [84] , Froehlich et al. have proposed a long-term route predictor that combines several techniques, including hierarchical clustering for trip clustering and the Hausdorff metric for calculating the similarity between pairs of trips.

2.4.3 Comparative table and discussion

Table 2.1 presents a detailed comparison of the 17 prediction models that have been reviewed in this chapter. The models are compared in this table according to three main criteria: (1) the type of approach used, (2) the type of data used in the experimental evaluation, and (3) the results obtained. Several observations can be made from this table.

Approach

Considering the applied technique, the above review demonstrates that most route predictors are data mining or probabilistic based models which in fact justify our choice of using a probabilistic based sequence predictor for the task of route prediction in this research. Besides, nine models assume the Markovian hypothesis by applying several well-known probabilistic Markov models (simple, k-order, variable-order and hidden Markov chains). Relying on this hypothesis the prediction of the future route to be traversed by a vehicle depends only on a part of information contained in mobility data (i.e. one or few locations previously visited by a driver). However, this hypothesis may not hold in real-life, and as a result, the prediction accuracy of these models can severely decrease [45]. One may think that the solution to this problem is to increase the order of Markov models. For instance, k-order Markov models ($k > 1$) typically consider that only the last k segments of a training sequence should be used to predict the next segment. However, the size of Markov models can increase exponentially when k is increased, thus making this solution impractical for many real-life applications. Another observation is the training process for these models discard a large amount of information from training sequences. Therefore, these models are lossy. However, this hypothesis may not hold in real-life as some relevant information and older driving history may also be useful for prediction. Moreover, by considering all the information contained in mobility data, both frequent as well as rare cases are included which consequently allows mitigating some data mining related inefficiencies (e.g. ignoring rare cases in sequential patterns based predictors).

Mobility data

Considering the mobility data used, Table 2.1 shows that two main types of data traces have been utilized namely: (1) synthetic data, and (2) realistic data collected from GPS devices mounted in vehicles, mobile phones, or other GPS devices.
Many researchers have used GPS traces consisting of millions of location points, collected during different time periods ranging from hours to months (e.g. [70] have collected data , for each driver , during several months).
Having the driving history for different time periods may considerably increase prediction accuracy since more aspects, as well as mobility contexts, can be considered for prediction such as the time of the day, the day of the week, and weeks of the

month. Besides, it is also interesting to note that many works have collected GPS data using mobile phones, and were designed with car navigation kits in mind. Yet, phones have many constraints such as to be running on batteries. Hence, data collected from mobile phones may be incomplete. Moreover, other problems may arise when working with GPS coordinates due to the fact that some GPS tracking devices are more accurate than others, and that GPS data may be incomplete or noisy (e.g. when a user enters a tunnel, building or when users power off their mobile phones during data collection) [70].
Generally speaking, the more the data is available for prediction, the more the prediction is accurate as the training data would contain more information especially about rare cases. Consequently, data size has a big influence on prediction accuracy. For that reason, it is very likely that the results shown in previous proposals would change if more training data were available. In addition, it is worth noticing, from Table 2.1, that the large number of GPS points stated in previous works do not actually reflect how efficient are their proposals. This is due to the following reasons:

- The lack of open and large scale mobility datasets besides taxi traces. Thus, many works have used taxicabs data. Yet, taxi traces do not fully match the criteria of spatial regularity assumed for route prediction where a taxi driver may take many different routes depending on the destination of clients.
- The existence of a small number of adequate datasets which are submitted to governmental privacy and sharing issues that hinder their use for comparison purposes.
- The cleaning and filtering techniques applied in the pre-processing stage that may considerably reduce the number of trips generated from the dataset. For instance, Froehlich et al., [84] have initially collected data for 27,479 trips, but the pre-processing step eliminated about 47,2% of the trips.
- The use of GPS data requires performing map-matching, which consists of determining on which road a vehicle is, based on its GPS coordinates. Map-matching is still an open research topic where no exact and precise method have been proposed due to various characteristics of GPS data such as sparseness, noise, and sampling-rate [86].

Performance

In terms of performance, many proposals have reported excellent accuracy for their prediction models. However, numerous studies have evaluated their model by simply dividing the data into two sets, using, for example 70% for training, and 30% for testing. This approach known as hold-out has well-known limitations. In particular, it is quite possible that the data is not well-distributed between these two sets,

which may lead to overfitting. Furthermore, separating the data into two sets means that a considerable amount of data will not be used for training, which may thus decrease the model's accuracy. To avoid these problems, cross-validation constitutes a powerful general technique to estimate model prediction performance [87].

TABLE 2.1: Literature review for route prediction studies.

Study	**Approach**			**Mobility Data**			**Performance**	
	Type of prediction	***Technique (s) used***	***Types of techniques***	***Trace type***	***Vehicle / user***	***Trajectory***	***Accuracy***	***Validation***
Terroso-Saenz et al. 2016 [48]	Route	Density based clustering +velocity	DM	Dataset 1: Real GPS (Smartphone), Dataset 2: Synthetic	Dataset 1: 20, Dataset 2: not reported	Dataset 1: 10606, Dataset 2: 10074	92%	Not reported
Neto et al. 2016 [59]	Route and destination	Markov Model	PM	Real GPS (GPS loggers)	21	/	Not reported	Cross-validation (k-fold)
Wang et al. 2015 [67]	Route	Markov Model	PM	Real GPS (driving)	1	117+41	91%	Training: 117 trip, Testing: 41 trip
Neto et al. 2015 [59]	Route, destination and stay points detection	PPM for route + clustering and semantic for location	PM+ DM +SI	Real GPS (Smartphone)	8	156	46% for route prediction, 62% for location prediction	Cross-validation
Chen et al. 2015 [66]	Route	Markov Model	PM	Real GPS (driving)	Not reported	Not reported	90.30%	Training: 20 days, Tuning: 7 days, Testing: 4 days
Helmholz et al. 2013 [85]	Route	Variation of Hausdorff metric	TM	Real GPS (Smartphone)	5	500	80%	Cross-validation (leave-one-out)

Qiu et al. 2013 [63]	Route and destination	Hidden Markov Model (HMM)	PM	Dataset 1: Real cell-tower collected data, Dataset 2: Synthetic	Dataset1: 10, Dataset2: 20	Trace 1: 10, Trace 2: 20	/	Cross-validation (leave-one-out)
Merah et al. 2013 [76]	Route	Sequential pattern mining	DM	Synthetic	70	70	Not reported	Not reported
Miklušcák et al. 2012 [58]	Route and destination	Feed-forward Artificial Neural Networks	DM	Synthetic	2000	Not reported	59.79-98,7%	Not reported
Chen et al. 2010 [78]	Route and destination	Time-based clustering	DM	Real GPS (Smartphone)	14	1000	60-80%	Cross validation (K-fold)
Petróczi and Gáspár-Papanek 2009 [61]	Route	Frequent itemset, + k-order Markov	PM + DM	Synthetic	Not reported	14,2	50- best 69.2% with Third-order Markov	Training: 12 000, Testing: 2200
Ye et al. 2008 [77]	Route	Continuous Route Pattern Mining (CRPM) + decision tree	DM	Real GPS (Smartphone)	17	900	74.3%	Training: 90% of data, Testing: 10% of data
Xue et al. 2009 [65]	Route	Variable-order Markov Models+ Probabilistic Suffix Tree (PST)	PM	Real GPS Taxi (driving)	5000	5000	40% conf=60%	Training: data of 6 months, Testing: one month
Froehlich and Krumm 2008 [84]	Route	Hierarchical clustering+ Hausdorff metric	TM	Real GPS (driving)	240	14,468	95-99%	Cross validation (leave-one-out)

Krumm 2008 [51]	Route	Markov Model	PM	Real GPS (driving)	100	1221	90%	Cross validation (leave-one-out)
Torkkola et al. 2007 [79]	Route and destination	Density based clustering	DM	Real GPS (Smartphone, another device)	10	40	Not reported	Not reported
Simmons et al. 2006 [70]	Route and destination	Hidden Markov Model (HMM)	PM	Real GPS (driving)	/	46	98%	Cross validation (k-fold)
Amirat et al. 2017 [88]	Route	Dependency Graphs	PM	Real GPS (driving)	10	33-320	76%-98%	Cross validation (k-fold)
Amirat et al. 2019 [89]	Route	Compact Prediction tree	DM	Dataset 1: Real GPS (driving), Dataset 2: Synthetic	Dataset 1: 10, Dataset 2:200k	Dataset 1: 20, Dataset 2: not reported	35%-96%	Cross validation

DM: Data Mining, PM: Probabilistic Model, SI: semantic inference, TM: Trip Matching.

2.5 Conclusion

In this chapter, we have provided a background and preliminaries on route prediction task followed by a fast overview of different approaches for route prediction. This chapter is not intended to be a comprehensive survey but instead a presentation of the main proposals for mobility prediction in general and route prediction in particular.
In this chapter, we have also mentioned the main limitations of the existing proposals (such as of ignoring rare cases and Markov hypothesis related inefficiencies, etc.) that we have tried to accommodate in the next two chapters by proposing two route-predictors.

Chapter 3

NextRoute: A lossless model for accurate route prediction

3.1 Introduction

This chapter introduces a novel route prediction model named *NextRoute*. The proposed model is lossless as it compresses location data in a prediction tree without information loss, and it is designed to use all the relevant information contained in the training data to perform prediction. In contrast to many other proposals, *NextRoute* provides an efficient noise tolerance strategy that loosened the similarity measure when matching the current trajectory of the vehicle with historical data. In this chapter, a general-purpose sequence prediction model called *CPT* (Compact Prediction Tree) [45] is first described. CPT constitutes the core component of our *NextRoute* route predictor and it provides the benefits of preserving all the information contained in training sequences to perform predictions. Next, the details of our proposal are presented. Finally, an experimental evaluation conducted using real-world and synthetic datasets has shown that *NextRoute* has provided quite encouraging results compared to two state-of-the-art models.

3.2 Compact prediction tree model

NextRoute model is based on the *Compact Prediction Tree (CPT)* sequence prediction model [45]. In this section, we first formulate the problem definition of roue prediction to deal with mobility sequences and then provide a detailed description of the CPT sequence predictor.

3.2.1 Definitions & problem formulation

Definition (mobility sequence) A mobility sequence $Ms =< rs_1, rs_2, \ldots, rs_n >$ is an ordered list of road segments traversed by a vehicle during a trip. For example, Table 3.1 depicts three mobility sequences corresponding to three vehicles V_1, V_2, V_3 where each vehicle's route is expressed by a list of road segment identifiers. For instance, the vehicle V_2 has traversed the road segment rs_1, rs_2, rs_3, and rs_5, in that order.

TABLE 3.1: Three mobility sequences.

Vehicle ID	Mobility sequence
V_1	$<rs_0, rs_2, rs_3, rs_4>$
V_2	$<rs_1, rs_2, rs_3, rs_5>$
V_3	$<rs_1, rs_2, rs_3, rs_4, rs_6>$

Definition (Route prediction problem) Let $TMs = \{Ms_1, Ms_2, Ms_3, \ldots, Ms_t\}$ be a set of mobility sequences used to build a prediction model M. The problem of route prediction consists of using M to forecast the next road segment rs_{n+1} of a given mobility sequence $Ms_k =< rs_0, rs_2, \ldots, rs_n >$.

3.2.2 Model overview

This model utilizes a tree data structure to store sequences. Its main distinctive characteristics compared to other prediction models are that (1) CPT stores a compressed representation of training sequences that is lossless or accept a small loss (controlled by the user), and (2) CPT measures the similarity of a sequence to training sequences stored in its tree to perform each prediction, to ensure that all relevant information is available for each prediction. The similarity measure is noise-tolerant and thus allows predicting the next symbols (or items) of subsequences that have not been previously seen in training sequences, whereas other proposed models such as PPM [71] and All-K-order-Markov [90] cannot perform predictions in such a case. The next sections describe the training and prediction phases of CPT adapted to the context of route prediction.

3.2.3 Training phase

Given the set of training mobility sequences, this phase aims at constructing the tree-based predictor. The latter includes three distinct structures namely: (1) a *Prediction Tree (PT)*, (2) a *Lookup Table (LT)* and (3) an *Inverted Index (II)*. During this phase, mobility sequences are considered one by one to incrementally build these three structures.
The prediction Tree(PT) structure stores and organizes all training mobility sequences by their prefixes. In the *PT*, a node contains a road segment and each training mobility sequence is represented by a branch starting from a direct child of the root node and ending by an inner node (i.e. Partial branch) or a leaf (i.e. Full branch). Accordingly, mobility sequences sharing common road segments share a common path in the tree; thus, similar sequences are compressed.
The prediction tree is constructed as follows: given a training mobility sequence, we search the tree to find if there is any branch that corresponds to the longest prefix of the sequence. If it is found, we concatenate the rest of the road segments of the sequence to this branch. Otherwise, the sequence is inserted to the tree as a new branch by creating a new child to the root node with the first road segment in the sequence. Then, the cursor is moved to the

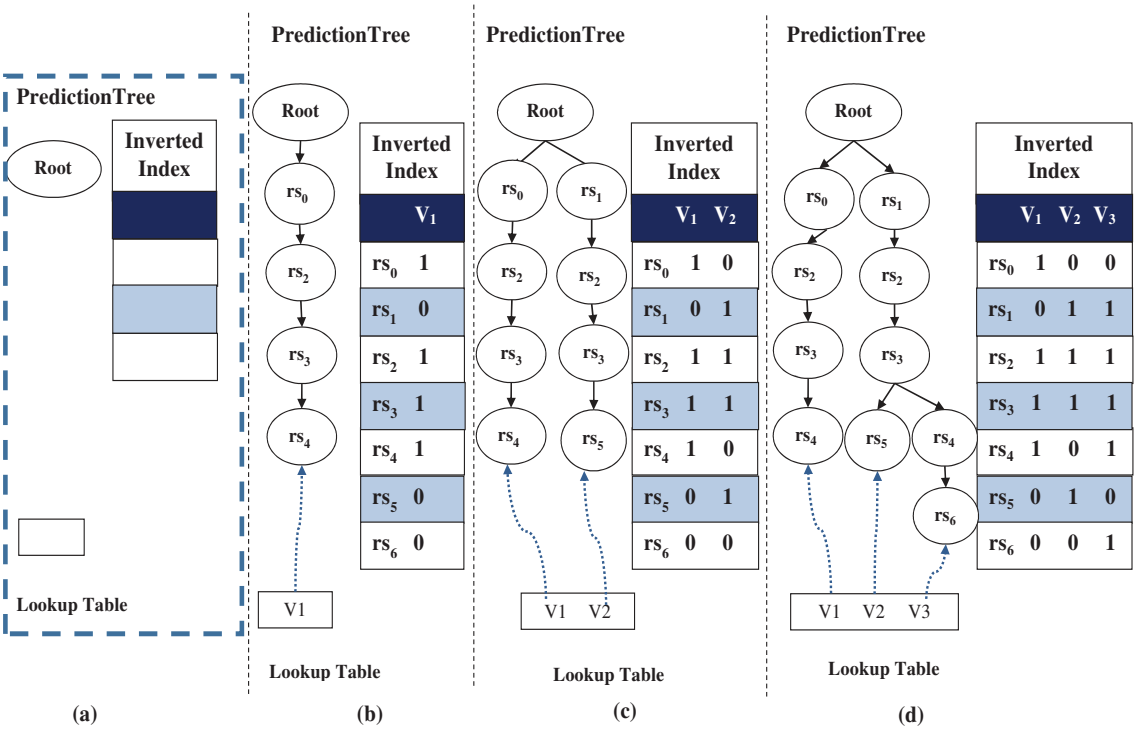


FIGURE 3.1: An example of the CPT training process. (a) the initial state, (b) the model after the insertion of the first mobility sequence $< rs_0, rs_2, rs_3, rs_4 >$, (c) the model after the insertion of the second mobility sequence $< rs_1, rs_2, rs_3, rs_5 >$, and (d) the model after the insertion of the third mobility sequence $< rs_2, rs_3, rs_4, rs_6 >$

.

newly created child and this process is repeated to insert the set of all road segments in the training sequence.

The process of building the tree for N training sequences takes $O(N)$ in time and is done by reading the sequences one by one with a single pass over the data. As for the spatial complexity, this process takes in the worst case $O(N*AveLength)$ where $AveLength$ denotes the average length of training sequences but overall the PT is more compact because the branches often overlap by sharing common nodes.

To further reduce the size of the model, CPT also implements a parameter called $splitLength$. The latter can be used to indicate whether the training sequences should be kept entirely or only their last $splitLength$ road segments are used to build PT. However, in the latter, CPT is no more lossless as parts of data are discarded. For the sake of illustration, Figure 3.1 portrays the construction of CPT starting by an initial state with empty data structures, followed by the successive insertion of the three mobility sequences depicted in Table 3.1.

To speed up frequency computation, CPT adopts a vertical bitmap representation of mobility data so-called *Inverted Index (II)*. The latter is a set of bit-vectors that concisely indicates for each segment rs_i the set of mobility

sequences that it contains. The bit corresponding to sequence S_j in its bitset for the road segment rs_i is set to 1 if the sequence S_j contains rs_i and 0 otherwise. The II is designed to quickly find in which sequences a given segment appears [45]. Similarly to PT, the temporal complexity of II is $O(N)$ where N is the number of training sequences while it requires $((N+b)*u)$ of spatial complexity where b denotes the size of a road segment in bytes and u is the size of unique road segments in R (i.e.u=$|R|$).
The II offers a very fast indexing strategy and it is flexible to conduct Boolean operations in data retrieval. This bitmap structure has been widely applied in various big data frameworks; thus, it is more suitable in our case dealing with the continuous and huge amount of mobility data collected from tracking devices (mobile phone, GPS receivers, etc.)
Finally, the *Lookup Table* (LT) is an associative array that allows locating any training mobility sequences in the PT with a constant access time. The LT associates the II to the PT. For each $sequenceID$, the LT points to the last element of the sequence in the PT (i.e.leaf). Therefore, CPT is a reversible model as it allows restoring original data from data structures. The LT structure takes $O(N)$ and $N*(b+p)$ for time and space complexities, respectively where N denotes the number of sequences, b is the size of the highest *road segment ID* in bytes, and p is the size of a pointer in bytes.

3.2.3.1 Training process complexity & optimizations

The temporal complexity of the training process takes $(O(N))$ where N denotes the number of training sequences whereas its spatial complexity is dataset-dependent. The size of the model takes more or less space depending on the dataset. If many sequences share common prefixes, a greater compression is achieved. To further reduce the size of the model, CPT provides two optimization strategies, namely *FSC (Frequent Subsequence Compression)* and *SBC (Simple Branches Compression)*, that mainly affect the form of the prediction tree by reducing its height and number of nodes.

FSC (Frequent Subsequence Compression). In many situations, sequences may share common subsequences (also called pattern). To deal with such a situation, FSC is applied. It aims at compressing frequent sequential patterns that are found in training sequences. Each pattern found is replaced in PT by a single new segment identifier R_i stored in a new structure named *Subsequence Dictionary (DCF)* and associated with the subsequence that it replaces. The DCF offers a fast way, with $O(1)$ of temporal complexity, to substitute each subsequence into its associated road segment and vice-versa. When introducing new training sequences into PT, the DCF is utilized to substitute previously retrieved frequent subsequences with their single association.

SBC (Simple Branches Compression). It consists of substituting each simple branch (defined as a branch leading to a single leaf) by a single node representing the whole branch. For instance, Figure 3.2 illustrated the resulting prediction tree after applying FSC and SBC on the PT depicted in Figure

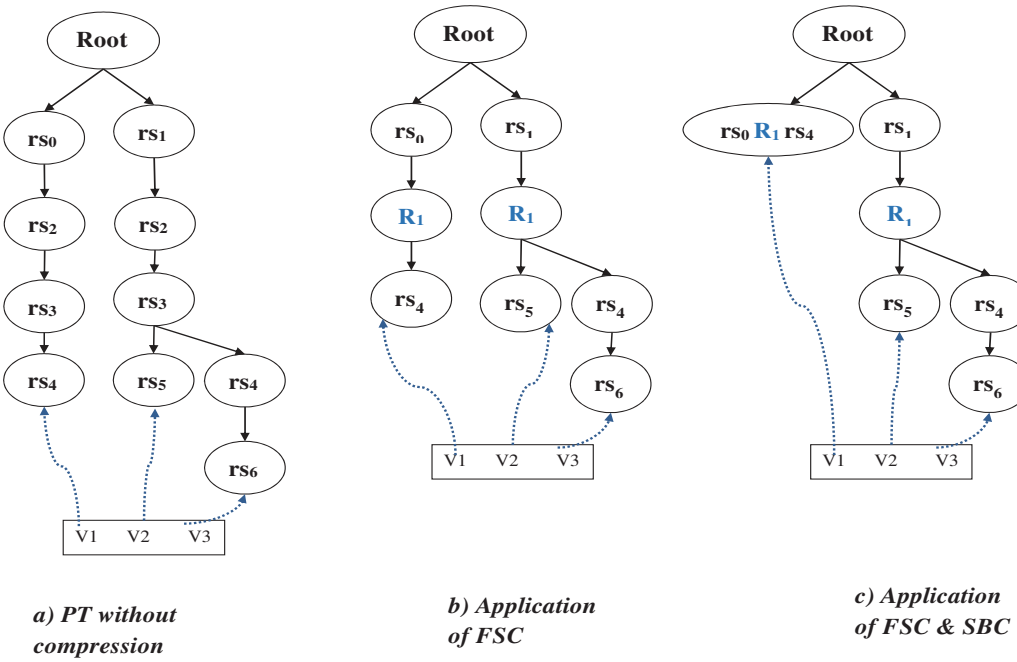


FIGURE 3.2: Application of FSC and SBC strategies.

3.1. By applying *FSC*, the frequent subsequence $rs_2\ rs_3$ is replaced with one node R_1 and stored in the *DCF*. *PT* is further compressed by substituting the whole branch ($rs_0\ R_1\ rs_4$) with a single node using the *SBC* strategy.

3.2.4 Prediction

The prediction phase utilizes the prediction model built in the previous step. It takes as input a mobility sequence $Ms =< rs_1, rs_2, \ldots, rs_n >$ of a vehicle v_i containing n road segments and a g parameter set by the user. The latter is similar to the k parameter of k-order Markov models. It indicates the size of the Last Traversed Road segments $LTRS_g(Ms) =< rs_{n-g+1}, rs_{n-g+2}, \ldots, rs_n >$ of Ms to be considered in prediction with $1 \leq g \leq n$. The prediction of the next segment of Ms is performed in three steps: (1) finding similar sequences, (2) extracting the consequent of similar sequences, and (3) predicting the next road segment.

3.2.4.1 Finding similar sequences

This step consists of using the *II* to discover all sequences whose prefix matches $LTRS_g(Ms)$ in any order and at any position. These sequences are called *the sequences similar to Ms*. To do so, the *II* initially performs the intersection of the bitsets of $LTRS_g(Ms)$. The resulting bitset indicates the set of similar mobility sequences to Ms. The *LT* structure is used afterward to identify and allow a direct access to these sequences in the *PT*.
Finding similar sequences is noise-sensitive. If some noise is introduced in Ms, *CPT* is only able to find similar sequences containing the same noise. More specifically, this situation worsens when dealing with GPS location

data prone to several noise and disturbances. To alleviate this challenge, *CPT* implements two strategies called respectively *Recursive Divider (RD)* and *PNR (Prediction with improved Noise Reduction)* [46].

- **Recursive Divider (RD)**. The RD strategy aims at gradually removing the noise from a given traveled mobility sequence TMs when searching for its similar sequences. This strategy tries to find similar sequences of each subset $Q \in TMs$ where $|Q| = k$ with $k = \{1, 2, \ldots, maxLevel\}$ and $maxlevel$ designates the maximum subset splitting size allowed for TMs. If a prediction cannot be made at level k, RD then explores the $k+1$ level if $k+1 < maxLevel$.

- **Prediction with improved Noise Reduction (PNR)**. To gain more flexibility with noisy data, PNR strategy could be also utilized during the prediction process. The PNR mainly relies on the hypothesis that noisy road segments appearing in training sequences are the ones with low frequency where the frequency of a road segment is defined as the number of sequences containing it. Consequently, PNR removes the segments having low frequency. To do so, PNR first deletes less frequent road segments from each sequence which are considered as noise according to a user-defined threshold called *Noise rate (NR)*. Then, a set of subsequences per level $k = \{1, 2, \ldots, |LTRS_g(Ms)| - 1\}$ is generated from $LTRS_g(Ms)$ of size $|LTRS_g(Ms)| - k$. Each subsequence is used afterward to find similar mobility sequences and update the *Count Table (CT)* accordingly. This process is repeated while the number updates of CT doesn't exceed a maximum number of updates NBR defined by the user.

3.2.4.2 Extracting the consequent of similar sequences

Once similar sequences to MS are retrieved, the consequents of each similar mobility sequence are retained. For a given mobility sequence Ms with its $LTRS(Ms)$, a mobility sequence S_{Ms} similar to Ms can be divided into three subsequences: the *context* subsequence comes first followed by the $LTRS(Ms)$ and in the third place comes the *consequent* subsequence. Formally, let $u =< rs_1, rs_2, \ldots, rs_m >$ be a mobility sequence similar to Ms. For a given g ($1 \leq g \leq m$), if we have $LTRS_g(Ms) =< rs_v, \ldots, rs_{v+g-1} >$, with $1 \leq v \leq m$, then we can write $S_{Ms} =< rs_1, rs_2, \ldots, rs_{v-1} > + LTRS_g(Ms) + < rs_{v+g}, \ldots, rs_m >$. Therefore, the context subsequence is $< rs_1, rs_2, \ldots, rs_{v-1} >$ and the consequent subsequence is $< rs_{v+g}, \ldots, rs_m >$ (see Figure 3.3).

3.2.4.3 Predicting the next road segment

Each consequent of a sequence similar to Ms is stored in a hash table called the $CountTable(CT)$. A CT is defined as a hash table with road segments as keys and a score as a corresponding value to each key. The score represents the frequency of a given road segment rs_j, which is defined as the number of times it appears in the consequent of mobility sequences similar to Ms. The

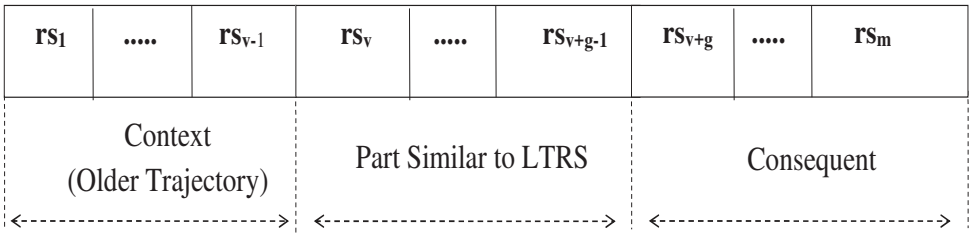


FIGURE 3.3: Outline of mobility sequence's consequent.

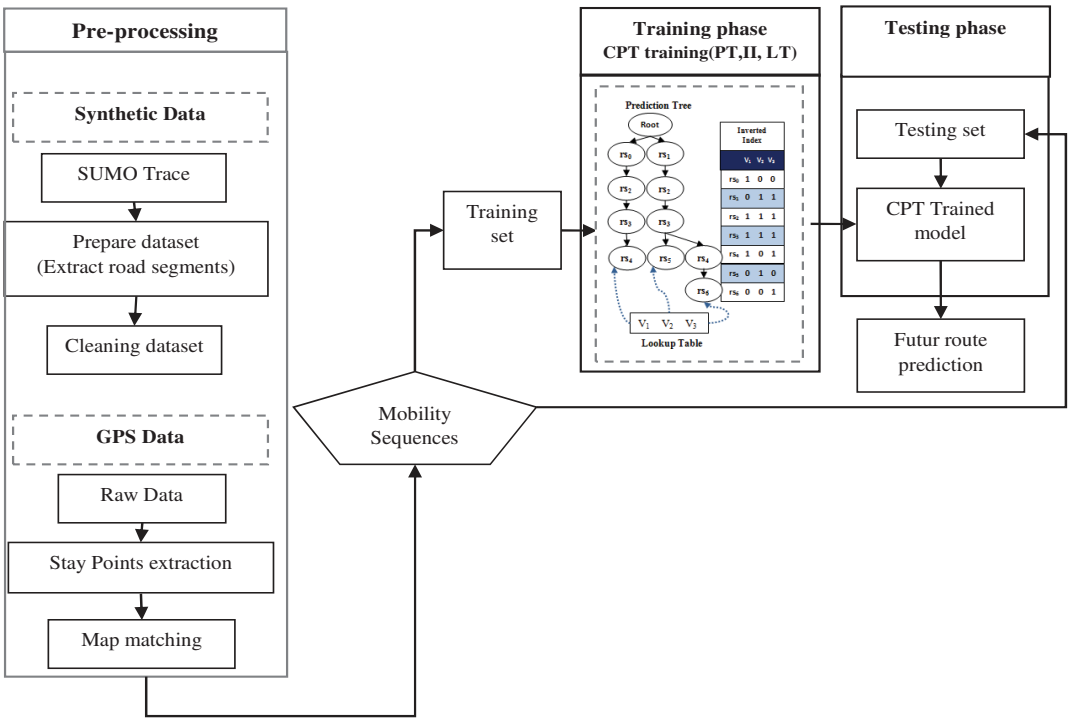


FIGURE 3.4: Architecture of NextRoute.

road segment with the highest score within the CT is the predicted next route of the vehicle v_i. In the case where the frequencies of two road segments are equal, the confidence is used. It is defined as the frequency of rs_j divided by the total number of sequences that contain rs_j (the cardinality of the bitset of rs_j in the II).

3.3 NextRoute model

The *NextRoute* model consists of two major modules: (1) the pre-processing or data preparation module, and the (2) route prediction module based on the CPT model presented in the previous section. The overall architecture of *NextRoute* is portrayed in Figure 3.4.

3.3.1 Pre-processing

To transform GPS trajectories into mobility sequences, *NextRoute* follows the pre-processing process presented in section 2.3.1.2. *NextRoute* applies the first and the second steps to extract trips from GPS trajectories. The resulting trips are then converted into mobility sequences by map-matching using a geometric based model [55]. The latter consists of finding, with a minimum number of computations, the "polylines" describing the curvature of routes traversed by users for a stream of coordinates recorded by a GPS device for every second. Note that this model is available as a cloud map-matching based API in [56].

3.3.2 Route prediction

The prediction module relies on the *CPT* sequence prediction model to forecast the next road segment of a vehicle based on its previous movements. To utilize this model, two main steps are performed: training and testing of prediction. Initially, the pre-processed data is split into two subsets. While the first part is used for training the three structures of CPT, the second part is used to test the predictions. The user of *NextRoute* needs to set the size of the *LTRS* to be considered (i.e. The g parameter).
The prediction process will consider the last $(g + 1)$ road segments of each testing mobility sequence. While the first g road segments are used to perform prediction based on the prediction constructed structures, the last road segment is used to check the prediction outcome (i.e. accuracy). First, the set of possible consequents are stored in the *CT* with their corresponding scores. The road segment having the highest score is then selected as the prediction result. Finally, the predicted segment is compared with the last road segment from the testing mobility sequence to determine if the prediction was a success or not.
To consider both global (collective) and personal (individual) movement behaviors of drivers. *NextRoute* implements two models.

3.3.2.1 Global model

The global model relies on the hypothesis that indicates that people always show collective mobility patterns. Human mobility has shown an observable degree of spatial regularity where people usually bear significant similarities in their movement while traveling from specific locations. For instance, persons who move from location *A* to *B* tend to take almost the same routes. This observation may be due to 1) the non-random-mobility of vehicles (e.g. vehicles move in specific lanes, routes, and directions), 2) the restrictions imposed by the road networks (e.g. only a limited number of routes from A to B are possible), or 3) the users' global preferences that tend to take the well-beaten roads, avoiding highways, especially when in newly visited places. To adapt *NextRoute* so it models the collective patterns, *NextRoute* is learned with

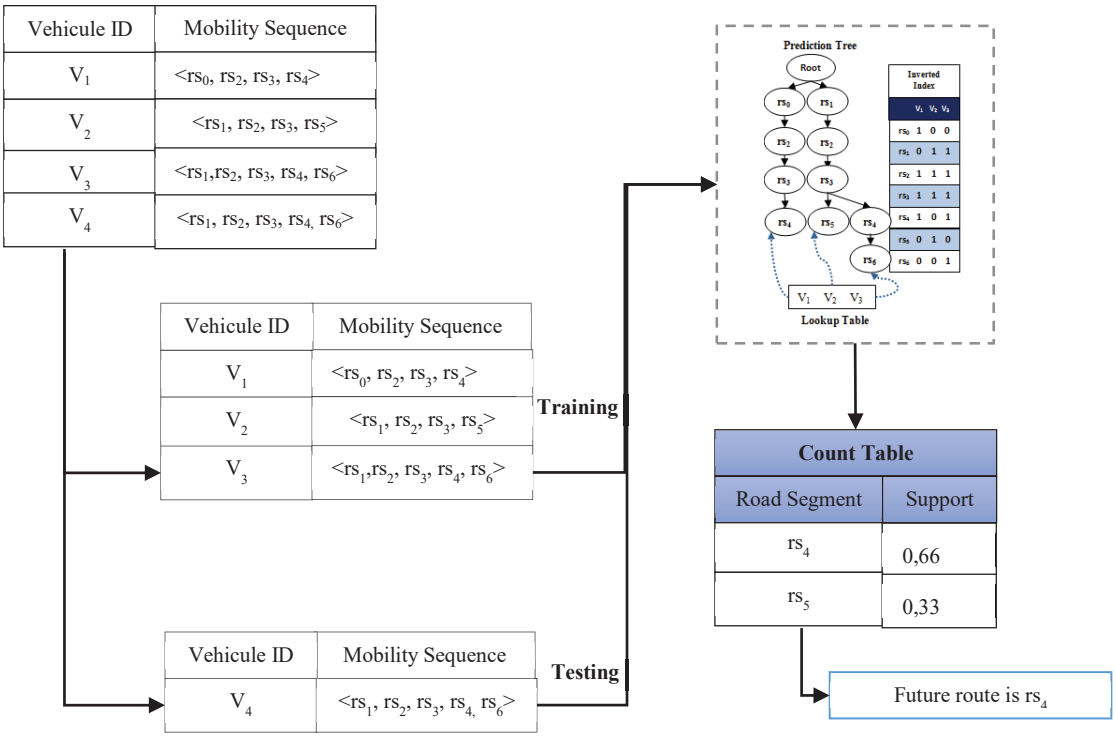


FIGURE 3.5: Illustration of global prediction with NextRoute.

mobility sequences of all drivers. Specifically, the global model is applied for newly seen drivers where their mobility patterns are indefinite.

For better understanding global model, consider the illustration depicted in Figure 3.5 that is based on the training data of Table 3.1. Let's consider a vehicle V_4 having its current mobility sequence ms=<rs_2, rs_3>. The mobility sequences that are similar to ms are those of the three vehicles V_1, V_2, V_3. Two road segments rs_4, rs_5 are thus stored in the CT with a frequency of 0.66 and 0.33, respectively. In the CT, the road segment having the highest frequency value is rs_4, and this road segment is thus chosen as prediction.

3.3.2.2 Personalized model

In addition to collective patterns, drivers also show their individual movement patterns. Each person has his driving habits, desires or preferences that influence his/her choice of a path to reach a destination. For instance, a user may always tend to take the same path he/she is familiar with even with the existence of a better route to the same destination. To adapt *NextRoute* so it models this individual behavior, *NextRoute* is trained on trajectories of that person rather than trajectories of all persons. Hence, the personalized model requires the existence of a mobility profile of the vehicle to perform prediction.

In fact, both models (global, personalized) complement each other. While the personalized model is useful for a user with an existing mobility profile, the global model seems more suitable for tourists and newly seen persons.

3.4 Experimental evaluation

To evaluate the performance of the proposed prediction model, a set of experiments were conducted. These experimentations were carried out on a test environment made of an Intel dual CPU 2.16GHz processor with 3GB of available RAM on an Ubuntu 12.04 operating system and 250GB of Hard Disk. In all the conducted experiments, the performance of *NextRoute* is compared with two state-of-the-art models namely: first-order Markov also known as PPM and used in [51, 60, 66, 67] and the LZ predictor [72] used for location prediction in [53]. These algorithms, in addition to CPT, were implemented in Java and their source codes can be downloaded as part of the SPMF library [91]. The next subsections describe the datasets used, the evaluation metrics, and the experimental results.

3.4.1 Datatsets & parameter setting

To better achieve realistic insight on people's mobility, it is desirable to utilize mobility data of vehicles used for personal purposes rather than those used for work such as taxicabs. However, and due to the privacy issues and governmental laws, the lack of realistic mobility data of vehicles used for personal purposes is noticeable. This fact has compelled us, as well as many researchers, to deal with several other types of datasets that hopefully well-represent mobility of vehicles.
In our experimentation, and to assess the performance of *NextRoute*, a real-life dataset, as well as a synthetic one representing two types of vehicle movement (simulation and taxicabs), were employed. Both datasets are prepared and then divided into training and testing sets. To compare the different models, 10-fold cross-validation was used, where each model was trained using 90% of the dataset and tested on 10% of the dataset for each fold.

Synthetic dataset. As a proxy for human mobility, a sumo-based scenario named LuST (Luxembourg SUMO Traffic) [92] is utilized. It provides a large-scale mobility simulation of Luxembourg urban areas within one day. LuST scenario accurately reproduces the vehicular mobility in Luxembourg by exploiting both realistic road network topology (i.e.OpenStreet Maps) and real traffic counting data from the Luxembourg Ministry of Transport. To generate mobility data from LuST, the process described in section 2.3.1.2 was applied to its scenario files.

GPS dataset Due to the lack of public and realistic vehicular data, a taxi cabs dataset of San Fransisco [93] is used. It contains GPS coordinates of approximately 500 taxis collected over 30 days in the SanFrancisco Bay Area. To reduce the computation time for our experiments, we have selected a subset of 10 randomly-chosen vehicles. To generate trips from GPS taxi trace, both T_{thre} and D_{thre} were set to 20 which indicates that a stay point is detected if the vehicle remains stationary or exhibits a low mobility for 20 minutes and

the distance between GPS coordinates of its trajectory within 20 minutes is 20 meters. Picking reasonable threshold values here is challenging since it is difficult to distinguish between transitory stops that are legitimate ends to a trip (e.g., stopping at desired destination) and normal traffic stoppage (e.g., stopping at a long traffic lights). To deal with this situation, we have used similar thresholds values used in other researches for trips retrieving (e.g. [94]). Besides, it is necessary to mention here that taxicabs are characterized by their high mobility. Such mobility leads to the generation of a low number of trips depending on the used stay points' thresholds. This explains the small number of trips generated from the SanFrancisco dataset with approximately one sequence per day with long mobility sequences that have an average length exceeding 330 road segments. To regard the sequential nature of mobility and allow predicting future locations, we have discarded from the generated trips those containing less than two road segments while preserving only those with a length greater than one road segment.

3.4.2 Evaluation metrics

To measure and compare the prediction performance of our proposal with other models, four well-used measures (success ratio, failure ratio, coverage, and overall accuracy) were employed. These measures are defined as follows:

- **Success ratio**. It determines the number of times that a correct prediction was accomplished. It estimates the number of trips where a successful prediction of the next route was made divided by the total number predictions made (i.e. by excluding the No Match cases).

$$\text{Success ratio} = \frac{\text{Number of successful predictions}}{\text{Number of predictions}}$$

- **Failure ratio**. It computes the number of incorrect predictions over the total number of predictions made.

$$\text{FailureRatio} = \frac{\text{Number of failed predictions}}{\text{Number of predictions}}$$

- **Coverage**. It estimates the number of times where a prediction has been made divided by the number of test trips. This measure designates the ability of the system to make a prediction (successful or failed). It is calculated from the *Number of No Match* prediction which indicates the number of sequences where no prediction was made, divided by the total number of test trips.

$$\text{Coverage} = 1 - \frac{\text{Number of No match}}{\text{Number of test trips}}$$

- **Overall Accuracy**. It is considered a global prediction indicator that measures how efficient the prediction was. It is computed as the number of successfully predicted routes divided by the total number of test trips. It is calculated as follow:

$$\text{Overall Accuracy} = \frac{\text{Number of successful predictions}}{\text{Number of test trips}}$$

3.4.3 Experimentation

In this section, we carry out a set of experiments to validate our proposal. In particular, we aim to examine:

1. The overall performance of *NextRoute* compared to other predictors.
2. How much the spatial regularity of human mobility affects prediction.

In what follows, a detailed description on the conducted experiments and the obtained results is provided.

3.4.3.1 Experiment 1 (Personalized and global prediction)

This experience aims at confirming the spatial regularity of human mobility while assessing the performance of personal and global mobility modeling.

A) Personalized model.
The personalized or individual predictor models the individual mobility patterns of each person using his past trajectories to forecast his future routes. Figure 3.6 and Figure 3.7 represent the attained performances of two randomly selected drivers denoted as *Driver1*, *Driver2*, respectively.

Results.
From a general point of view, our evaluation shows that closer performance for both *Driver1* and *Driver2* was attained with a successful prediction ratio between 70% and 98%, a failure prediction less than 30% and high coverage of 80 to 93%.
Likewise, these results indicate that prediction accuracy is improved by increasing the number of mobility sequences considered leading to about 87.5% for personalized prediction. This could be ascribed to the fact that more repeated trips, including personalized mobility behavior of a driver, are involved in training data. Thus, the more routine mobility is captured, the more accurate prediction is achieved which consequently improves the prediction outcome.
It is worth noticing that intra-sequences similarity could be decreased in some situations when the driver's mobility differs from a trip to another depending on his destination. This statement is well perceived with taxicabs data, we are using, where drivers' destinations are client-dependent.
Besides, drivers' destinations could be also affected by temporal factors such

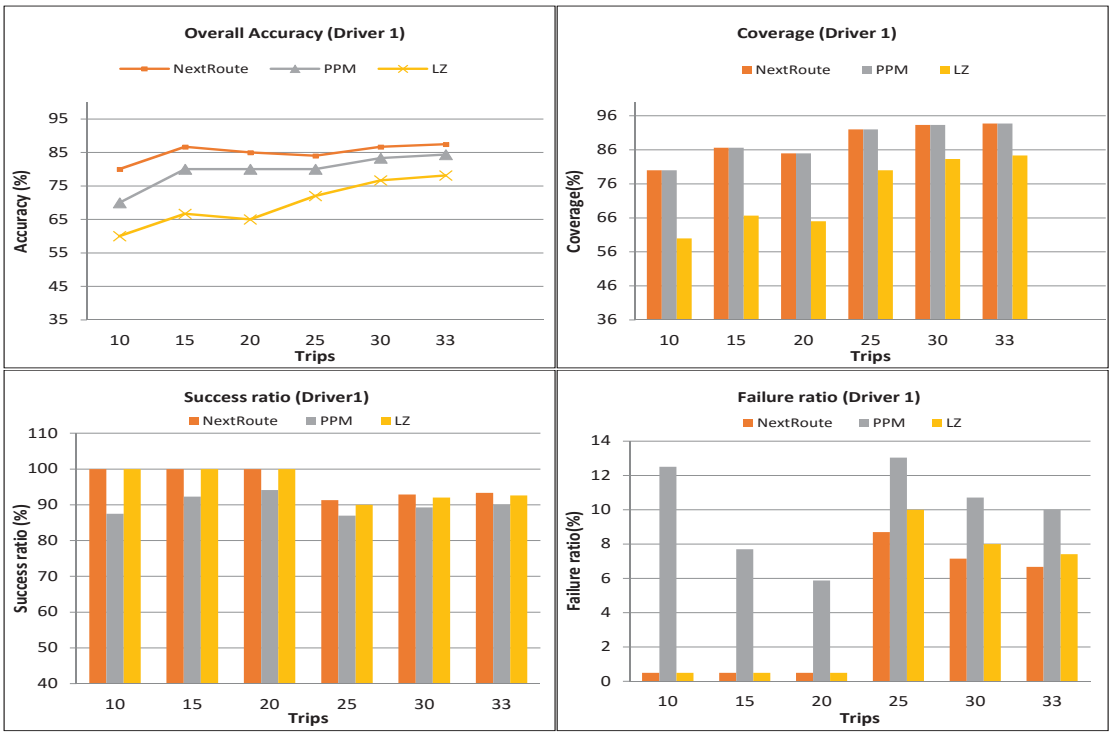


FIGURE 3.6: Prediction performance of Driver1.

as day-of-week and week-of-month where a driver could choose another route relying on the time of day (e.g. to avoid traffic jams in some hot area in a specific time of day). Consequently, we believe that the regularity of the driver's behavior should be well studied by incorporating other factors to guarantee more accurate predictions.

B) Global prediction model. This model discovers global and collective mobility patterns in trajectories of all drivers. The obtained results are portrayed in Figure 3.8.

Results. From the results described above, we found out that the global mobility behavior of persons is also confirmed. This statement is well-perceived with taxicabs covering pre-defined areas where the traveled trips are to some definite places such as taxicabs bringing customers from/to airports, hotels, shopping malls, or touristic locations. Therefore, a high degree of similarity among mobility sequences could be captured which boosts the prediction accuracy.
Besides, in contrast to the personal model, global prediction performance (successful ratio, coverage and overall accuracy) decreases with increasing number of mobility sequences. This may be due to many reasons. First, the intra-trips similarities decrease as heterogeneous trips belonging to different drivers are encompassed in data. Second, these trips also characterize different drivers' behavior with different daily habits and destinations on weekdays and weekends, which expand the differences among trips. Again, and

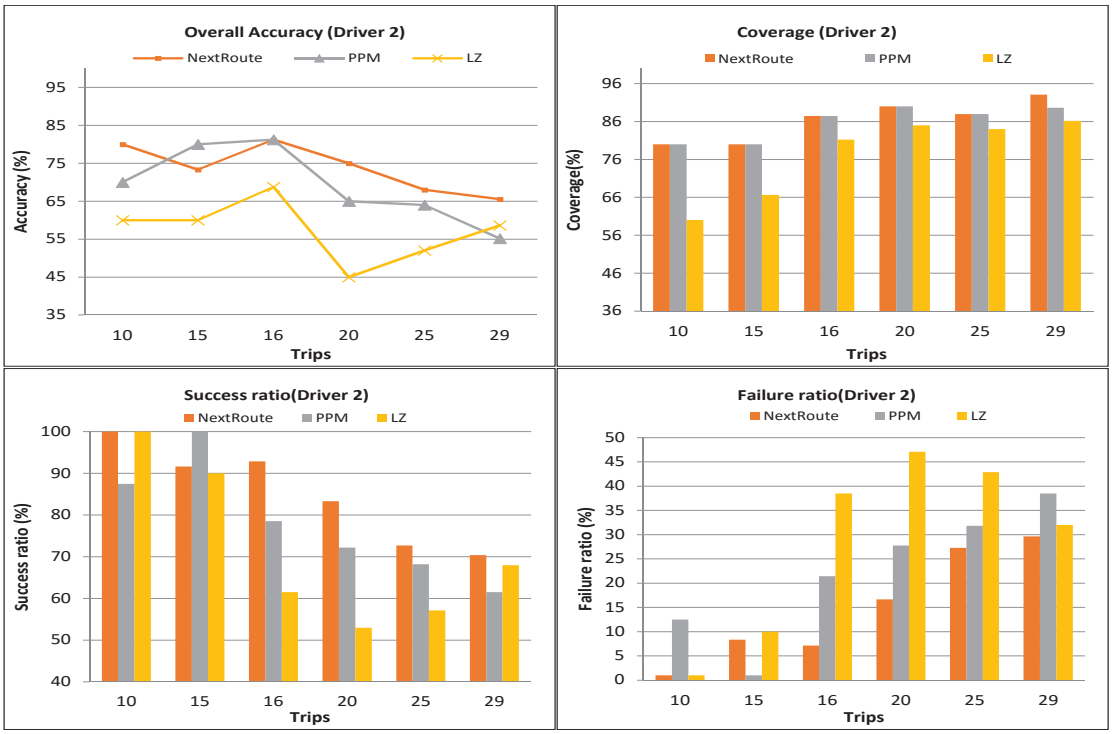


FIGURE 3.7: Prediction performance of Driver2.

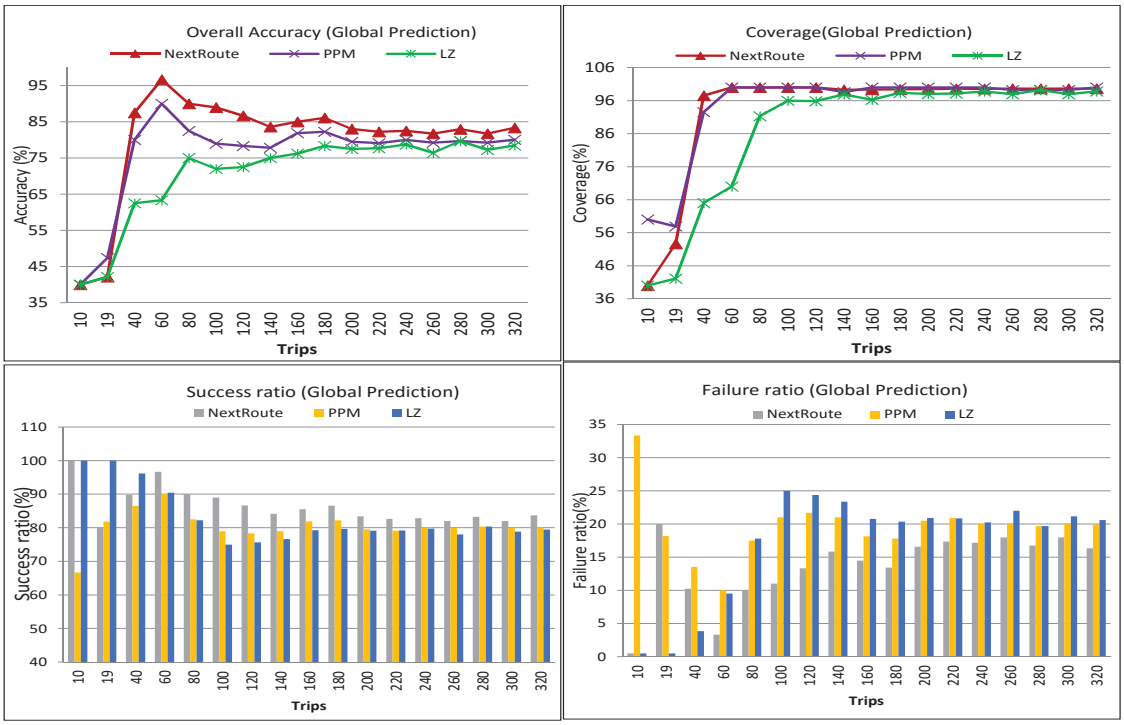


FIGURE 3.8: Performance results of global model.

to overcome this situation, it is recommended to consider the temporal factors in addition to spatial ones in prediction.
Moreover, as mentioned earlier, decreased predictor quality could be due to the use of taxi traces where the vehicle's final destination change depending on the customer's desire. Furthermore, the prediction could be affected by the selection process of the best route among candidate routes in CT which is only frequency-dependent. Hence, we suggest associating a weight to each candidate route that considers, in addition to the frequency, temporal factors and other selection criterions such as less congested routes, recently traveled trajectories, etc.

C) Performances comparing other models.

It is also found, from the above results, that prediction accuracy is considerably influenced by the number of vehicles. The models become more accurate as more vehicles are considered. This can be ascribed to the fact that prediction models usually perform better when more training data is available. Besides, the results indicate that *NextRoute* outperforms the other models in terms of accuracy, reaching an accuracy of more than 94%. A reason why *NextRoute* performs so well is that it considers several previous segments unlike Markov models such as PPM that assume the Markovian hypothesis. Besides, another reason why *NextRoute* performs well is that it is noise-tolerant when it matches a trip with similar trips, as it can consider that some road segments are missing or ordered differently on different trips but they can still be similar to each other. This is different from other models that require a very strict ordering between road segments in trips, and cannot predict for a trip if its last road segments have not been previously seen in exactly the same order.
Considering the prediction coverage, the best coverage was obtained by *NextRoute* and PPM followed by LZ. The coverage is an important metric as it measures the ability of the prediction model to make a prediction. It is best if a predictor can make a prediction even if it is a failure. This is because additional work could then be done to increase the accuracy of the model to fit the desired output or parameters to turn a failed prediction into a successful one. Another reason why prediction coverage is important relies on the fact that prediction coverage affects the usability of real-world prediction-based applications. In such applications, user satisfaction is the main focus. The user may be more satisfied if he/she gets a prediction output (e.g. message) from the application even if it is erroneous rather than getting a system's output indicating the incapability of the system to make prediction.
In terms of failure and success ratios, results also exhibit that *NextRoute* has the lowest failure ratio with the highest success ratio, in most cases. This is considered an excellent result that shows that additional efforts could be done in future work to further reduce the failure ratio and increase the success ratio.

3.4.3.2 Experience 2 (Scalability)

In this experiment, we study the scalability of *NextRoute* against PPM and LZ predictors by incrementing the number of mobility sequences (trips). Scalability is an important factor for most prediction systems as it determines the ability to scale a prediction model, which can directly or indirectly decrease its performance and thus affect its usability. To better represent the scalability, LuST dataset characterized by its high number of trips is used. Note that the LuST dataset encompasses only a single trip for each driver. Hence, only the global behavior of drivers is to be considered in this experience.
As depicted in Figure 3.9, *NextRoute* has achieved a success ratio up to 80% in all cases. In terms of failure ratio, *NextRoute* has the lowest failure ratio comparing to other state-of-the-art predictors. Failed predictions, obtained in this experiment, could be ascribed to the data used which is limited to one day only. Hence, the spatial regularity of human mobility is not well captured and routine drivers' behavior is missing.

Figure 3.9 also illustrates the accuracy obtained with LuST dataset when the number of vehicles is increased. Initially, the results exhibit that prediction accuracy is affected by the number of vehicles considered. Overall, for a number of trips less than 10.000, *NextRoute* becomes more accurate as more vehicles and thus more trips are considered. As mentioned earlier, this could be referred to the fact that predictions become more accurate as more data is available. It is also found that *NextRoute* has achieved a prediction accuracy of more than 80%. *NextRoute* performs so well for its lossless property that allows considering several previous segments while forecasting future routes. The overall accuracy has been then decreased for a number of trips greater than 10.000 where newer drivers' mobility is involved. Thus, trips' similarities decreased as each trip in LuST belongs to a single driver. In terms of prediction coverage, the results indicate that *NextRoute* coverage has been increased by incrementing the number of trips until 10.000 trips and has attained full coverage (100%) with a higher number of trips.

3.4.3.3 Experience 3 (Time and space complexity)

In this experiment, we empirically compare the influence of increasing the number of trips on execution time and memory consumption of *NextRoute* against other models. The execution times (training time and testing time) were in seconds whereas the size of each model is computed in terms of the number of nodes. The LuST dataset is used in this experiment. Training time, testing time, and model size of each predictor are illustrated in Figures 3.10, 3.11 and 3.12 respectively.

As expected, by increasing the number of trips, time and space requirements increase. This is especially well noticed in the case for the training time and model size for a number of trips greater than 1.000 trips. Considering training time and model size, the best performance was obtained by *NextRoute* followed by PPM and then LZ. Thus, *NextRoute* quickly creates

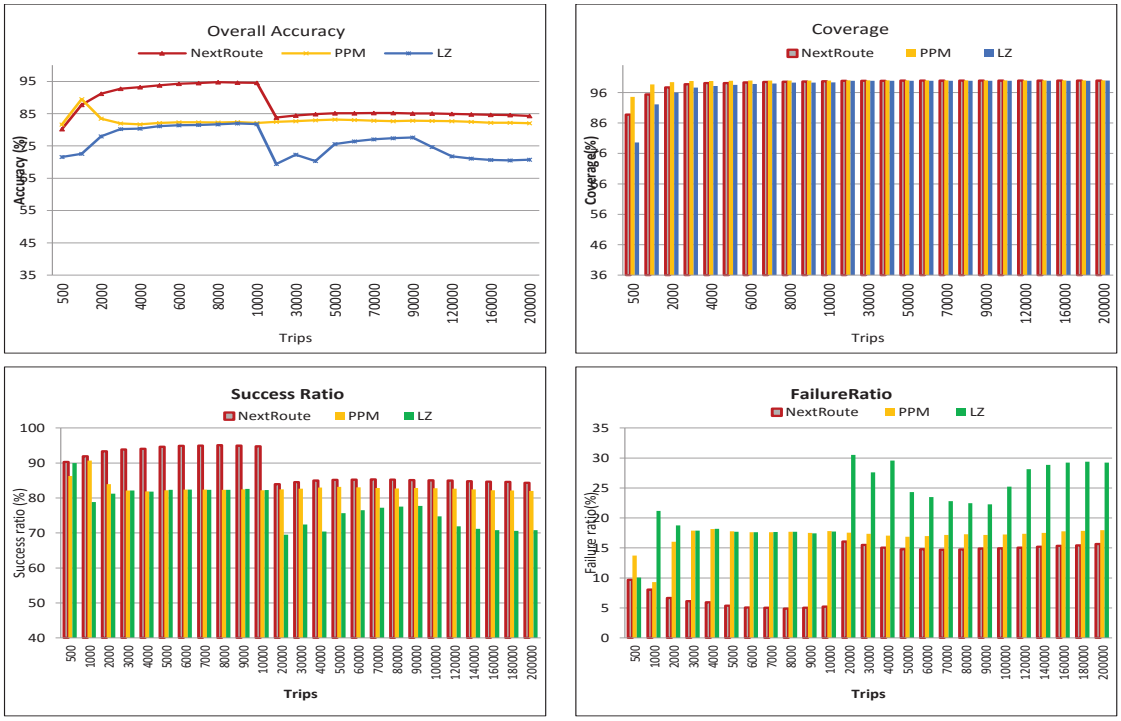


FIGURE 3.9: Perfomance results of NextRoute with LuST dataset.

and updates its prediction structures. The training time is an important metric as it assesses how fast the model can be updated by inserting new trips.

Considering the testing time, results show that *NextRoute* requires more time to perform prediction. This may be referred to the matching process using the PT, which finds branches that correspond to the current trajectory traversed by the user (driver). Nevertheless, this is considered tolerable because *NextRoute* provides more accurate predictions, and overall the prediction time remains very short in all cases (less than one second when considering more than 10.000 trips). Besides, the long prediction time is a well-known problem of tree-based models. Hence, it is an interesting topic for future work to consider how to speed up the prediction process for such models.

In terms of model size, it is found that *NextRoute* has constructed a CPT prediction model that comprises fewer nodes than PPM and LZ. This could be ascribed to its compact presentation and compression mechanisms where trips could share common paths and as more trips are added to PT, many of them could overlap which allows reusing exiting nodes and paths.

3.5 Conclusion

In this chapter, we have presented *NextRoute* as a simple and efficient route prediction model to forecast the future routes of vehicles. Based on a novel,

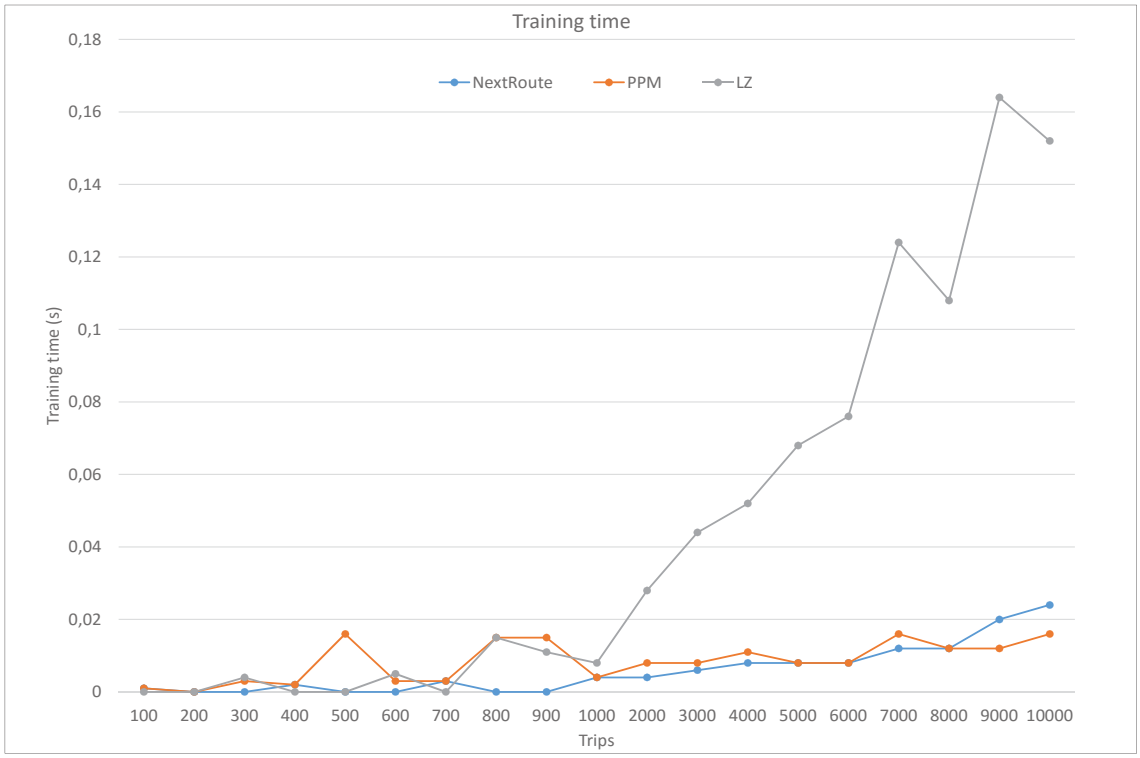


FIGURE 3.10: Training time with LuST dataset.

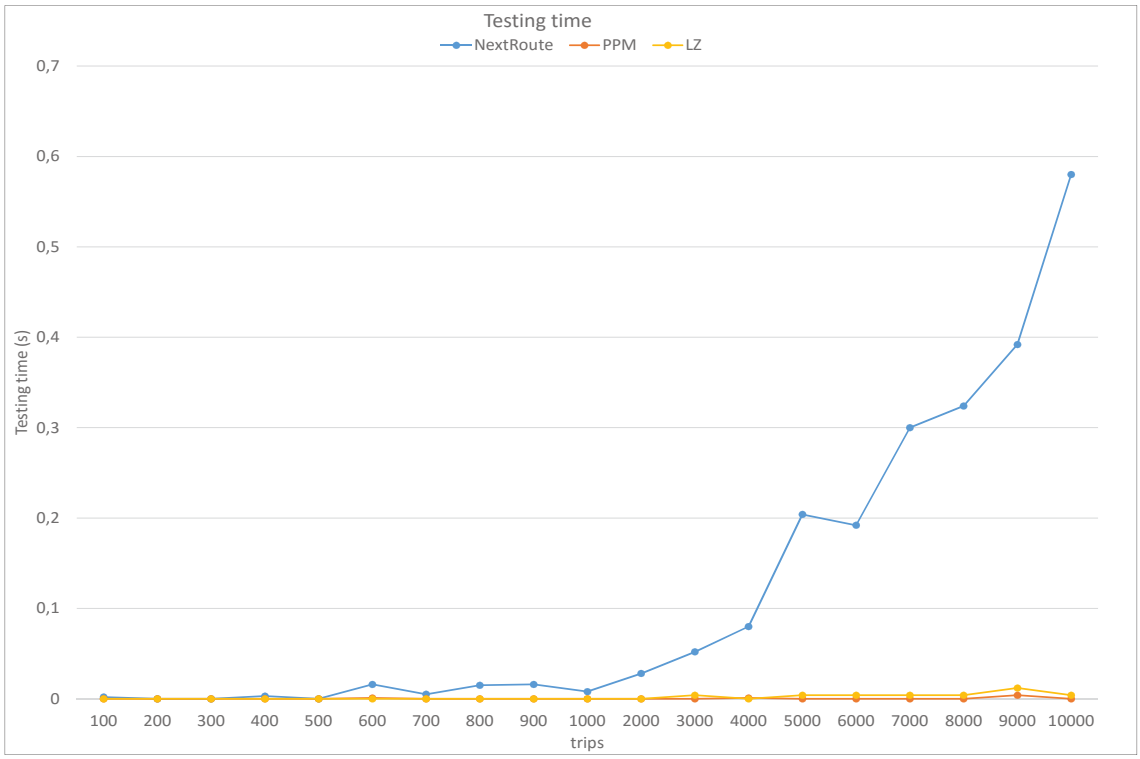


FIGURE 3.11: Testing time with LuST dataset.

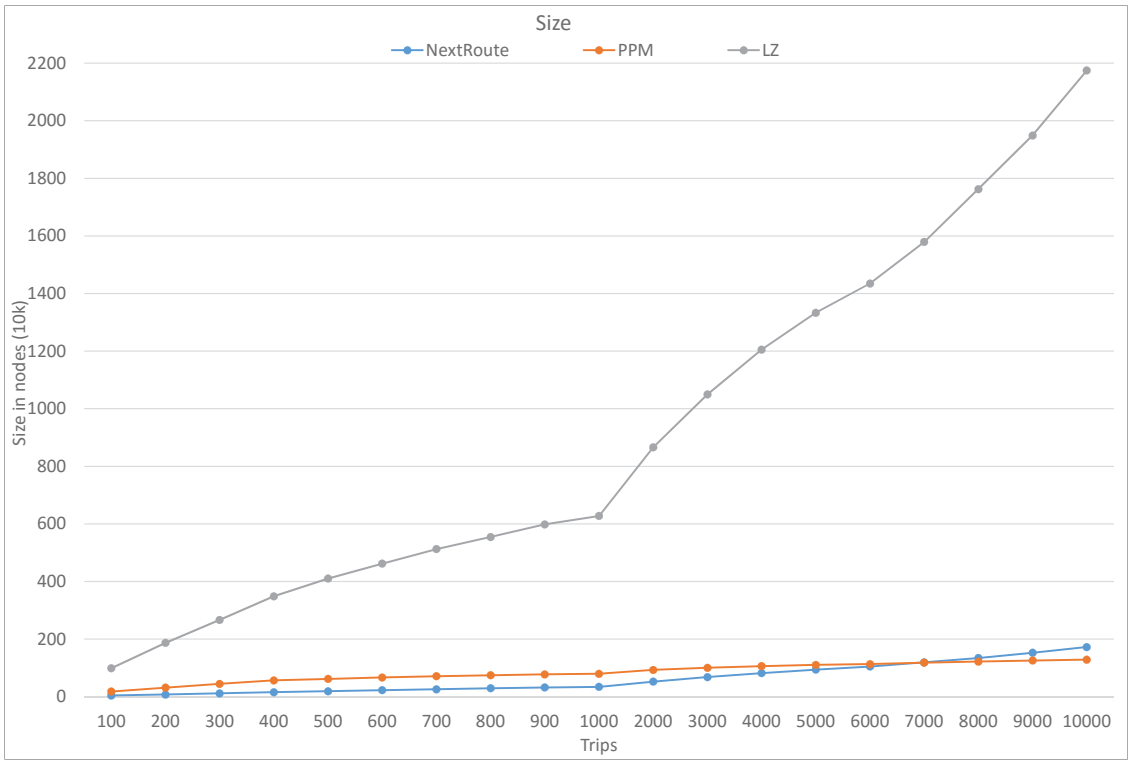


FIGURE 3.12: Model size with LuST dataset.

lossless and accurate sequence predictor, *NextRoute* implements two models; Global (GM) and personalized(PM) models. These models are employed to exploit global and personal driver's mobility behavior in prediction. The GM discovers global behaviors with all available mobility data of all users whereas PM models the individual patterns of each driver relying on his past mobility data. *NextRoute* was evaluated using realistic and synthetic vehicular datasets. It has shown good performances in all datasets, a relatively low failure ratio, and high prediction coverage, outperforming other state-of-the-art models. Relying on the performed evaluation, we can draw interesting conclusions.

- First of all, our evaluation shows how *NextRoute* attains satisfactory prediction performance in most cases with an overall accuracy that is greater than 70% with the highest prediction coverage, the lowest failure ratio and reasonable execution times, and model size.

- Secondly, the execution time required to predict, using the CPT model, may affect to a high degree the performance of our proposal. Therefore, an additional effort should be done to alleviate this impact whereas at the same time a deep study of the domain of application should be done to select the most suitable configuration before the real implementation of the framework.

- Moreover, to better achieve realistic insight on people's mobility, it is desirable to utilize mobility data of vehicles used for personal purposes rather than those used for work such as taxicabs.

- Last and not least, even though *NextRoute* is initially designed for route prediction tasks, it could be employed in several real-world mobility applications with location awareness and sequential behavior consideration such as successive points of interest recommendation in location-based social networks, handoff management in cellular networks, animal migration research.

Chapter 4

MyRoute: a Graph-dependency Based Model for Real-time Route Prediction

4.1 Introduction

Route prediction is an important service having numerous applications in mobile computing and pervasive systems. Although several route prediction proposals have been made and have shown to perform well, an important drawback is their noise sensitivity toward the mobility patterns that they can learn. The smallest deviation in trajectory data affects the prediction result, and thus the prediction accuracy. Additionally, many proposals were designed regardless of any constraints on computational and memory limitations of mobile devices which makes the real implementation and deployment of these models impractical as on-line applications.
In the light of the above challenges, we propose, in this chapter, a real-time graph-based model for route prediction called *MyRoute*, which adopts the Dependency Graph (DG) predictor , previously used for web prefetching [95]. The proposed approach first builds a mobility prediction graph and then uses it to predict the next route of a driver by attempting to match its current trajectory with graph paths. However, graph matching is difficult due to the presence of noise in data. To address this issue, the proposed approach creates graph arcs not only between consecutive road segments (nodes) in a path but also with the following road segments appearing within a user-defined *lookahead* window.
The remainder of this chapter provides a detailed description of *MyRoute*. It starts with the definition of the mobility graph and then describes the architecture of *MyRoute* framework. To model both global and personal mobility behaviors, two prediction schemes are proposed in Section 4.3.3.3. The set of experiments performed to assess the quality of the proposed predictor is then presented in Section 4.4. Finally, Section 4.5 concludes the chapter with a discussion and analysis of the obtained results.

4.2 Graph dependency-based Predictor

MyRoute implements a dependency graph-based predictor that attempts to forecast the future location of a vehicle. Driven by historical mobility data collected from vehicles, the predictor constructs and gradually updates a probabilistic graph called a *Mobility Graph*(MG). The choice of such a model for our predictor relies on the idea that graphs can perfectly represent the structure of road networks, and therefore accurately model vehicle movements. Moreover, MG is also based on the idea that vehicle mobility is order dependent as a vehicle passes through a sequence of route segments to reach some locations of interest, by following a specific order and traversing road segments in specific directions. Thus, the graph modeling of our predictor could perfectly match this underlying idea by representing order dependencies as arcs between graph nodes. In what follows, we describe our graph predictor staring from the basic model that models mobility sequences, and the extended model that incorporates additional transitions in the graph to handle noisy location data.

4.2.1 Mobility graph: the basic model

A mobility graph *MG* model mainly aims to create a probabilistic graph that stores mobility sequences traversed by drivers. In a *MG*, each road is represented as a graph node. The graph nodes are connected with arcs (edges) that embed the sequential visiting order among locations from the mobility sequences also called a *dependency order*. In *MG*, two graph nodes A and B are connected through an arc that defines the dependency order between A and B if and only if at some point in time B was traversed after A in a mobility sequence. For each established dependency (arc), a weight value is associated to measure the number of times that location B had been traversed after A.

4.2.2 The extended model

As previously mentioned (in Chapter 2), GPS location data are prone to several disturbances, noise, and inaccuracies that significantly affect the prediction outcome. More specifically, noisy or uncertain locations principally affect the map-matching process by generating fake snapped roads which consequently creates illusory graph transitions and confuses prediction.
To alleviate this issue, and further enhance the prediction precision, the basic graph model is extended. In the extended model, the graph construction is essentially the same, except that MG created arcs not only between consecutive road segments (nodes) in a mobility sequence but also to several upcoming road segments appearing within a user-defined *lookahead* window. By doing this, susceptible erroneous roads are skipped and more flexibility to deal with missing data is achieved which consequently reduces the effect of

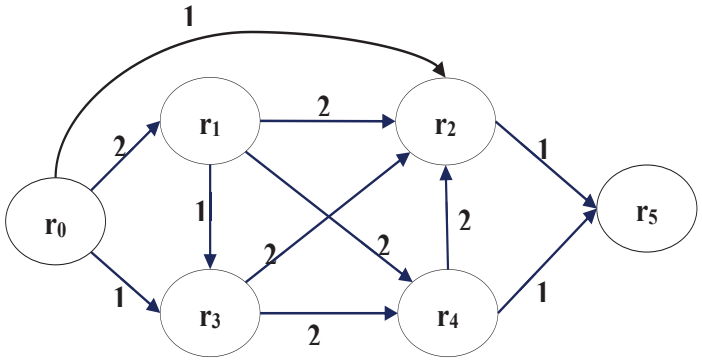


FIGURE 4.1: A sample of mobility graph with *lookahead*=2.

noise in GPS readings, and provides probabilistic path observation for prediction purposes. The extended *MG* model with noise tolerance strategy is formally defined as follows.

- **Definition (mobility graph)**. A mobility graph MG is defined as $MG = (R, A)$ where R is a set of all road segments (vertices or nodes) and $A \in R \times R$ is a set of all directional arcs (here also called *dependencies*) in the graph representing movements on the road network. An arc $a(r_x r_y) \in A$ connects two road segments r_x and r_y in MG. The road from which the movement a starts is called *the source of* a, and is denoted as $Source(a) = r_x$, whereas the other node r_y is called *the destination of* a, and is denoted as $Dest(a) = r_y$. An arc $a(r_x r_y)$ is created in MG if and only if r_y appears within w movements after r_x in a mobility sequence, where w is the *lookahead* window size. Moreover, a weight value $(w(a))$ is associated to each arc a in MG, which indicates the number of times that $Dest(a)$ was traversed by drivers after $Source(a)$.

Given the set of mobility sequences and with respect of *lookahead* parameter, the mobility prediction graph is built by gradually inserting mobility sequences one by one, where each mobility sequence is represented by a path comprising the set of roads that it contains. In the case where sequences share common roads, the weights of the shared arcs are incremented instead of creating new arcs. Consequently, significant space reduction can be achieved. For instance, Figure 4.1, portrays the mobility graph built from mobility data depicted in Table 3.1. For the sake of illustrating the benefit of noise tolerant strategy, let s=<r_1,r_2,r_7,r_3> be a mobility sequence and r_7 be a noisy location (road). Figure 4.1 shows that *MG* also incorporates the connections (arcs) from r_2 to r_7 and from r_2 to r_3 for *lookahead* =2 even though they do not appear in the original data. By doing this, an additional movement to the following roads of a susceptible erroneous location is created which boosts the frequency of transitions that skip the noisy location during the prediction step and thus promotes the ignorance of this location.

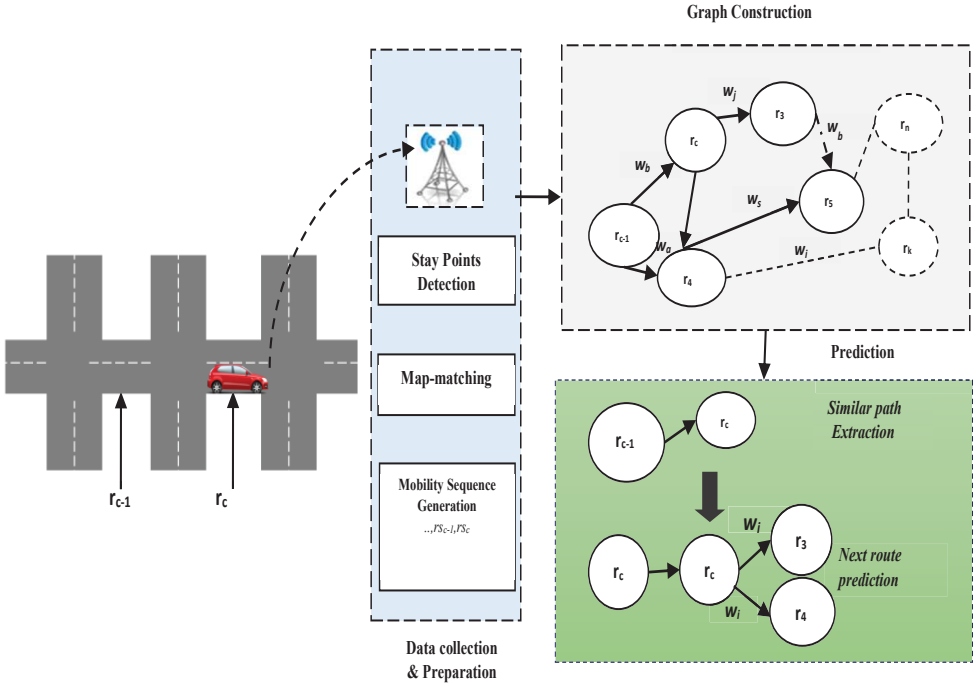


FIGURE 4.2: The overall architecture of *MyRoute*.

4.3 System architecture

In this section, we will provided a description of our prediction model, called *MyRoute*, which adopts the mobility graph predictor defined above. *MyRoute* aims at forecasting future roads, in collaboration with servers (i.e. service providers), that are likely to be within the reach of a driver given his current location.

As shown in Figure 4.2, *MyRoute* consists of three major steps, namely: data preparation, graph construction and prediction. The prediction phase, in its turn, is subdivided into two sub-steps: graph matching and next road extraction.

While the graph construction phase is run offline at the server site (e.g. authorities or service providers), the prediction phase, is performed in real-time at the client site (i.e. driver). The driver sends its current location to the server which map-matches this location into a symbolic location, performs prediction and then conveys to the driver the predicted location that he is susceptible to visit in the future.

This partitioning of work between the server and the client is natural. The server exploits mobility patterns collected from vehicles to build the mobility graph that is used afterward to perform intelligent predictions. On the other hand, the client-side (driver) transmits his current location and receives the forecasted with a minimum processing cost.

4.3.1 Data preparation

In this phase, mobility data (i.e. GPS coordinates) of each vehicle that include its current location with any other previously visited locations are periodically collected and then sent to the server site. During data collection, mobility data is split into trips by identifying the set of stay points. A stay point is a geographic area expressed as a set of consecutive GPS records where the distance from the first and last GPS record exceeds a distance threshold D^{thre} and the driver spent more time than a threshold T^{thre} (see definition in Section 2.3.1.2). The resulting trips are then transformed into mobility sequences by map-matching GPS trajectories using a cloud map-matching based API [55](See Section 2.3.1.2 for more details).
By receiving movement data from vehicles, the server initially checks whether a user profile for that vehicle exists in the movement database. If so, the data are appended as entries into the corresponding profile. Otherwise, a new profile for that vehicle is created incorporating its mobility sequences.

4.3.2 Graph construction

Once mobility sequences are obtained, the server initially builds and then extends the mobility graph by incrementally inserting mobility sequences. As described earlier, road segments as represented as graph nodes, whereas vehicle movements between pairs of road segments within the *lookahead* window size are represented by arcs. The weight of each arc is initially set to 1.
By receiving new mobility sequences from vehicles, the graph building process initially scans the paths of the graph to find out whether it fully or partially includes the mobility patterns (subsequences) of these sequence. In this situation, only the weights of the arcs corresponding to the existed transitions are incremented without the need to create new arcs. Otherwise, new paths (nodes and transitions) corresponding to newly seen patterns in these sequences are created. Note that, in some situations where movement sequences are highly heterogeneous and do not overlap (have no common road segments and/or transitions), the mobility graph becomes disconnected. In our model, we assume that MG is connected, comprising at least two nodes, and has no isolated nodes. These assumptions, hold in real-life, representing the mobility of vehicles in active urban areas.

4.3.3 Prediction module

To forecast the next route segment which will be visited by a given driver D, his current trajectory denoted as CT (Current Trajectory), is required. CT includes at least the current road segment where D is located with any other traversed roads that belong to the same trip, if available.

Formally, let RS=$\{r_1, r_2, \ldots, r_n\}$ be the set of all road segments in a road network RN. CT=$p_j, p_{j+1}, \ldots, p_c$ is a sequence of road segments traversed by D where p_c is his current road segment. Having the trajectory CT, the prediction is performed as follow:

4.3.3.1 Graph matching

To find the best match of CT to the set of previously driven trips, graph matching consists of measuring the likelihood of graph paths with CT.
More formally, graph matching attempts to search a path SP=r_i,r_{i+1},. . .,r_m in MG that matches with CT where $r_i \in RS$. SP is said to match CT if and only if every road segment in SP appears in the same order in CT, that is $\forall i, r_i = p_i$ and $m = c$.
It is worth mentioning that graph matching is a noise-sensitive process. Finding the path that exactly matches CT is challenging with the existence of erroneous positions. Using MG, built according to the *lookahead* window, more flexibility to handle noisy data could be attained. MG allows creating arcs not only between consecutive road segments (which may be noisy locations) but also to the upcoming road segments within the *lookahead* window. In other words, if the first road segment Ne that comes after a given node Nx in a mobility sequence $s = Nx, Ne, Ny$ is considered as noise, another arc that skips Ne and go directly to next road segment Ny will be created, given that $w \geq 2$.

This noise tolerance strategy allows *MyRoute*, unlike other Markov-based predictors such as PPM, to forecast the future route of CT that have not been previously seen in mobility sequences which consequently increase the prediction coverage of *MyRoute*.

Another interesting property that is implicitly implemented in MG refers to the fact that noisy data are usually characterized by their low frequencies. Therefore, even though an arc a_k is created between Nx and the other node suspected as noise Ne, the weight (frequency) of a_k would be the lowest among the set of arcs emanating from Nx ($dest(a_k)$), and thus (a_k) is automatically ignored thereafter with the existence of other arcs with higher weights.

4.3.3.2 Future road prediction

The second step is to retrieve the next road segment, denoted as N_r, that a driver will visit following SP. N_r is predicted as the destination of the arc having the highest weight emanating from the last road segment in SP.
More formally, let E={a_1,a_2,. . .,a_n} be the set of outgoing arcs for the last node of SP(r_m). Then, N_r is defined as: $N_r = Dest(a_k)$ such that $w(a_k) \geq w(a)$ for $a \in E$ and $\forall a_k \in E$ and $source(a_k)$=r_m.

For example, consider that the current trajectory of a driver is CT={r_0,r_3,r_4}. Based on the mobility graph of Figure 4.1, two candidate arcs are considered, which are $a_1(r_4\ r_2)$ and $a_2(r_4\ r_5)$. These arcs have weights of 2 and 1, respectively. Therefore, the next route segment is predicted to be $N_r = r_2$ (the destination of a_1).
In the case where candidates segments have equal weights, other selection criteria could be employed such as retaining the road with the highest frequency in mobility data. Again, as the prediction in MG is frequency-dependent,

arcs emanating to the susceptible noisy location are discarded which complies with the hypothesis that noisy data are characterized by their low frequency.
Up to this point, we have only described how *MyRoute* performs short-term prediction using *MG*. However, long term prediction that attempts to forecast several future routes is also easily achievable by incrementally finding the path having the highest weight among the set of outgoing paths strating from the predicted next route N_r.

4.3.3.3 Graph based models

Having presented the proposed prediction scheme, this subsection explains how the mobility graph is adapted to consider global (collective) and personal (individual) movement behaviors of drivers.

1. **Global MG (GMG).** The movements of persons often show strong collective characteristics. People usually tend to follow the crowd, and the majority of people prefer to take the well-beaten and typical paths, especially when in an unfamiliar area. As a result, we have decided to build the Global mobility graph (GMG). This model utilizes the trajectories of all drivers to discover collective patterns; thus, the prediction graph is trained from mobility data of all drivers. GMG is mainly utilized to perform predictions for a driver when no prior knowledge about his mobility patterns is available such as the case of drivers visiting the area for the first time.

2. **Personal model (PMG).** This model consists of modeling upon individual mobility profiles for each driver comprising his previous trajectories and then constructing the mobility graph relying on these profiles. As PMG only concerns one single driver at once; the prediction graph is trained with his mobility sequences rather than the data of all drivers. PMG constitutes a sub-graph of the GMG model. Thus, a significant space reduction, comparing to GMG, is achieved. This finding is empirically confirmed.

Overall, to forecast the next location of a given driver, the GMG model is used unless the driver's mobility matches one of the pre-existing mobility profiles. In this case, PMG is used. In other words, PMG is used if the driver's mobility history exists whereas GMG is devoted to newly seen drivers.

4.4 Experimental evaluation

4.4.1 Experimental settings

To evaluate *MyRoute*, we compared its performance with PPM and LZ algorithms (more details about these methods can be found in Section 2.4.1). We set *lookahead* window to 2 as it has empirically shown to give the best result after many tests whereas PPM and LZ78 do not have any parameters to

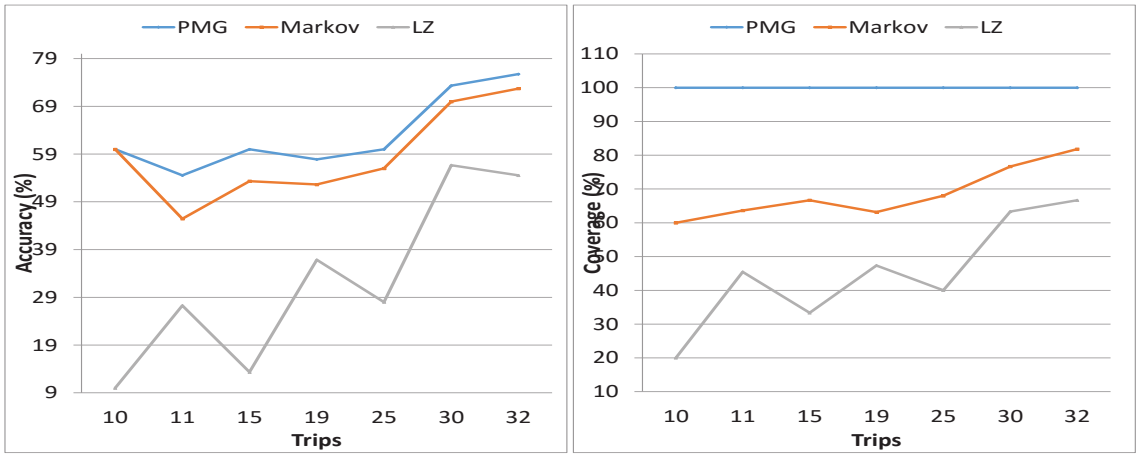


FIGURE 4.3: Performance evaluation of Driver 1.

set. To allow reproducing the experiments, the source code of the algorithms used for comparison are provided with SPMF framework [96]. Experiments have been performed on a computer equipped with 4th generation Intel i5 CPU, 8GB of RAM and 1TB of Hard Disk. Experiments were carried on the SanFransisco cabs, and LuST [92] datasets which are the same as the sets in Section 3.4.1. To generate mobility sequences from SanFransisco cabs GPS dataset, both T^{thre} and D^{thre} thresholds were set to 20 which are the same as the setting in Section 3.4.1. Again, among the generated trips, only those containing at least two road segments were considered.

4.4.1.1 Evaluation metrics

To measure the performance of *MyRoute*, the performance metrics accuracy and coverage are computed in each test. The formal definition of each metrics is presented in Section 3.4.2.

4.4.2 Experimentation

In this set of experiments, we aim to assess the performance of our graph-based predictor *MyRoute* and compare it with PPM and LZ predictors.

4.4.2.1 Experiment 1

We aim in this experiment to compare the performance of GMG and PMG models against PPM and LZ using the taxicabs dataset. For *MyRoute* PMG model, mobility profiles of two randomly chosen taxi drivers denoted as *Driver1* and *Driver2* were used. For GMG, a set of mobility sequences of 10 randomly selected taxicabs were exploited to evaluate the predictors (MyRoute(GMG), PPM, LZ).

Prediction of the two models are illustrated in Figures 4.3, Figure 4.4 for PMG and Figure 4.5 for GMG.

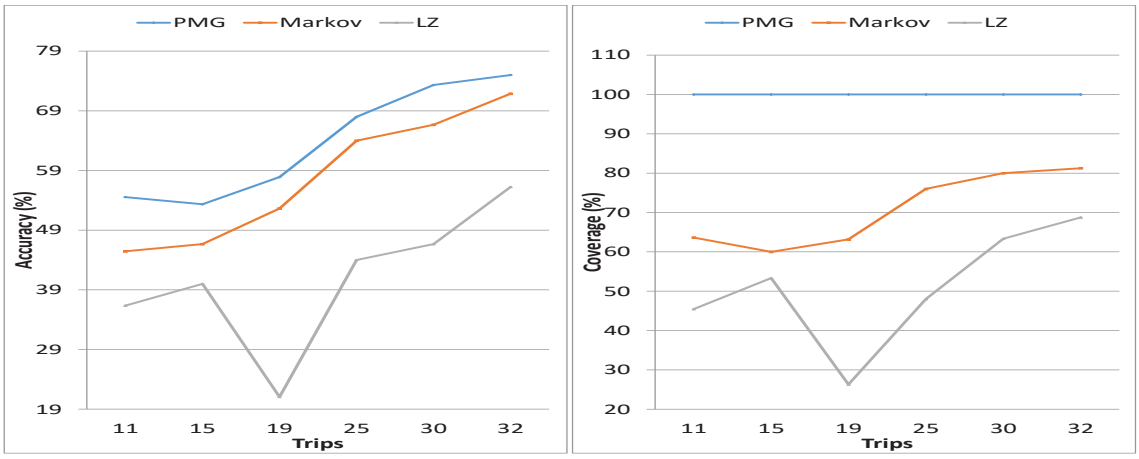


FIGURE 4.4: Performance evaluation of Driver 2.

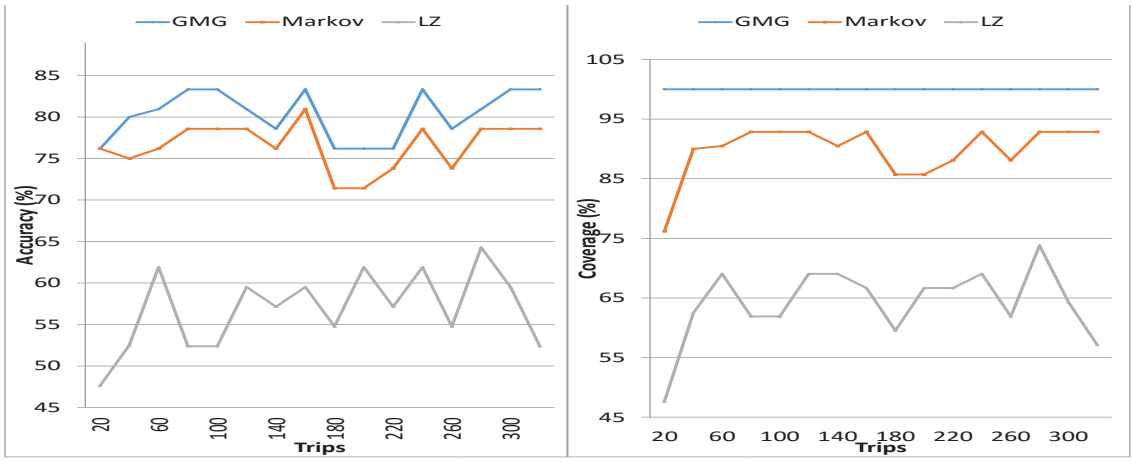


FIGURE 4.5: Performance evaluation of GMG.

4.4.2.1.1 Results

For PMG, obtained results indicate that prediction accuracy generally increases as the number of mobility sequences is increased. This could be ascribed to the fact that as more trips are considered, the more individual driver's trips are repeated and therefore intra-trips similarity increases. Besides, for GMG, interestingly, we had expected to obtain limited regularity using Cabspotting traces for global prediction. Taxicabs and their on-demand destinations are arbitrary and fully client–dependent which make them hardly predictable.

Nevertheless, a high degree of regularity was achieved. Results have shown regular and predictable patterns of taxi drivers which has confirmed the collective mobility behavior of taxi drivers as well. The spatial regularity captured is ascribed to the fact that multiple cabs may share common driving paths covering specific geographic areas and traveling to them to wait for new customers such as airports, hotels, shopping malls, touristic places, and therefore a high degree of similarity could be found in their movement exhibiting thus predictable mobility patterns.
Furthermore, overall, results depicted in Figure 4.5 have reported, that GMG exhibits more predictability comparing to PMG where GMG has achieved a prediction accuracy of about 83.5% with a number of 40 learned trips and around 70% of prediction coverage, outperforming PMG model.
Results have also shown that *MyRoute* models (PMG and GMG) outperform the PPM and LZ models. The obtained results exhibit that *MyRoute* is the best predictor along with the whole scenarios with a prediction accuracy between 50% and 60% with full coverage in most cases. *MyRoute* has also shown good performance even with a higher number of trips in global prediction where many heterogeneous trips from 10 different drivers were involved. In this experiment, the LZ predictor was the less performing model in terms of both accuracy and coverage in all cases. This could be referred to the fact that these models are too restrictive to data gaps and noise where 1) the smallest deviation in mobility may lead to different prediction and 2) they require a full match of a driver trajectory with the prediction model which is not always possible using GPS trajectories comprising erroneous and noisy data.
On the contrary, the proposed mobility graph structure, used in *MyRoute* is more resistant to noise. Incomplete or noisy trajectories are handled by creating transitions to several upcoming road segments, relying on the *lookahead* window. The way noise is handled in *MyRoute* also explains the high coverage obtained by *MyRoute*, compared to other models. The *lookahead* transitions in *MyRoute* model allow creating additional links between road segments, which often permits performing predictions by skipping segments when otherwise no prediction could be done. This, considerably increases the coverage. It is also observed, in this experiment, that the number of trips

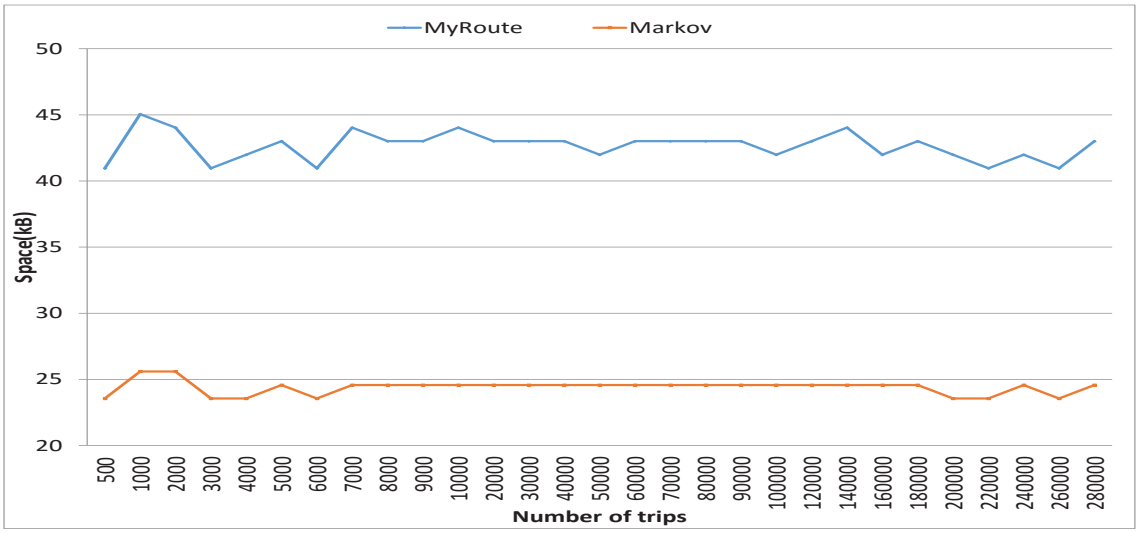


FIGURE 4.6: Model storage space.

is relatively small. This is because taxi driving is characterized by active mobility with few stationary states that exceed the stay point detection thresholds. Note that prediction times are not reported in this experiment as they were negligible for all models.

4.4.2.2 Experiment 2

The second experiment assesses the scalability of our model against the PPM Markov model in terms of model size and prediction time by increasing the number of trips. In this experiment, the LuST dataset, comprising more than 280.000 sequences, is used as it is more convenient for varying the size of the dataset.

This experiment is important since *MyRoute* is designed to be deployed in an online environment using mobile devices with low processing and storage capacities. Results of this experiment are shown in Figure 4.6 and Figure 4.7.

In Figure 4.6, we report how the model size is affected by an increase of the number of trips. Obtained results indicate that the number of trips does not have a great impact on model size. This is due to three main reasons. First, the number of nodes and possible transitions between them is static in the *RN*; thus, at a given time during graph construction the size of *MG* stays unchanged or shows a negligible augmentation if *MG* already comprises most of the roads in *RN*. Besides, as mentioned before, people always tend to take the same set of road routes to reach a given destination starting from the same source location (i.e. global behavior). Hence, these frequent routes are incorporated earlier in the graph, which avoids a noteworthy increase in the size of *MG* thereafter. Furthermore, the increase in the number of trips does not prominently affect the model size because when trips are added to the MG, only nodes and transitions that do not exist are added. As many of then overlap no additional arcs need to be created. In other words,

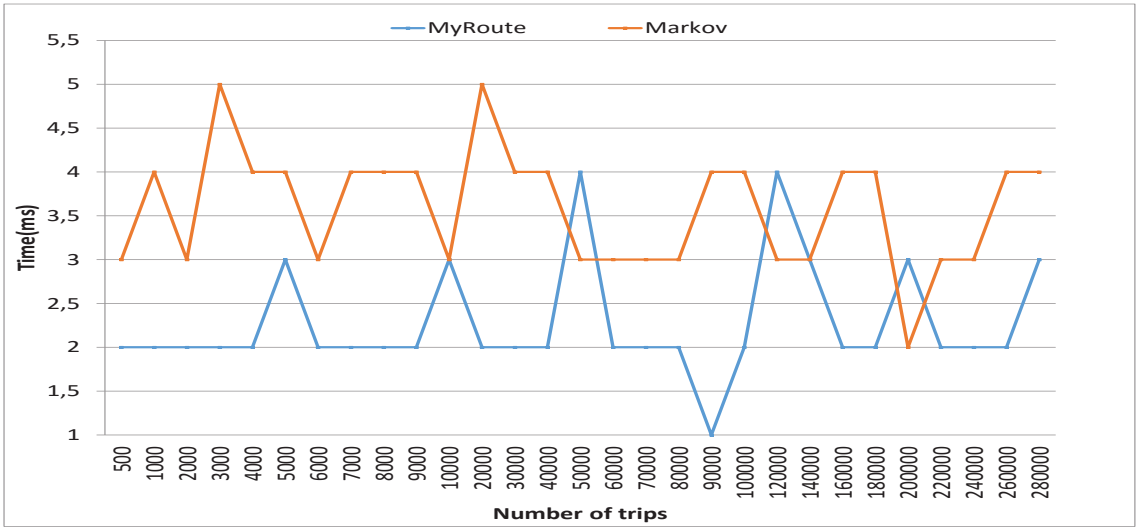


FIGURE 4.7: Prediction time estimation.

new arcs are added to a mobility graph only if they represent movements between road segments that were not previously encountered. Otherwise, only the weights are updated, thus the size mobility graph remains almost the same.
Besides, results also demonstrate that *MyRoute* requires more storage space than PPM. This is understandable since the number of nodes in both models is the same. However, more space is required in *MyRoute* due to the additional *lookahead* -transitions created and stored with their associated weights in the mobility graph. Even though *MyRoute* is the largest model in terms of model size, the mobility graph size remains small and suitable for real-life use as many mobile devices are currently equipped with several gigabytes of memory. Besides, in our graph description, we have only applied a standard graph representation where no storage space optimization was made. Yet, the graph size could be further optimized by adopting graph compression methods.

Regarding the processing time, Figure 4.7 shows that *MyRoute* requires less time to perform a prediction compared to PPM and that prediction time for both models takes only a few milliseconds. This experimental finding confirms the feasibility of deploying *MyRoute* for real-time prediction. These results can be explained by the fast incremental graph construction process that permits reusing the same graph structure by only inserting new connections and updating weights for movements in the *lookahead* window without the need to re-construct the graph from scratch. This accelerates and facilitates the graph building and the matching process as well.

4.5 Conclusion

In this chapter, a novel route prediction model has been proposed. *MyRoute* adopts a graph representation of mobility sequences where nodes are road

segments and each arc represents an ordering of two road segments. A distinctive characteristic of the proposed mobility graph structure is the creation of additional links to several upcoming road segments using a *lookahead* parameter. This provides noise tolerance abilities for route prediction. Moreover, *MyRoute* implements two prediction schemes called GMG and PMG to model both collective and individual human mobility behaviors, respectively. An extensive experimental evaluation has been conducted using synthetic and realistic datasets. From the experimental evaluation, several interesting conclusions could be deduced.

First of all, experiments have shown that the proposed *MyRoute* framework has good performance and scalability comparing to PPM and LZ predictors with an accuracy of 76% for PMG and 83.5% for GMG with a full prediction coverage. In particular, it outperforms PPM Markov model in terms of accuracy, coverage, prediction time, and scalability, but not in terms of model size. This is however understandable as PPM is intolerant to noise, whereas *MyRoute* stores additional arcs based on its *lookahead* window to provide noise tolerance.

Secondly, evaluation also shows how the adoption of a *lookahead* window parameter allows improving the performance and the tolerance of noisy and missing data.

Thirdly, in our model, we had focused on analyzing and studying the spatial regularity of human mobility. However, the temporal aspect of mobility was not investigated in this current study. To take this aspect into account, the prediction graph should be adapted accordingly by introducing the concept of a temporal dependency graph that incorporates the temporal context in the mobility graph [97].

Finally, by creating additional arcs the size of our model increases which highly affects the performance of the proposal. Our current graph storage scheme has not been optimized and thus is rather wasteful. Accordingly, additional work needs to be done by investigating methods that will substantially reduce the graph size via graph pruning and/or compression thus making it more suitable for real-time configurations.

Chapter 5

POI Recommendation: Literature Review

5.1 Introduction

The widespread use of handheld mobile devices and the increasing popularity of location-based social networks such as Foursquare have led to the emergence of an overwhelming number of location-aware services. One such service is Point-of-interest (POI) recommendation, which consists of suggesting locations to a user that he may be interested in visiting. This service aims to help mobile users discover interesting new locations (e.g. restaurants and stores), while on the move.

In recent years, numerous models have been proposed to support this service. These models apply several techniques (collaborative filtering, Markov chain, etc.) and consider various influence factors such as geographical, temporal and social influence.

This chapter begins with a background on the field of POI recommendation. It defines some key terms, reports the problem formulation and illustrates the set of factors that influences recommendation. Next, a literature review including the most relevant works in the field of POI recommendation is provided. It reveals the academic progress according to the target task (standard recommendation or successive recommendation that considers the sequential information in check-in data).

5.2 Location-based social networks

The widespread use of smart mobile devices has aroused the prosperous and popularity of Location Based Social Networks (LBSNs) such as Foursquare [1] and Yelp [3]. These networks assemble users' check-in data including their GPS records, visited locations and comments on these locations as well as other contextual data. For instance, Figure 5.1 portrays how the check-in information is recorded in Foursquare, comprising the user name, POI, check-in timestamp, and geographical position in the map.

LBSNs also allow users to define and create friend relationships and share information. Figure 1.3 depicts a typical LBSN, displaying the check-in activity of users and the friendship relationships among users.

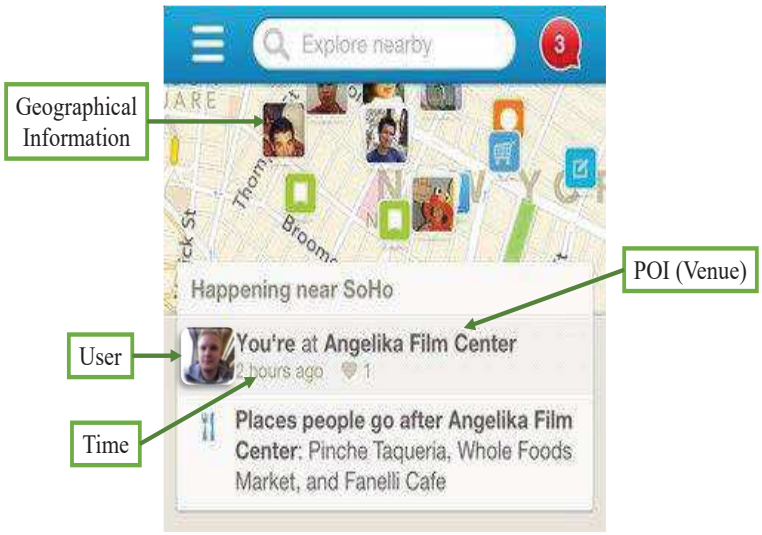


FIGURE 5.1: Collection of check-in information in Foursquare.

5.3 POI recommendation

POI recommendation aims at exploiting users' check-in data to suggest one or more locations for users in LBSN that they are deemed likely to visit in the future. In this section, we first define key terms in POI recommendation followed by the problem formulation of POI recommendation. Next, a description of a set of factors that influences the recommendation is provided. Finally, a brief description of the different types of mobility data used for POI recommendation is presented.

- **Definition 1 (Location or POI)**. A location is a limited geographical area, which has a unique identifier L_{id}.

- **Definition 2 (Check-in)**. Formally, a check-in L_i is a triple L_i=< u_i,T_i,L_{id}> that determines a user u_i has visited POI L_{id} at time T_i. Note that a check-in L_i=< u_i,T_i,L_{id}, Co_i> could be also extended to include a context Co_i of the check-in activity that contains user comments, user feedback, place type, etc.

- **Definition (check-in or location sequence)**. A check-in sequence $S =< u_i, T_i, L_i >, < u_i, T_j, L_{i+1} >, \ldots, < u_i, T_n, L_m >$ is an ordered list of check-in locations visited by a user u_i where T_k is the check-in timestamp and $k \in i, \ldots, m$ with $i \leq m$. For the sake simplicity, we sometimes denote a check-in or location sequence as $S = L_i, L_{i+1}, \ldots, L_m$.

  POI recommendation suggests to a user a set of POIs via mining the check-in records. Given Definitions 1 and 2, the problem of POI recommendation can be defined as follows:

- **Definition (POI recommendation)**. POI recommendation aims to recommend to a user u the set M of POIs that he may be interested in

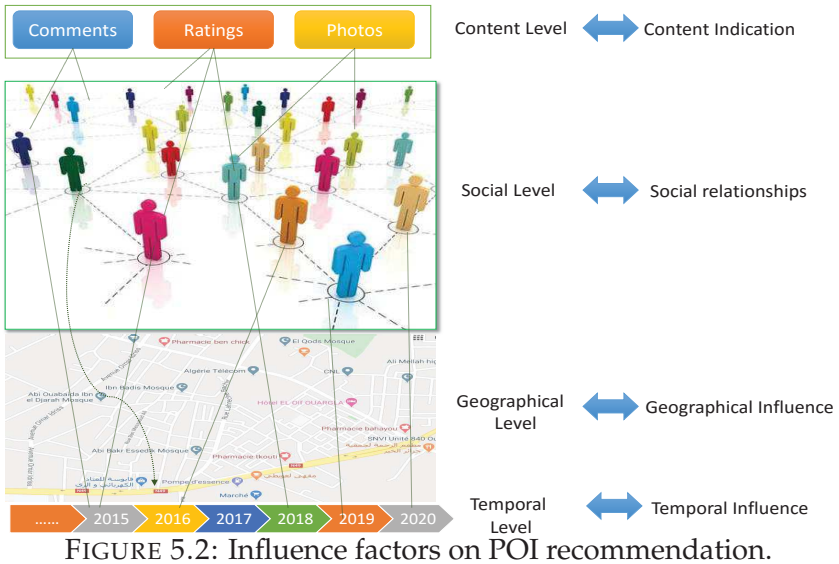


FIGURE 5.2: Influence factors on POI recommendation.

visiting in the future given the check-in data, or the set of check-in sequences whenever the sequential order of POI is considered, for all users.

5.3.1 Influence factors

To increase the effectiveness of POI recommendation systems, contextual check-in information such as time, geographical positions, social relationships and geo-tagged content (photos, comments, etc.) were considered. These recommenders thus consider the temporal, geographical, social, and content indication factors for recommendation (Figure 5.2). In the following, we first describe how each of the above factors influences the check-in activity. Next, a brief description of the sequential factor in recommendation is given. The latter, widely considered in recent studies, matches with the successive POI recommendation task we investigate in this research.

- **Temporal Influence**. The temporal influence factor is critical for recommendation. It indicates that a user destination is generally time dependent. For instance, a user may tend to go to his workplace in the morning and to eat at a restaurant at lunchtime, during weekdays, whereas he may visit shopping malls during weekends. Thus, users' preferences vary depending on time and that a temporal regularity can be observed in human mobility behavior.

- **Social Influence**. Another important influential factor is social influence, which is based on a user's behavior and preferences, and his relationships to other users. The social influence factor is based on the idea that friends often share common interests, tastes and preferences. In other words, users may follow their friends' recommendations and visit the same places. For example, friends may workout at the same

gym, and sometimes eat at the same restaurant.

- **Geographical Influence**. The geographical influence factor is an important factor that differentiates the POI recommendation from conventional item recommendation because the check-in activity highly dependent on locations' geographical properties. This factor is based on the assumption that a user is more likely to visit nearby locations than distant ones. In other words, users may prefer to eat at a restaurant close to their current location or residency (e.g. work or house) instead of a distant one.

- **Content indication**. In LBSN, users could generate content including geo-tagged photos, comments, messages, tips and ratings for the visited POIs. This content may explicitly or implicitly express the user preferences, satisfaction, sentiments regarding the POI which lead thus to boost recommendation efficiency [98]. Indeed, content such as users' comments provides an explicit mean to recognize the preference of users and deduce the check-in behavior rather than only depending on check-in activity. For instance, if a user has checked-in at a Chinese restaurant, two deductions could be inferred. First, the user likes Chinese food but may dislike this restaurant or likes both the Chinese food and the restaurant. With the existence of comments on this check-in activity, an extra information is obtained. Thus, more accurate recommendation could be achieved.

- **Sequential influence**. The sequential influence aims at considering the sequential nature of human mobility behavior in recommendation [36–38, 99]. This factor considers the varieties of users' needs depending on their recent locations and movements. For instance, Figure 5.3 illustrates a sample of check-in sequences followed by three users. *User 1* usually leaves home to go to university, then have lunch at a restaurant, and finally go to a shopping mall. This phenomena and sequential behavior of movement is called *sequential influence factor*.

  By considering this factor, a novel task so-called *successive POI recommendation* has emerged attracting a great research interest. Recommenders for this task consider the hidden sequential relationships in the mobility behavior of users to perform recommendation. In particular, given the check-in sequence of a user comprising his current location and his historical movements (if available), successive POI recommendation consists thus to generate a list of POI suggestions to locations that the user may be interested in visiting in the near future.

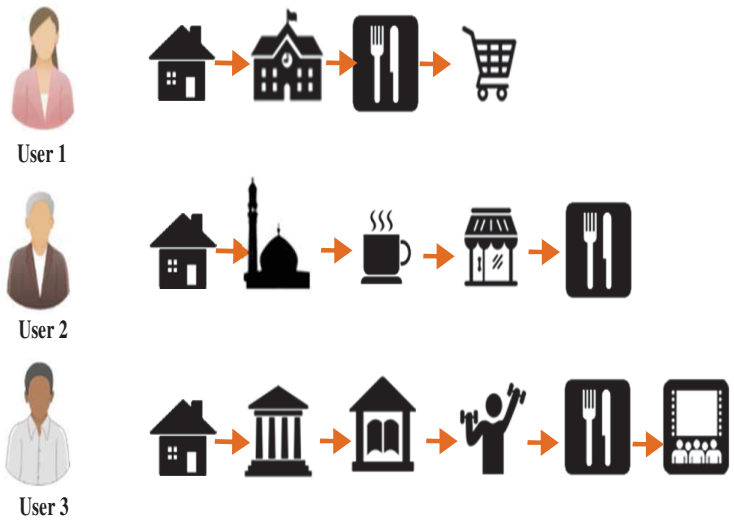


FIGURE 5.3: Sequential check-in activity of three users.

5.3.2 Mobility data

Different types of data are used in recommendation. POI recommenders could be divided into recommenders that exploit GPS data [24, 25], check-in data [11,22,23,37,100] or geo-tagged content (photos, comments,etc) [39,101].

- **Recommendation using GPS data**. Mobile devices, equipped with GPS, constitutes sensors of human mobility as they are always held by users as long as they move and a rich source of mobility data. These devices have promoted a set of location-based services including POI recommendation.
  The GPS data used is collected through voluntary users of Web-based GPS data management service while user comments regarding to geographical visits are utilized.
  However, using GPS data for recommendation suffers from several limitations 1) The availability of GPS data is limited due to the privacy issues as most people don't feel comfortable sharing their real mobility data, 2) content and social context is usually not included in GPS data; thus, only the spatial and temporal patterns are considered in most cases, and 3) POI identification mainly relies on stay point detection process. However, it is not easy to distinguish between closer locations and also identify the type (category) of each POI where the use of a third-party library to indicate the type of each stay point is mandatory.

- **Recommendation with check-in data**. To overcome and alleviate the main limitations of GPS-based recommenders, most of current POI recommendation proposals are typically performed using check-in data. Nowadays, most recent studies show that POI recommendation is performed with LBSN data, because they are readily available and give an

Dataset	Type	Statistics	Period
Geolife [102]	GPS trajectory	17,621 trajectories from 182 users	April 2007-August 2012
Brightkite [103]	Check-ins & Friendships	4,491,143 checkins from 58,228 users	April 2008 - October 2010
Gowalla [103]	Check-ins & Friendships	6,442,890 check-ins from 196,591 users	February 2009 – October 2010
Twitter [104]	Geo-tagged Tweets	22,506,721 tweets from 225,098 users	September 2010 -January 2011
Foursquare 1 [105]	Check-ins & Friendships	22,809,624 check-ins from 114,324 users on 3,820,891 venues	April 2012-January 2014
Foursquare 2 [106]	Check-ins	33,278,683 checkins by 266,909 users on 3,680,126 venues (in 415 cities in 77 countries)	April 2012 - September 2013
Foursquare 3 [107]	Check-in,tip and tag data of restaurant venues	3112 users and 3298 venues with 27149 check-ins and 10377 tips	October 2011-February 2012

TABLE 5.1: Sample of LBSN datasets for POI recommendation.

easy access to user activity records on a large scale. In Table 5.1, a brief description of a sample of check-in datasets and their main characteristics is portrayed.

5.4 Literature Review

To reveal the academic progress in the area of POI recommendation, this section gives an overall overview of the relevant studies on POI recommendation. As portrayed in Figure 5.4, these studies could be classified according to the target task (general or successive), type of the method used, and influence factors considered. In this section, we categorize POI recommenders relying on the target task (general POI recommendation and successive POI recommendation).

5.4.1 General POI recommendation

General POI recommendation is a sub-field of recommendation systems that borrows ideas from conventional recommendation systems such as movie recommendation and considers specific properties of human mobility on LBSNs. General POI recommendation includes content-based, traditional collaborative filtering (CF) and extended collaborative filtering recommenders. In particular, we discuss the consideration of three influential factors in the extended CF namely: geographical, social and temporal influences.

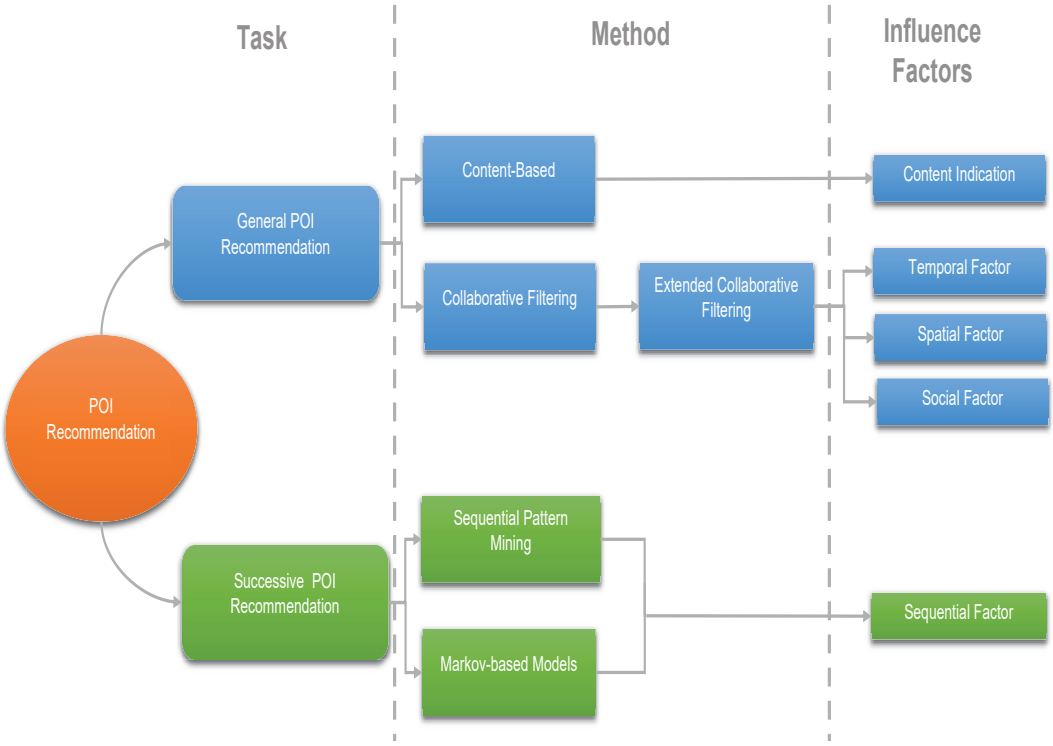


FIGURE 5.4: Taxonomy for POI recommendation.

1. **Content-based recommendation**. In LBSNs, users could generate rich contents including tips, ratings for POIs, and geo-tagged photos about the POIs. Though these contents are not always available for each check-in record, the availability of such contents especially the user comments, plays a significant role in improving POI recommendation [35, 98, 107–110].
   Content-based recommender tries to suggest new locations (items) similar to a set of locations a given user has liked in the past. Most content-based recommenders usually consist of a chained process comprising three stages, location abstraction, user profiling and eventually POI recommendation.

   - Given the set of mobility data and social content (comments, photos, etc.), location abstraction aims at examining, summarizing and representing locations, previously rated by users, by a set of features (e.g. type, position, etc.).
   - The user profiling stage consists of building a model on profiling interests, relying on the obtained features, by learning users' preferences from their mobility histories. For instance, the features are used to train machine learning models such as classification where data is divided into two classes: locations that the user "likes" and locations that the user "dislikes".

   In literature, the studies in [35, 110] have used location-tagged social network contents (e.g. publications, comments, photos, geo-tagged

contextual information (i.e. tags of POIs)) as well as sentiment information to infer a user score for POIs. Yin et al. [35] have proposed a location content-aware recommender that exploits users' content information about their favored locations in one city (e.g., home city) to suggest a list of POIs (spatial items) in a newly visited city. Their proposal is composed of offline modeling and online recommendation parts. The offline modeling part is mainly designed to learn personal interest of each user and the local preference of each city by capturing item co-occurrence patterns and exploiting item contents. To do so, a projection of user's activity history into a latent space, which integrates content knowledge of spatial items, is performed.
To recommend the top k locations for a user u_i in a given city C_i, the online recommendation part, intuitively merges the learned interests of u_i and the local preference of C_i. Yet, the main limitations of content-based recommender is that accurate recommendation is conditioned with the respect of 1) the availability of sufficient context and 2) the appropriate context analysis.
Actually, a good contextual location description help efficient differentiation of locations that a user likes from those that a user dislikes [111]. Nevertheless, such description is not always available due to users' privacy issues where many of them might decline sharing their mobility context with LBSN operators. Besides, an adequate content analysis should be also performed to avoid the over specialization [111] principally due to limited or incomplete information about the contents of the location or the users. Note that content analysis can be also considered as a pre-processing step to discard irrelevant locations, which consequently speed up recommendation.
To deal with the two aforementioned limitations, collaborative filtering-based recommenders have been proposed. These recommenders are used to accommodate these issues as they only exploit the user-item matrix without requiring more contextual data about user interests.

2. **Traditional POI recommendation**. Traditional POI recommendation systems, which use collaborative filtering (CF) technique, have been proven effective in practice for POI recommendation [112, 113].
Unlike content-based recommenders, in which the core idea is to recognize the common features of locations that have already been visited, evaluated and rated by a user, collaborative filtering approaches mainly depend on the ratings of a user along with other users ratings in the system [114]. CF-based recommenders mainly consist of collecting and analyzing data from LBSN users, including their behaviors activities or preferences, to capture collective patterns of agreement among users' ratings for items (i.e.POI). They discover and exploit similarities between persons (preferences, behaviors, habits), and between locations (categories, distance, visit duration), to perform recommendation.
The preferences or rating values could be obtained explicitly when a

user assigns a rating value on a POI or implicitly basing on the users visiting of a given POI by users.
The input of a CF based recommender is a user-item rating matrix (i.e., user-POI frequency matrix). The underlined assumption of these recommenders is that if two users have similar behavior pattern (e.g., visiting similar POIs) in the past, they will most likely behave similarly in the future visiting the same or similar locations.
In literature, CF based recommenders generally apply memory-based [22, 23, 26, 101] or model-based models [24, 25, 33, 34].

(A) **Memory-based CF**. To estimate the rating that a user would give to a location, memory-based approaches take social relationships between users into account. POIs are recommended to users by considering ratings and feedbacks given by experts, users and their friends. Memory-based models are further categorized as item-based [26] and user-based [22, 23, 29].

(a) **User-based CF**. User-based POI recommenders mainly rely on the assumption indicating that similar users have similar tendency to visit similar locations. A user-based CF recommender could be applied in what follow. Assume that we want to recommend N POIs to a user u_a. First, relying on a similarity measure metric, the recommender selects the top k users similar to u_a. Then, it deduces the set $P_{candidate}$ of POI not previously visited by u_a and has been preferred by similar users to u_a. Finally, the recommender ranks u_a's preferences on $P_{candidate}$ and retrieves the top N POIs as recommendations. More formally, for a given user u_a, user-based recommendation system first estimates the similarity $sim(u_a, u_i)$ between u_a and another user u_i. Then, the rating value to a location $l_j \in P_{candidate}$, which u_a has never visited before, is calculated by Equation 5.1.

$$P_{u_a,l_j} = \bar{r}_a + \frac{\sum_{u_i \in U_{l_j}} (r_{i,j} - \bar{r}_i) \cdot sim(u_a, u_i)}{\sum_{u_i \in U_{l_j}} sim(u_a, u_i)} \tag{5.1}$$

where U_{l_j} includes all the users that have visited l_j, $\bar{r}_a$ and $\bar{r}_i$ are the average ratings for the users u_a and u_i have given to all other visited locations, and $r_{i,j}$ is the rating u_i gives to l_j .
To recommend locations around a geographical area, which could be located in the same city or a nearby city, Boa et al. [22], have proposed a location-based and preference-aware recommender system, which considers user mobility history. It first learns the user's preferences from previously visited locations. Then it exploits feedbacks and opinions from people living in that area and which share common interests with

that user, considered as local experts, to perform recommendations. This approach was shown to be especially useful for tourists or persons that visit a city or neighborhood for the first time.

(b) **Item-based CF**. Item-based recommender follow the same process as of user-based recommender except it estimates the similarity between items instead of users. The item-based collaborative filtering aims at discovering the k most similar items (i.e. locations) to a target item l_t. This can be done by computing the similarity between the locations that the user u_a has rated and l_t. The rating prediction for u_a to l_t can be made by taking a weighted average of all the ratings of the user is given in Equation 5.2.

$$P_{u_a,l_j} = \frac{\sum_{l_j \in L_{u_a}} r_{a,t} \cdot sim(l_t, l_j)}{\sum_{l_j \in L_{u_a}} sim(l_t, l_j)} \tag{5.2}$$

where the L_{u_a} denotes the set of all the locations visited by u_a, $r_{a,t}$ is the rating or score given to location l_t by u_a, and $sim(l_t, l_j)$ denotes the similarity between locations l_t and l_j .
Once the most similar items are found, the prediction is then computed as a weighted average of u_i's ratings on these similar items.

In literature, Levandoski et al. [26] have suggested an item-based CF called LARS (Location Aware Recommendation System). The latter classifies location-based ratings into three types to recommend POIs: 1) spatial ratings for non-spatial items represented as a four-tuple (*user*, *ulocation*, *rating*, *item*), where *ulocation* represents a user location, 2) non-spatial ratings for spatial items represented as a four-tuple (user, rating, item, ilocation), where *ilocation* represents an item location, 3)and spatial ratings for spatial items represented as a five-tuple (*user*, *ulocation*, *rating*, *item*, *ilocation*). Based on the type of location-based rating available, LARS applies two techniques namely user partitioning, travel penalty, in an isolate or combined fashion, to support spatial ratings and spatial items. While user partitioning is used to exploit user locations with ratings spatially close to querying users, travel penalty technique exploit users' locations to avoid exhaustively processing all spatial recommendation candidates and penalize candidates far from the users.

Similarity measure. The core component of memory-based methods is how to compute the similarity weight $sim(u_i, u_j)$ between user u_i and u_j or between two locations l_i, l_j denoted as $sim(l_i, l_j)$. In literature, two main similarity metrics have been utilized namely

Jaccard coefficient and Pearson correlation.

- **Jaccard coefficient**. This similarity measure is defined as the size of the intersection divided by the size of the union of the two sets. Using Jaccard coefficient, the similarity between two users u_i and u_j in the user-based recommender is calculated by Equation 5.3, and the similarity between two locations l_i and l_j in item-based recommender is calculated by Equation 5.4.

$$sim(u_i, u_j) = \frac{\left|L_{u_i} \cap L_{u_j}\right|}{\left|L_{u_i} \cup L_{u_j}\right|} \tag{5.3}$$

$$sim(l_i, l_j) = \frac{\left|U_{l_i} \cap U_{l_j}\right|}{\left|U_{l_i} \cup L_{l_j}\right|} \tag{5.4}$$

  where L_{u_i}, and L_{u_j} denote the locations visited by the user u_i and u_j, respectively whereas U_{l_i} and U_{l_j} symbolizes all the users who have visited the locations l_i and l_j, respectively.

- **Pearson correlation**. Pearson correlation measures the linear correlation between two variables X and Y. It is calculated as the covariance of the X and Y divided by the product of their standard deviations. Using Pearson correlation, the similarity between two users u_i and u_j in the user-based recommender is calculated by Equation 5.5, and the similarity between two locations l_i and l_j in the item-based recommender is calculated by Equation 5.6.

$$sim(u_i, u_j) = \frac{\sum_{l_k \in L_{u_i,u_j}} (r_{i,k} - \bar{r}_i)(r_{j,k} - \bar{r}_j)}{\sqrt{\sum_{l_k \in L_{u_i,u_j}} (r_{i,k} - \bar{r}_i)^2} \sqrt{\sum_{l_k \in L_{u_i,u_j}} (r_{j,k} - \bar{r}_j)^2}} \tag{5.5}$$

$$sim(l_i, l_j) = \frac{\sum_{u_k \in U_{l_i,l_j}} (r_{k,i} - \bar{r}_i)(r_{k,j} - \bar{r}_j)}{\sqrt{\sum_{u_k \in U_{l_i,l_j}} (r_{k,i} - \bar{r}_i)^2} \sqrt{\sum_{u_k \in U_{l_i,l_j}} (r_{k,j} - \bar{r}_j)^2}} \tag{5.6}$$

  where L_{u_i,u_j} is the set of all the locations visited by both users u_i, u_j and U_{l_i,l_j} is the set of all the users who visited both locations l_i and l_j with $r_{i,j}$ is the rating u_i gives to l_j.

Discussion. Although memory-based models were shown to be

easy to implement and to perform well in recommendation, these models have several drawbacks. First, memory based models exhibit poor performance with sparse data. They provide low accuracy on sparse location check-in datasets as not enough information is available to calculate the similarity between users, ratings and locations. Therefore, the inferred similarity measured from the history of check-in activity may not be reliable [115]. Second, for newly seen users, which do not have user profiles, no rating or check-in history would exist and thus, the similarity value of these users to any other users is 0. Furthermore, memory-based CF are known by their poor scalability on large dataset as they should load the whole user-item matrix to produce recommendation which is spatially and computationally expensive.

(B) **Model-based**. In model-based CF approaches, machine learning methods, such as Bayesian networks [33,34], clustering [24], latent Dirichlet allocation [11,116], classification [117] and matrix factorization [12,31,32], have been utilized. In contrast to memory-based CF approaches, model-based CF approaches are more scalable and can better cope with sparse check-in datasets. Overall, two main categories of model-based CF could be distinguished: latent factor models and classification or regression models.

(a) **Latent factor models**. The matrix factorization model is the most used method among latent factor models. It mainly relies on the assumption that there are certain latent factors related to both users' preferences and items' properties. Take a restaurant in POI recommendation as an example, latent factors could be price, flavor, etc.
These latent factors may affect the check-in activities where each check-in is a result of the joined influence of user's preferences and location's property on these factors. For the the sake of illustration, a user who likes Chinese food and has concerns in the price may be interested in a restaurant that serves Chinese food with a price discount.
In [31], a weighted matrix factorization based recommender was proposed to deal with the data sparsity problem. Their proposal takes into account the spatial clustering phenomena of human mobility, i.e., individual visiting locations tend to cluster together, and considers not only the user and POI latent factors but also activity and influence area vectors. All these latent factors are used in the factorization model to capture the spatial clustering phenomenon in terms of two dimensional kernel density estimation and also clarify why the incorporation of such a phenomenon into matrix factorization helps to deal with data sparsity.
To perform accurate context-aware recommendation, the model

of Yan et al. [33] has considered each user's mobility behavior from four perspectives: user, geographical information, time, and activity. The model applies the well-known *latent Dirichlet allocation (LDA)* technique on user feedback and geo-tagged contextual information (i.e. tags on POIs) to infer a user score for POIs.

(b) **Classification and regression models**. Similar to standard classification-based methodologies, recommending locations using classification techniques require a preliminary step to select and then extract features from training check-in data. Once the set of features are retrieved, a learning model is built. It consists of assigning a positive label to observed pairs of user and item (i.e., a user has checked-in at that POI) and negative label otherwise. Based on these labels, a classification or a regression process is performed on the testing check-in data to infer the likelihood of a user's preference in a POI.
In [117], Noulas et al. have addressed the problem of predicting the next venue for users of LBSN. Their proposal consists of selecting a set of key features that could identify the factors of driving use displacement such as transitions between types of locations, place popularity, mobility flows between places. To increase the efficiency of their proposal, the authors have extended the system by combining these features using linear regression and $M5$ model trees.

3. **Influence factors consideration**. To increase the effectiveness of traditional CF based recommender systems, CF recommenders have been extended with contextual check-in information such as time, geographical positions and social relationships. These extended systems thus consider the temporal, geographical and social influence factors for recommendation (see section 5.3.1).

(A) **Geographical influence**. To address the problem of POI recommendation by considering the geographical influence, several studies have been proposed [15,103,118–123] where the main ones are:

(a) **Power law mobility pattern**. Power law mobility pattern relies on two main observations. First, some studies have shown that people always tend to visit nearby places rather than places far away and that almost 20% of successive check-ins are recorded within 1km while 60% between 1 and 10km [103,124]. Besides, it has been also shown that few places are frequently visited and checked-in by users whereas most of other places have few check-ins and rarely visited. The probability distribution of co-occurrence of two POI by the same user over the distance between the two POIs is depicted in Figure 5.5 and formulated as follow.

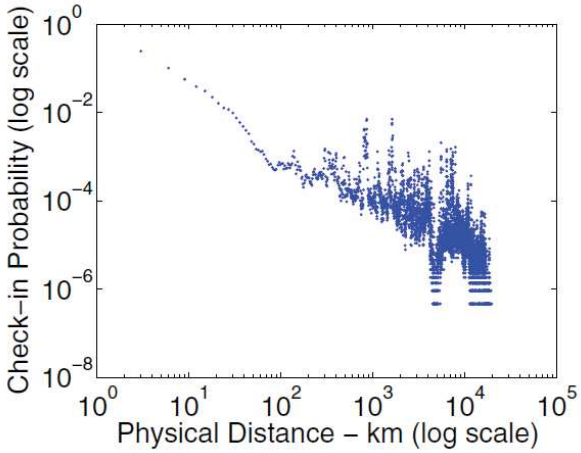


FIGURE 5.5: Power law distribution pattern [15].

$$y = a * x^b \tag{5.7}$$

where x represents the distance between two POIs. The parameters a and b are the parameters of the power-law distribution that could be obtained by fitting a linear regression model after applying a logarithmic operation to Equation 5.7. Relying on the power low distribution, the probability of visiting l_j for user u_i given the past checked-in locations is computed as follows:

$$Pr(l_i|L_i) = \frac{Pr(l_i \cup L_i)}{Pr(l_i)} = \prod_{l_y \in L_i} Pr(d(l_i, L_i)) \tag{5.8}$$

here $d(l_j, l_y)$ represents the distance between l_j and l_y, and $Pr(d(l_j, l_y)) = a * d(l_j, l_y)^b$. In the context of POI recommendation, the approach of Ye et al. [15] mines patterns that are similar for several users in their check-in activity (e.g. withdraw activities at ATMs) and extract power law distribution patterns from mobility data.

(b) **Gaussian distribution**. Gaussian distribution based models rely on the assumption that a user's mobility tend to be around some activity centers such as one's home and workplace [103]. This assumption has motivated Cheng et al. in [119] to propose a Multi-center Gaussian Model (MGM) to capture geographical influence on user preferences. The conditional probability of visiting the POI l given the multicenter set C_u $P(l|C_u)$ for user u is formulated by Equation 5.9.

$$P(l|C_u) = \sum_{c_u=1}^{|C_u|} P(l \in c_u) \frac{f_{c_u}^{\alpha}}{\sum_{i \in C_u} f_i^{\alpha}} \frac{N(l|\mu_{C_u}, \sum_{C_u})}{\sum_{i \in C_u} N(l|\mu_i, \sum_i)} \tag{5.9}$$

FIGURE 5.6: Check-in distribution in multi-centers [103].

where $P(l \in c_u) \propto \frac{1}{d(l,c_u)}$ denotes the probability of the POI l belonging to the center c_u whereas $\frac{f_{c_u}^{\alpha}}{\sum_{i \in C_u} f_i^{\alpha}}$ denotes the normalized effect of the check-in frequency on the center c_u and the parameter α maintains the frequency aversion property. $N(l|\mu_{C_u}, \sum_{C_u})$ is the probability density function of Gaussian distribution with mean μ_{C_u} and covariance matrix $\sum_{C_u}$.

(c) **Kernel density estimation**. Instead of using power law and Gaussian distribution models to capture geographical influence from a global perspective, Zhang et al. in [121] have proposed a POI recommendation system to assess personalized geographical influence using a kernel density estimation (KDE) model [125]. Zhang et al. have claimed that considering the geographical influence on each distinct user should be personalized rather than modeling though a common distribution. As shown in Figure 5.6, it is difficult to use the same distribution for different users. To this end, Zhang et al. [126] have leveraged the kernel density estimation KDE [125] to model the geographical influence using a personalized distance distribution for each user. To do so, KDE first generates a sample X_u for a user by calculating the distance between every pair of locations visited by the user. Then, the distance distribution can be computed using the probability density function f over distance d formulated as follow.

$$f(d) = \frac{1}{|X_u|\,\sigma} \sum_{d' \in X_u} K\frac{(d - d')}{\sigma} \tag{5.10}$$

where σ is a smoothing parameter, called *the bandwidth*. $K(.)$ denotes the Gaussian kernel that is formulated as follow.

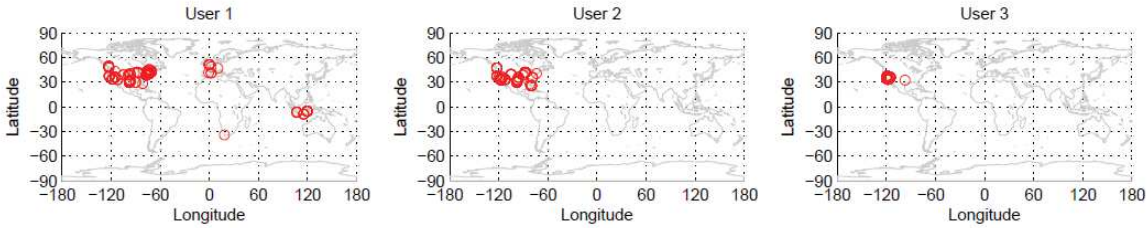


FIGURE 5.7: Distributions of personal check-in locations [121].

$$K(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$$

Given $L_u = l_1, l_2, \ldots, l_n$ as the visited locations of user u. The conditional probability of visiting a new POI l_j given the past checked-in POI set L_u is formulated as:

$$P(l_i|L_u) = \frac{1}{|L_u|} \sum_{l_i \in L_u} f(d_{ij})$$

where d_{ij} is the distance between l_i and l_j , $f(.)$ is the distance distribution function in Equation 5.10.

(B) **Social influence**. Social-based recommendation is built on the idea that friends often share common interests, tastes and preferences. Thus, users may follow their friends' recommendations and visit the same places. This factor has been broadly considered in POI recommendation [15, 19–21]. Specifically, Ye et al. [15] have considered both geographical and social influences for recommendation. Their recommender system uses a Bayesian and CF based system that considers preferences of friends as more important than those of other users to perform recommendations.

In contrast, and although several studies have shown the benefit of considering social relationships in recommendation, other studies [127] have demonstrated that around 96% of friends share less that 10% common visited points of interests. This indicates that a large portion of user's friends choice in terms of POI does not affect one's future interest. Hence, social influence may provide a limited contribution on users' check-in behaviors. For example, friends who love Chinese food from different towns will visit their own local Chinese restaurants. This phenomenon changes from other traditional recommendation scenarios such as the online movie recommendation in Netflix or item recommendation on Amazon.

(C) **Temporal influence**. The temporal influence factor is critical for recommendation. It mainly relies on temporal regularity observed in human mobility. This factor indicates that a user destination is generally time-dependent. Based on the assumption that user

preferences vary depending on time, several recent studies have considered this factor [17, 18, 118]. For instance, Yuan et al. [118] and Gao et al. [17] have designed systems that partition a day into several time periods (slots) of the same length. Then, their recommenders perform recommendations that are tailored to the time of the day.

Particularly, Gao et al. in [17] have proposed a location recommendation framework, called *LRT*, based on a low-rank matrix factorization model with time effect consideration. *LRT* consists of three steps: temporal division, temporal factorization, and temporal aggregation. Temporal division aims at subdividing the user-location into T sub-matrices according to the T temporal states where $T = 24$ and each temporal state $t \in [1, T]$ corresponds to an hour of the day. Thus, each sub-matrix only contains the check-in activity of each temporal state. Temporal factorization consists of factorizing each resulting sub-matrix into user check-in preferences U_t and location characteristics L. Finally, to integrate a user's check-in preferences of different temporal states, a low-rank approximation is constructed and aggregated using four temporal aggregation strategies namely: sum, mean, maximum and voting.

(D) **Fusing several influence factors**. Up to this point, we have mainly focused on studies, which regard only one influence factor at once. However, most recent researchers tend to regard and fuse several influence factors in the same recommender. For instance, authors in [121, 127] have considered the geo-social influence in their proposed recommender. The geo-social influence indicates that people tend to explore nearby POIs that they or their friends have visited before. The proposed works have exploited geographical influence and social relationships independently, and then combine their input and/or output in a fused fashion.

5.4.2 Successive POI recommendation

All the aforementioned studies have considered only the geographical, temporal and social influence factors. These studies consider several contextual information but ignore the sequential behavior of user mobility when visiting POIs. They thus assume that preferences for visiting POIs are independent of their visiting orders. Besides, these models do not take the current location of a user into consideration and hence fail to recommend the immediately next location that the user is likely to visit.

Recently, the task of successive POI recommendation, that considers the sequential influence, has been widely studied [14, 36–44, 100, 128–134]. Recommenders for this task take into account the hidden sequential relationships in the mobility behavior of users to perform recommendations. More specifically, given the current location of a user and his historical movements

(if available), a set of POI suggestions to locations that the user may be interested in visiting in the near future are generated.

To accomplish this task two representative models, namely Markov chain [36, 39, 41, 134] and sequential patterns mining [14, 37, 42–44, 100] have been employed.

1. **Markov-based recommendation**. Markov chain models have been widely used in recommendation as a core method in successive POI recommender or partially combined with other techniques. Specifically, n-order Markov models (see description in Section 1) have been utilized to perform recommendations based on the current user location and his n previously visited locations, where n is the order of Markov chain.

   For instance, the authors in [36] have proposed a Markov model with region localization. Their recommender implements a novel matrix factorization method, so-called *Factorized Personalized Markov Chain Local Region (FPMCLR)*, that incorporates a personalized Markov chain in the check-in sequence and focuses on the localized region that a user is moving around. By considering only the localized region of users, the authors has reported that a significant reduction of the computation cost has been achieved while discarding noisy data which consequently boots recommendation accuracy. In another work, Kurashima et al. [39] have embed user preference and his current location into the probabilistic behavior model by fusing topic models and Markov models. Their proposal tries to infer the photographer's behavior using 1) a topic model, based on probabilistic latent semantic analysis (PLSA) [135], to measure the personal interest of photographers and 2) a Markov model to predict the typical routes of photographer. To do so, the authors have utilized geo-tagged and time-stamped photos as personal travel route histories of users.

   Even though Markov-based recommenders have the advantage of being simple and easy to implement their main limitation is that the parameter n representing the number of previous locations to be considered must be set. Considering more than one location to perform prediction can lead to better performance by finding sequences of locations visited by multiple users sharing common mobility behavior. However, this may also significantly increase the time required to perform recommendation. Another limitation of Markov models is that many of them assume that the Markovian hypothesis holds, which states that knowing the last n locations of a user would be sufficient to accurately recommend the next location. Nevertheless, this assumption is not always true. Thus, it may result in low recommendation accuracy. Moreover, Markov models are order sensitive. In other words, a small variation in the sequential order of previously visited locations implies different recommendations even when the same set of locations has been visited. Lastly, as the order n of a Markov model is increased, its size can

grow exponentially with respect to the number of locations. Thus, several Markov models have a large space complexity, which makes them unpractical in real-life application.

2. **Sequential patterns based recommendation**

   (A) **Sequential patterns mining**. The task of sequential patterns mining consists of finding all sequential patterns appearing in at least 2 sequences of the input sequence database. In what follows, a set of definitions, partially adapted to the context of recommendation, are provided to well understand the problem of sequential patterns and sequential rules mining.

   (a) **Definitions**

   - **Definition (Sequence database and sequence)**. A sequence database SDB is a set of sequences where each sequence is assigned to a unique *sid* (Sequence ID). Given the set of items (here also called locations), a sequence is an ordered list of items $S_k =< L_1, L_2, \ldots, L_p >$ such that $L_1, L_2, \ldots, L_p \in L$ with L denoting a set of locations. For instance, Table 5.2 depicts a sample of a sequence database containing three sequences having respectively the *sids* S_1, S_2, and S_3.
   - **Definition (Subsequence)**. A sequence $S_1 =< L_{x1}, L_{x2}, \ldots, L_{xn} >$ is said to be a subsequence of a sequence $S_2 =< L_{y1}, L_{y2}, \ldots, L_{ym} >$ (denoted as $S_1 \subseteq S_2$), if and only if there exist integers $1 < a_1 < a_2 < \ldots < a_n \leq m$ such that $L_{x1} = L_{ya1}, L_{x2} = L_{ya2}, L_{xn} = L_{yan}$. For example, consider the SDB of Table 5.2, the sequence $S =< L_1, L_2, L_3 >$ is a subsequence of two sequences S_1 and S_2.
   - **Definition (Sequential pattern)**. A sequential pattern is a subsequence of many sequences in the sequence database. A sequential pattern X is said to be a frequent if the number of sequences in SDB that contains X divided by the number of sequences in SDB exceeds a specific threshold called *minimum support threshold (minsup or minfreq)*. Thus, this parameter specifies a minimum number of sequences in which a pattern must occur to be considered frequent. For example, the sequences $S_1 =< L_1, L_2 >$ and $S_2 =< L_8, L_7 >$ are sequential patterns, however, only $< L_1, L_2 >$ is considered as a frequent sequential pattern when *minfreq*=0.66.
   - **Definition (Sequential rule)**. A sequential rule $R_i : X \Rightarrow Y$ can be defined as a relationship between two sequential patterns $X, Y \subseteq L$ such that $X \cap Y = \emptyset$ and $X, Y \neq \emptyset$. X is called the *antecedent* of R_i whereas Y is the *consequent of* R_i.

SID	Sequences
S_1	$< L_1, L_2, L_3 >$
S_2	$< L_1, L_2, L_3, L_4, L_5, L_6 >$
S_3	$< L_1, L_2, L_8, L_7, L_6 >$

TABLE 5.2: A sample of sequence database.

- **Definition (Confidence)**. Let R_i: $X \Rightarrow Y$ be a sequential rule where X and Y are two sequential patterns, the confidence of R_i is calculated as follows.

$$Confidence(R_i) = \frac{Support(X \cup Y)}{support(X)}$$

(B) **Successive recommenders**. To capture both static and evolving sequential user preferences for POI recommendation, the authors in [100] have proposed a recommender system to extract sequential check-in patterns with temporal intervals between dependent POIs. Their proposed recommender builds a bi-weighted low-rank graph to learn the two individual user's behavioral preferences (static and evolving) by identifying a set of common graph bases and calculating the distributions with only sparse observations.
Another proposed approach, named *LORE* [134], uses both sequential patterns mining and Markov chains for recommendation. In *LORE*, the set of sequential patterns are first mined and then used to construct a transition graph among locations. A developed Markov chain model that considers the effect of all visited locations on recommendations is used afterward to assess the probability that a user will visit a location in the future.

(C) **Limitations**. Although sequential patterns based systems allow considering sequential influence, they have several shortfalls.

- **Sequential patterns and sequential rules**. First, sequential patterns are often misleading for the user. The reason is that patterns are found merely on the basis of their frequency (the number of sequences in which they appear). For instance, consider the sequential pattern L_2,L_3, from the SDB depicted in Table 5.2, meaning that user has visited L_2 and L_3 in that order. This sequential pattern has a support of 66,66% because it appears in sequences S_1 and S_2. However, this pattern is misleading because even though it appears in 66,66% of the sequences, there are also two sequences where L_2 is only not followed by L_3 but by L_6 (sequences S_2,S_3). Therefore, if someone had to take decisions only based on this pattern, it could lead to taking wrong decisions. A solution to this problem

would be to add a measure or probability of the confidence that a pattern will be followed. However, adding this information to sequential patterns is not that easy because they can include several locations and sequential patterns mining algorithms are not designed for that. An alternative that considers the confidence of a sequential pattern is sequential rule mining. The latter is a process that allows extracting rules of the form R_i: $X \Rightarrow Y$ where X and Y are two frequent sequential patterns. R_i indicates that if the locations in X have been visited they will be followed by visiting locations in Y. For each rule R_i, a confidence value could be associated.

- **Short-term recommendations**. Locations in a sequential pattern or a sequential rule are ordered by time but may still not represent locations that are consecutively visited by users (some locations may be ignored due to their low frequency). Therefore, recommendations based on a sequential pattern or a rule may represent locations that are far from each other and are often only visited after visiting some other places not appearing in the pattern. Thus, the recommendations may sometimes be irrelevant where a user may want to decide where to go next rather than later (i.e. short-term recommendation).
  For illustration purpose, consider the location sequences in the sequence database depicted in table 5.2 and that $minfreq = 0.66$. From these sequences, a set of frequent patterns P={(L_1), (L_2), (L_3), (L_6),(L_1,L_2),(L_2,L_3),(L_1,L_3),(L_1,L_2,L_3), (L_1,L_2,L_6)} is extracted. To recommend a location, a set of recommendation rules are derived from these sequential patterns. For example, if a user visits location L_1 and then L_2, two rules could be employed (R_1:$L_1, L_2 \Rightarrow L_3$ and R_2: $L_1, L_2 \Rightarrow L_6$) to perform recommendations based on patterns (L_1,L_2,L_3) and (L_1,L_2,L_6), respectively. These two rules have the same support value (occurrence frequency) and confidence value (conditional probability). If the rule R_2 is used, the location L_6 is recommended to the user. However, it can be argued that this is not a good recommendation since the location L_6 has never been visited directly after L_2 (the current location of the user) by other users. But L_3 was immediately visited after L_1,L_2 in two out of three sequences.
- **Order-sensitivity**. Although sequential patterns or rules are useful, they are generally too specific and restrictive in terms of the order required for visiting locations (they are order-dependent). Hence, a small variation in the order of locations visited by a user may lead to different recommendations or the inability to make recommendations. For example, tourists usually tend to visit the same famous touristic places in a city

but the sequential order may be slightly different for different tourists. Thus, recommendation using sequential patterns may fail if the order of locations visited by a tourist does not match any of the extracted sequential patterns. In the above example, if a user visits L_2 and then L_1, then no recommendation can be made based on the set of patterns P.

- **Dynamic check-in data stream**. Standard sequential patterns or sequential rules mining approaches are designed to handle static data and typically failed at handling dynamic data. But in real-life, new check-in data continuously arrive at a fast rate (a stream of data). Thus, patterns discovered in check-in data may quickly become obsolete. Since many of these approaches do not provide an update mechanism, they must be applied from scratch when new data arrives to find up-to-date patterns. Because of that, the time and space complexity of these approaches can grow exponentially when applied on dynamic data, which makes them inefficient. This is especially a problem when data is frequently updated.
  Addressing this issue requires to design incremental algorithms for mining patterns in streams of check-in data. However, mining sequential patterns in data streams is a challenging problem for both POI recommendation, and the data mining community. In literature, some incremental algorithms were proposed for mining sequential patterns [136, 137] and association rules [138, 139]. Yet, finding sequential patterns and association rules are two different problems from sequential rules mining since the latter not only consider the sequential ordering between elements of a pattern but also the pattern's frequency and confidence. Thus, the straightforward use of the existing incremental mining algorithms is not feasible. To the best of our knowledge, no algorithm was designed for incrementally discovering sequential rules that can be used to continuously identify strong relationships between check-in locations. Hence, the design of such algorithms is required.

5.5 Conclusion

In this chapter, we have provided a general overview of POI recommendation including backgrounds and literature review of different approaches to recommending POI in location-based social networks. This chapter is not intended to be a comprehensive survey, and in particular, we have only summarized a few of the approaches being proposed.
We have seen two main categories pursued in the literature: general POI

recommendation and successive recommendation. In the first category, numerous approaches have been developed, which have various characteristics. The most popular technique is collaborative filtering (CF) which exploits user ratings and preferences to perform recommendation. Generally, POI recommendation approaches should consider numerous factors, including the geographical, temporal and social influence on user mobility behavior. For this reason, many studies have extended traditional CF approaches to consider such factors. On the other hand, in the second category, we have focused on successive POI recommenders that consider the hidden sequential relationships in the check-in behavior of users. These recommenders mainly aim to suggest a set of POI recommendations to locations that a user may be interested in visiting in the near future given his current location and historical check-ins (if available). Most of these approaches use Markov chains or sequential patterns mining based models. Although these models have shown to perform well, they have several drawbacks and limitations. In sight of shortfalls of sequential patterns based recommenders, we propose in the next chapter a sequential rules mining based recommender system, named *STS-Rec*. It addresses the main drawbacks of sequential patterns mining approaches, and has the ability of considering sequential, temporal and social influence to efficiently perform recommendation.

Chapter 6

STS-Rec: Socio-Temporal and Sequential Point-of-interest Recommendation

6.1 Introduction

This chapter describes a sequential rule based recommendation system, so-called *STS-Rec* [140]. The latter addresses the main drawbacks of sequential pattern mining approaches and considers temporal and social influences to perform short-term POI recommendation. To take into account the temporal influence, *STS-Rec* adapts its mining strategy to include the temporal context in location data. Hence, the conventional rule mining problem is pulled out to mine time-extended recommendation rules. Besides, *STS-Rec* efficiently discovers patterns indicating the evolving periodic sequential behavior of human mobility by loosening the strict ordering constraint on location sequences by discovering POSR (Partially Ordered Sequential Rules) [47]. Besides, *STS-Rec* also extends the concept of sequential rules with the use of a sliding-window (a window-size constraint). This constraint allows performing short-term recommendations to suggest where people will go next rather than later. Therefore, this constraint extracts patterns of locations that appear within a maximum number of consecutive locations in location sequences.
This chapter is organized as follows. Section 2 presents a detailed description of the process of mining recommendation rules and the adopted approaches to include the influence factors. Sections 3 and 4 describe respectively the offline modeling and online recommendation modules. Section 4 presents the experimental settings and analyzes the performance of *STS-Rec*. Finally, Section 5 concludes the chapter.

6.2 Recommendation rule mining

The central idea of *STS-Rec* is to mine sequential rules from the set of location sequences, extracted from check-in data of LBSN users, and then use these rules in recommendation. For the sake of clarity, we call the process of sequential rule mining as the process of recommendation rule mining. In this

section, we first define some key terms, formulate the problem of POI recommendation and then provide a detailed description of recommendation rule mining. More specifically, we describe the mining process that discovers Standard Sequential Rules (SSR) generated from sequential patterns used in [134] and its limitations and then present our recommendation rule mining process of POSR rules. Lastly, we describe how the mining process is adapted so that it takes into account the temporal and social influences.

6.2.1 Definitions

Definition (Location sequence, location subsequence). A location sequence $S = L_i, L_{i+1}, \ldots, L_m$ is an ordered list of locations visited by a user u_i during a time period T. A location sequence $S_1 = L_{x1}, L_{x2}, \ldots, L_{xn}$ is said to be a subsequence of a sequence $S_2 = L_{y1}, L_{y2}, \ldots, L_{ym}$ (denoted as $S_1 \subseteq S_2$), if and only if there exist integers $1 < a_1 < a_2 < \ldots < a_n \leq m$ such that $L_{x1} = L_{ya1}, L_{x2} = L_{ya2}, L_{xn} = L_{yan}$.

Definition (Location history). Given the set of users U in LBSN, a location history (LH_i) of a user $u_i \in U$ contains the set of all location sequences of u_i. For example, Table 6.1 depicts a sample of visited locations and location histories for three users u_1, u_2, u_3 in three days. In this chapter, we denote as LH the location histories of all users $u_i \in U$.

Definition (Location pattern). A location pattern P_i is a subsequence of some location sequences in a location history LH. The visiting frequency of a pattern P_i, denoted as $frequency(P_i)$ is the number of times that a user has visited the locations contained in P_i in LH over the total number of sequences in LH ($|LH|$). P_i is said to be *frequent* if its frequency exceeds a user-defined parameter called *minfreq* (minimum frequency). For example, from location histories of Table 6.1, the sequences $s = L_3, L_4$ and s'=L_5, L_6 are frequent location patterns for *minfreq*=0.2.

Definition (Recommendation rule). Let L be the set of all locations in a geographic zone Z. A recommendation rule $R : P_1 \Rightarrow P_2$ is a relationship between two frequent location patterns $P_i = L_x, L_{x+1}, \ldots, L_n$ and $P_j = L_y, L_{y+1}, \ldots, L_m$ such that $P_1 \cap P_2 = \emptyset$ and $P_1, P_2 \neq \emptyset$. P_1 and P_2 are called the antecedent and consequent of R, respectively. The rule R is interpreted as if a user follows locations in P_1, he will then visit those in P_2.

Definition (Confidence of a recommendation rule and valid rule). The confidence of a recommendation rule $R : P_x \Rightarrow P_y$ is defined as follow:

$$Confidence(R) = \frac{Frequency(P_x \cup P_y)}{Frequency(P_x)} \tag{6.1}$$

It represents the conditional probability that a user will visit the locations in P_y after visiting the locations contained in P_x. A recommendation rule $R : (P_x \Rightarrow P_y)$ is said to be a valid rule if and only if $frequency(P_i \cup P_j) \geq minfreq$ and $Confidence(R) \geq minconf$ where $minfreq$ (minimum frequency) and $minconf$ (minimum confidence) are two user-defined thresholds.

User	Day	Check-in Locations	Location History	Location sequences
u_1	D_1	<u_1,8:15,L_6><u_1,10:03,L_5> <u_1,10:56,L_7><u_1,14:34,L_9>	LH_{u1}	S_{11}: L_6,L_5,L_7,L_9 S_{12}: L_1,L_2,L_3,L_4,L_7,L_{12},L_{10} S_{13}: L_3,L_4,L_7
	D_2	<u_1,06:14,L_1><u_1,7:19,L_2> <u_1,10:07,L_3><u_1,11:38,L_4> <u_1,16:09,L_7><u_1,20:02,L_{12}> <u_1,21:45,L_{10}>		
	D_3	<u_1,8:34,L_3><u_1,11:23,L_4> <u_1,22:59,L_7>		
u_2	D_1	<u_2,5:25,L_1><u_2,6:57,L_2> <u_2,09:50,L_9><u_2,11:36,L_3> <u_2,12:02,L_4><u_2,14:47,L_7> <u_2,17:38,L_6><u_2,18:03,L_{10}>	LH_{u2}	S_{21}:L_1,L_2, L_9,L_3,L_4, L_7,L_6,L_{10} S_{22}: L_4,L_3,L_7, L_9 S_{23}: L_3,L_4,L_{12}
	D_2	<u_2,09:37,L_4><u_2,11:09,L_3> <u_2,15:55,L_8><u_2,17:45,L_9>		
	D_3	<u_2,10:39,L_3><u_2,11:58,L_4> <u_2,23:02,L_{12}>		
u_3	D_1	<u_3,4:15,L_5><u_3,07:03,L_6> <u_3,12:53,L_{11}><u_3,16:03,L_7>	LH_{u3}	S_{31}: L_5,L_6,L_{11},L_7 S_{32}: L_5,L_6,L_3,L_4,L_1,L_8,L_{12}
	D_2	<u_3,5:18,L_5><u_3,07:03,L_6> <u_3,10:58,L_3><u_3,11:39,L_4> <u_3,20:20,L_1><u_3,21:35,L_8> <u_3,23:28,L_{12}>		

TABLE 6.1: Sample of location sequences and histories.

6.2.2 Problem statement (Successive POI recommendation)

Given the location histories LH of a set of users U, the list of influence factors to be considered, and the location sequence S of a user u_i (which contains his/her current location as well as its previously visited locations), point-of-interest recommendation consists of suggesting to u_i one or more locations that he is deemed likely to visit next by using the sequence S and recommendation rules learned from LH.

6.2.3 Mining recommendation rules using SSR

To generate SSR recommendation rules, the typical method is to first mine frequent location patterns (i.e. with frequency values greater than a predefined $minfreq$ threshold). From the set of frequent patterns retrieved, recommendation rules are generated if the confidence of the rule built upon a frequent pattern is greater than a predefined $minconf$ threshold. The rule is called in that case a *valid rule*. Figure 6.2 (A) depicts a sample of SSR mined based on location histories of three users (Table 6.1) for $minfreq$ = 0.2 and $minconf$= 0.4. For instance, the pattern $P_1 = L_1, L_2, L_3, L_4$ has a frequency of 0.25 (P_1 appears in 25% of location sequences). From this pattern P_1, the rule $R_1 : L_1, L_2, L_3 \Rightarrow L_4$ can be generated with a confidence value of 100%. It is interpreted as if a user u_k has visited the locations L_1 and L_2, and is currently at location L_3, he will probably visit location L_4 in the future. Thus, based on this rule, location L_4 is recommended to the user.
However, SSR rules are too precise and the imposed ordering is too strict, as each location must appear exactly in the same order in a location sequence for a rule to match with this sequence. In other words, recommendation using SSR requires that the user must visit the set of locations in the same order

as that of locations in the rule. As a result, the recommendation fails if a user visits locations in a slightly different order than the one in the recommendation rules i.e. tourists usually tend to visit the same famous touristic places in a city but the visiting order is not the same for different tourists. For example, if a user u_k has visited L_3, L_1 and then L_2, no recommendation can be made since no rule matches the visiting order of locations of u_k.

SSR
$R_1 : L_1, L_2, L_3 \Rightarrow L_4$
$R_2 : L_3, L_4 \Rightarrow L_{12}$
$R_3 : L_1, L_2, L_3, L_4 \Rightarrow L_{10}$
$R_4 : L_1, L_2, L_3, L_4 \Rightarrow L_7$
.....

(A)

POSR
$R_1 : L_1, L_2, L_3 \Rightarrow L_4$
$R_2 : L_3, L_4 \Rightarrow L_{12}$
$R_3 : L_1, L_2, L_3, L_4 \Rightarrow L_{10}$
$R_4 : L_1, L_2, L_3, L_4 \Rightarrow L_7$
$R_5 : L_3, L_4 \Rightarrow L_8$
$R_6 : L_5, L_6 \Rightarrow L_7$
.....

(B)

TABLE 6.2: A sample of some recommendation rules generated using SSR and POSR.

The strict ordering in standard sequential rules may also considerably affect how the frequency of location patterns is calculated. To determine whether a pattern is frequent, the order in a sequence must be exactly matched. Any change in the ordering results in a new pattern and therefore, two patterns that consist of the same set of locations are considered as two different patterns and used differently. Consequently, the number of frequent patterns may be decreased and therefore this may negatively affect the ability to make recommendations (i.e. recommendation coverage). For instance, the pattern $P_2 : L_3, L_4$ and $P_3 : L_4, L_3$ are two patterns containing the same set of locations but appeared in a different order. While P_2 is considered as a frequent pattern with $frequency(P_2)$=5/8; this is not the case with P_3 (i.e.$frequency(P_3)$=1/8<$minfreq$).

6.2.4 Mining partially ordered recommendation rules

To mine partially ordered recommendation rules, the *TRulegrowth* algorithm [47] is used. *TRulegrowth* takes as input the set of location sequences in a location history LH, $minfreq$ and $minconf$ threshold and outputs the set of partially ordered sequential rules (POSR).

- **Definition**(Partially-ordered recommendation rule). A partially-ordered recommendation rule $R : P_1 \Rightarrow P_2$ is a recommendation rule that represents a relationship between two unordered location patterns $P_i = L_x, L_{x+1}, \ldots, L_n$, $P_j = L_y, L_{y+1}, \ldots, L_m$. The rule R is interpreted as if a user has visited the locations in P_1 in any order, he will then visit those in P_2.

The mining process of POSR, using *TRuleGrowth* algorithm, consists of first generating the set of valid rules of 1*1 sizes (one location in each side of

the rule) and then expand these rules by applying two rule expansion procedures namely LEFTEXPAND and RIGHTEXPAND which allow discovering larger rules.
The expansion procedures intend to retrieve frequent locations that are susceptible to generate valid rules by extending the antecedent (LEFTEXPAND) or consequent (RIGHTEXPAND) of a rule with these acquired frequent locations. More details about the generation of POSR rules with the *TRuleGrowth* could be found in [47].
Yet, as we are only interested, in this research, in retrieving one location at a time for the application of POI recommendation, a modified version of *TRuleGrowth* is exploited. It simply consists of applying the LEFTEXPAND procedure while generating recommendation rules. Consequently, only the antecedents of rules starting from rules of 1*1 sizes are expanded.

In contrast to standard sequential rules, POSR rules are more flexible to order variations. For instance, let's take the POSR rule $R_1 : L_1, L_2, L_3 \Rightarrow L_4$ depicted in Figure 6.2 (B). This rule indicates that a recommendation of location L_4 can be performed when a user either visits $\langle L_1, L_2, L_3\rangle$,$\langle L_2, L_1, L_3\rangle$, $\langle L_3, L_1, L_2\rangle$, $\langle L_3, L_2, L_1\rangle$, $\langle L_1, L_3, L_2\rangle$ or $\langle L_2, L_3, L_1\rangle$. Hence the order is completely ignored in the antecedent of the rule. Using this type of rules can considerably improve recommendations as a user does not need to visit a set of locations in the antecedent of a rule in the exact same order to deduce a recommendation using that rule.
Another advantage of using POSR instead of SSR is that the frequency of location patterns is generally increased since many patterns containing the same set of locations are considered as the same location pattern. Thus, each pattern may be considered as appearing more often than the corresponding SSR, which allows generating more rules. This may increase the coverage of the system (its ability to make a recommendation). For instance, we observe in the example of Figure 6.2 that a new set of rules is obtained using the POSR mining process, that was not generated using SSR. In particular, the rules $R_5 : L_3, L_4 \Rightarrow L_8$ and $R_6 : L_5, L_6 \Rightarrow L_7$ don't appear in the set of SSR rules since both patterns L_3,L_4,L_8 and L_5,L_6,L_7 are infrequent for $minfreq = 0.2$.

Besides, unlike SSR based recommenders, POSR can be easily adapted to perform short-term recommendations. For the sake of illustration, let's suppose that a user has visited the sequence of locations L_1, L_2. Standard sequential rule mining process generates several rules that include the pattern L_1,L_2 such as the rules $R_1 : L_1, L_2 \Rightarrow L_{10}$, $R_2 : L_1, L_2 \Rightarrow L_7$, etc.
Actually, the recommendation of L_{10} (based on rule R_1) seems less relevant than L_7 (based on rule R_1). This is because L_{10} appears in all the sequences that contain L_1, L_2 as the last visited location after visiting many other places. This fact fully matches with the hypothesis adopted in several works consisting that short-term recommendation is more useful than long-term recommendation since users are more interested in recommendations to decide where they will go next rather than later. As time passes, a user's desires and

interests may change, which may affect the quality of recommendations. For instance, new markets and malls may open in an area, or a user may change his habits.

To overcome this situation and only mine rules to locations that could be visited in the near future, the concept of POSR is extended by considering a sliding window constraint. This latter allows mining only the rules from locations occurring within a sliding window, i.e. within a maximum number of consecutive locations in each location sequence. For the sake of illustration, Figure 6.3(a) shows a sample of recommendation rules generated with a windows size w=2, *minfreq*=20% ($\cong$ 2 sequences) and *minconf*=0.4 (40%), for the location histories in Table 6.1. As it can be seen, the rule $R_3 : L_1, L_2 \Rightarrow L_{10}$ is not mined since L_{10} appears as the fifth and sixth location after the last location in the antecedent of the rule (i.e. L_2) in sequences which exceed w. Using a sliding window constraint offers thus two main advantages. The first one is that it allows considering the sequential influence and therefore eliminating rules that do not satisfy this factor for recommending the next POIs. The second one is that the number of recommendation rules can be considerably reduced by discovering only those that are relevant for short-term recommendation.

(A) Before arrangement

POSR with window=2	
Rules	**Frequency/Confidence**
$R_1 : L_1 \Rightarrow L_2$	Frequency=0.25, Confidence=0.66
$R_2 : L_1 \Rightarrow L_3$	Frequency=0.25, Confidence=0.66
$R_3 : L_2 \Rightarrow L_3$	Frequency=0.25, Confidence=1
$R_4 : L_2 \Rightarrow L_4$	Frequency=0.25, Confidence=1
$R_5 : L_3 \Rightarrow L_4$	Frequency=0.62, Confidence=0.83
$R_6 : L_3 \Rightarrow L_7$	Frequency=0.37, Confidence=0.5
$R_7 : L_4 \Rightarrow L_7$	Frequency=0.37, Confidence=0.5
$R_8 : L_5 \Rightarrow L_6$	Frequency=0.25, Confidence=0.66
$R_9 : L_1, L_2 \Rightarrow L_3$	Frequency=0.25, Confidence=1
$R_{10} : L_2, L_3 \Rightarrow L_4$	Frequency=0.25, Confidence=1
$R_{11} : L_3, L_4 \Rightarrow L_7$	Frequency=0.37, Confidence=0.5

(B) After arrangement

POSR with window=2	
Rules	**Frequency/Confidence**
$R_1 : L_1, L_2 \Rightarrow L_3$	Frequency=0.25, Confidence=1
$R_2 : L_2, L_3 \Rightarrow L_4$	Frequency=0.25, Confidence=1
$R_3 : L_3, L_4 \Rightarrow L_7$	Frequency=0.37, Confidence=0.5
$R_4 : L_2 \Rightarrow L_3$	Frequency=0.25, Confidence=1
$R_5 : L_1 \Rightarrow L_3$	Frequency=0.25, Confidence=0.66
$R_6 : L_1 \Rightarrow L_2$	Frequency=0.25, Confidence=0.66
$R_7 : L_2 \Rightarrow L_4$	Frequency=0.25, Confidence=1
$R_8 : L_3 \Rightarrow L_4$	Frequency=0.62, Confidence=0.83
$R_9 : L_4 \Rightarrow L_7$	Frequency=0.37, Confidence=0.5
$R_{10} : L_3 \Rightarrow L_7$	Frequency=0.37, Confidence=0.5
$R_{11} : L_5 \Rightarrow L_6$	Frequency=0.25, Confidence=0.66

TABLE 6.3: Some recommendation rules generated using POSR with a window constraint w=2.

User	Day	Check-in Locations	Location History	Temporal Location Sequences
u_1	D_1	<u_1,8:15,L_6><u_1,10:03,L_5> <u_1,10:56,L_7><u_1,14:34,L_9>	LH_{u1}	St_{11}: <t_2>L_6,L_5,L_7,<t_3>L_9 St_{12}: <t_1>L_1,L_2,<t_2>L_3,L_4, <t_4>L_7,<t_5>L_{12},L_{10} St_{13}: <t_2>L_3,L_4,<t_5>L_7
	D_2	<u_1,06:14,L_1><u_1,7:19,L_2> <u_1,10:07,L_3><u_1,11:38,L_4> <u_1,16:09,L_7><u_1,20:02,L_{12}> <u_1,21:45,L_{10}>		
	D_3	<u_1,8:34,L_3><u_1,11:23,L_4> <u_1,22:59,L_7>		
u_2	D_1	<u_2,5:25,L_1><u_2,6:57,L_2> <u_2,09:50,L_9><u_2,11:36,L_3> <u_2,12:02,L_4><u_2,14:47,L_7> <u_2,17:38,L_6><u_2,18:03,L_{10}>	LH_{u2}	St_{21}: <t_1>L_1,L_2, <t_2>L_9,L_3, <t_3>L_4,L_7,<t_4>L_6,L_{10} St_{22}: <t_2>L_4,L_3,<t_3>L_7,<t_4>L_9 St_{23}: <t_2>L_3,L_4,<t_5>L_{12}
	D_2	<u_2,09:37,L_4><u_2,11:09,L_3> <u_2,15:55,L_8><u_2,17:45,L_9>		
	D_3	<u_2,10:39,L_3><u_2,11:58,L_4> <u_2,23:02,L_{12}>		
u_3	D_1	<u_3,4:15,L_5><u_3,07:03,L_6> <u_3,12:53,L_{11}><u_3,16:03,L_7>	LH_{u3}	St_{31}: <t_1>L_5,L_6,<t_3>L_{11}, <t_4>L_7 St_{32}: <t_1>L_5,L_6,<t_2>L_3,L_4, <t_5>L_1,L_8,L_{12}
	D_2	<u_3,5:18,L_5><u_3,07:03,L_6> <u_3,10:58,L_3><u_3,11:39,L_4> <u_3,20:20,L_1><u_3,21:35,L_8> <u_3,23:28,L_{12}>		

TABLE 6.4: Sample of temporal location sequences.

6.2.5 Sociotemporal recommendation rule mining

As previously mentioned, the recommendation is affected by temporal and social influences. Up to this point, we have only explained how to mine recommendation rules by only considering the sequential influence. In what follows, a description of the sociotemporal recommendation rule mining of *STS-Rec* is presented.

6.2.5.1 Temporal influence

To consider the temporal influence, the time information is embedded as a context in location data to finally form time-extended locations, patterns, and sequences. However, time is continuous. Mining valid rules in such data is challenging since users may rarely visit a location at the exact same hour, minute and second. Besides, using such data would considerably decrease the quality of rules found as the frequency of rules would be low as few users would visit exactly the same place at the same time. To address this issue, the time is discretized using bins. This process, called *Time Binning*, consists of dividing a day into multiple bins of equal length (e.g. [8-10h[, [10-12h[for 2-hour bins). For example, the last column of Table 6.4 depicts the set of temporal sequences generated using bins of four hours, from the check-in POI data, where t_i indicates the i^{th} time bin.

In what follows, we first redefine the problem of mining POSR rules with the existence of the temporal context and then we present our adaptation of *TRuleGrowth* that regards the temporal influence factor. Note that, in this work, we only deal with temporal information as a context. However, this same approach can be easily adapted afterward to handle other contextual

data associated with check-in data.

a) Problem statement. We define the problem of mining partially-ordered recommendation rules by considering the temporal context as being the same as the problem of mining of recommendation rules with standard POSR except that it deals with location histories of users built upon time-extended sequences (also called temporal location sequences) which incorporates the visiting time of each POI.

- ***Definition (Temporal visit, temporal location)***. A temporal visit is a set of locations visited by a user in the same time period or slot that could range from one to several hours. Formally, a temporal visit is written as Tv_i=($< t_i > L_i, L_{i+1}, \ldots L_k$) where t_i is a time slot. We denote as $Loc(Tv_i)$ the set of locations visited at $Time(Tv_i) = t_i$ in Tv_i. We also denote as a temporal location $tl_j =< t_j > L_j \in Tv_i$ if and only if $L_j \in Loc(Tv_i)$ and $t_j = Time(Tv_i)$.

- ***Definition (Temporal sequence)***. A temporal sequence $Ts =< t_i > L_i, L_{i+1}, \ldots L_k < t_j > L_j, L_{j+1}, \ldots, L_m < t_k > \ldots < t_n > L_n, L_{n+1}, \ldots, L_p$ is a set of temporal visits at different time slots.

- ***Definition (temporal subsequence, temporal pattern)***. A temporal sequence Ts_2 is said to be subsequence of Ts_1 if and only if $\forall Tv_i \in Ts_2, \exists Tv_j \in Ts_1$ where $Time(Tv_i) = Time(Tv_j)$ and $Loc(Tv_i) \subseteq Loc(Tv_j)$. Likewise standard sequential pattern definition, a temporal pattern $Pt = Tv_i, Tv_{i+1}, \ldots, Tv_{i+k}$ is defined as a subsequence of some temporal location sequences in a location history. A temporal pattern Pt_i is said to be frequent if its frequency exceeds *minfreq* threshold ($frequency(T_p) \geq minfreq$). Given the sample of temporal location sequences shown in Table 6.4, the sequences $Ts_1 =< t_2 > L_3 < t_2 > L_4$ and $Ts_2 =< t_1 > L_1, L_2 \Rightarrow < t_2 > L_3$ are frequent location patterns for $minfreq$=0.2.

- ***Definition (SID of location)***. We denote as $SID(tl_i)$ the set of location sequences that contains a temporal location tl_i thus $|SID(tl_i)|$ will represent the number of sequences in this set (i.e. The cardinality of SID(tl_i)).

- ***Definition (temporal recommendation rules)***. A recommendation rule R_t is said to be a temporal recommendation rule if it is derived from temporal patterns. More formally, R_t is denoted as $R_t : Pt_i \Rightarrow Pt_j$ where Pt_i, Pt_j are two location patterns with timestamps (i.e. temporal patterns).

b) T-POSR algorithm. To take the temporal context into account, an extension of the original *TRuleGrowth* algorithm called *T-POSR* is proposed. T-POSR initially generates rules that represent correlations between two frequent 1-pattern sizes. Then, a recursive LEFTEXPAND procedure is called to generate larger rules by extracting frequent temporal locations that are

susceptible to generate valid rules. In contrast to the original *TRuleGrowth* algorithm, T-POSR only expands the rules on their left side (antecedent) as we are interested in location recommendation using rules with one location in their consequents.

TABLE 6.5: Temporal rules in LH.

Temporal Rule	Count values
$Rt_1 :< t_1 > L_1 \Rightarrow < t_1 > L_2$	Frequency=0.25, Confidence=1
$Rt_2 :< t_1 > L_1 \Rightarrow < t_2 > L_3$	Frequency=0.25, Confidence=1
$Rt_3 :< t_1 > L_1, L_2 \Rightarrow < t_2 > L_3$	Frequency=0.25, Confidence=1
$Rt_4 :< t_1 > L_2 \Rightarrow < t_2 > L_3$	Frequency=0.25, Confidence=1
$Rt_5 :< t_1 > L_5 \Rightarrow < t_1 > L_6$	Frequency=0.37, Confidence=1
$Rt_6 :< t_2 > L_3 \Rightarrow < t_2 > L_4$	Frequency=0.5, Confidence=0.66

Figure 6.5 depicts a sample of temporal recommendation rules mined from the location data in Table 6.4. Using these rules, *STS-Rec* could both use and suggest POIs by regarding not only the locations visited by the user but also their visiting time. For instance, the rule $Rt_3 :< t_1 > L_1, L_2 \Rightarrow < t_2 > L_3$ can be read as: if a user u_k has visited L_1 in time bin t_1, and he is currently in location L_2 where a time is t_1, u_k is more likely to visit the location L_3 in time slot t_2. The pseudo-code of T-POSR and the procedure LEFTEXPAND are presented in Algorithm 1 and Algorithm 2, respectively whereas their overall description is presented in what follows.

- ***T-POSR***. The T-POSR algorithm takes as input the location histories of users and a set of user-defined parameters (*minfreq*, *minconf*, sliding *window size*, *time slot size*) and gives as output the set of valid temporal recommendation rules respecting the given constraints. As T-POSR is mainly characterized by its consideration of temporal context, our main modifications were on how to adapt the original *TRuleGrowth* so it could deal with temporal locations, patterns and sequences.

  The algorithm first starts by scanning the location history LH and computes the SID of each temporal location in LH (lines 4 to 19). The algorithm next calculates the frequency of the generated temporal locations to retain those that are candidates to construct valid rules. The temporal locations retained are the ones where their frequencies exceed the $minfreq$ threshold ($frequency = |sid(tl_i)|/|LH| \geq minfreq$) since only these temporal locations can be part of a valid rule (line 21 to 24). For each pair (tl_i,tl_j) of frequent temporal locations found, a candidate rule ($tl_i \Rightarrow tl_j$) (respectively $tl_j \Rightarrow tl_i$) of size 1*1 is then generated if and only

Algorithm 1: T-POSR (LOCATION HISTORY LH, $minfreq$, $minconf$, $window\_size$, $time\_slot\_size$)

Input:
LH: Location history
$minfreq$: minimum frequency threshold
$minconf$: minimum confidence threshold
window_size:size of sliding window constraint
time_slot_size: size of time slot
Output: The set of temporal recommendation rules

```
1  Begin
2    if (temporal_influence) then
       /* Scan the location history and find the SID of each
        temporal location in LH */
3      TLS ← ∅                          /* temporal locations set */
4      foreach Ts_i ∈ LH do
5        foreach temporal visit Tv_i ∈ Ts_i do
6          foreach Location L_i ∈ Loc(Tv_i) do
7            tl_i ←< Time(Tv_i) > L_i
8            TL_Tvi ← TL_Tvi ∪ {tl_i}
9            foreach (tl ∈ TL_Tv_i) do
10             if ( tl ∉ TLS) then
11               TLS ← TL ∪ {tl}
12               SID(tl) ← SID(tl) ∪ {Ts_i}
13             else
14               SID(tl) ← SID(tl) ∪ {Ts_i}
15             end
16           end
17         end
18       end
19     end
```

(20)
```
        /* check the frequency of each temporal location to
     construct rules */
```
(21) **foreach** $tl_i \in TLS$ **do**
(22) **if** $(\frac{|SID(tl_i)|}{|LH|} \geq minfreq)$ **then**
(23) **foreach** $tl_j \in TLS$ *and* $tl_i \neq tl_j$ **do**
(24) **if** $(\frac{|SID(tl_j)|}{|LH|} \geq minfreq)$ **then**
(25) $SID(tl_i \Rightarrow tl_j) \leftarrow \emptyset$
(26) $SID(tl_j \Rightarrow tl_i) \leftarrow \emptyset$
(27) **foreach** $ts \in (SID(tl_i) \cap SID(tl_j))$ **do**
(28) **if** $Occurence_{ts}(tl_j) - Occurence_{ts}(tl_i) > 0$ *and* $Occurence_{ts}(tl_j) - Occurence_{ts}(tl_i) + 1 \leq window\_size$ **then**
(29) $SID(tl_i \Rightarrow tl_j) \leftarrow SID(tl_i \Rightarrow tl_j) \cup \{ts\}$
(30) **if** $(Occurence_{ts}(tl_i) - Occurence_{ts}(tl_j) > 0$ & $Occurence_{ts}(tl_i) - Occurence_{ts}(tl_j) + 1 \leq window\_size)$ **then**
(31) $SID(tl_j \Rightarrow tl_i) \leftarrow SID(tl_j \Rightarrow tl_i) \cup \{ts\}$
(32) **if** $(\frac{|SID(tl_i \Rightarrow tl_j)|}{|LH|} \geq minfreq)$ **then**
(33) $LEFTEXPAND(tl_i \Rightarrow tl_j, SID(tl_i), SID(tl_i \Rightarrow tl_j), window\_size)$
(34) **if** $SID(tl_i \Rightarrow tl_j)/SID(tl_i) \geq minconf$ **then**
(35) $r \leftarrow (tl_i \Rightarrow tl_j)$
(36) $RecRules \leftarrow RecRules \cup \{r\}$
(37) **if** $(\frac{|SID(tl_j \Rightarrow tl_i)|}{|LH|}) \geq minfreq$ **then**
(38) $LEFTEXPAND(tl_j \Rightarrow tl_i, SID(tl_j), SID(tl_j \Rightarrow tl_i), window\_size)$
(39) **if** $(SID(tl_j \Rightarrow tl_i)/SID(tl_j) \geq minconf)$ **then**
(40) $r \leftarrow tl_j \Rightarrow tl_i$
(41) $RecRules \leftarrow RecRules \cup \{r\}$

if tl_i occurs before tl_j (tl_j occurs before tl_i) with the respect of sliding-window constraint (line 25 to 31). Note that $Occurrence_{Ts}(tl_i)$ is an integer that denotes the position of temporal location tl_i in the sequence Ts. For example, from temporal sequence St_{21} in Table 6.4, the occurrences of the temporal locations <t_2>L_9 and <t_4>L_{10} in St_{21} is 3 and 8, respectively.

The algorithm then finds out frequent rules from those generated where their frequencies $|\text{SID}(tl_i \Rightarrow tl_j)|/|LH|$ (respectively $|SID(tl_j \Rightarrow tl_i)|/|LH|$) is no less than $minfreq$. Once a rule ($tl_i \Rightarrow tl_j$) (respectively $tl_j \Rightarrow tl_i$) is identified as frequent, LEFTEXPAND procedure is recursively called in order to generate longer rules produced upon the obtained frequent rules (i.e.($tl_i \Rightarrow tl_j$))(lines 32,33). Finally, the confidences of the resulting frequent rules are calculated. Those with a confidence value that is not less than $minconf$ threshold and are then outputted as valid rules (lines 34 to 41).

- ***The LEFTEXPAND procedure***. LEFTEXPAND procedure takes as input a rule of the form $R : TL_i \Rightarrow TL_j$, the set of sequences that contains both patterns TL_i and TL_j and a set of thresholds ($minfreq$, $minconf$, $window\_size$, $time\_slot\_size$). LEFTEXPAND mainly seeks at finding the candidate temporal locations which could generate valid rules upon the inputted rule by expanding its antecedent. To do so, LEFTEXPAND initially searches for the sequences from the set $SID(TL_i \Rightarrow TL_j)$ to extract each temporal location tl_k where 1) $tl_k \notin TL_i$, $tl_k \neq TL_j$, and 2) tl_k occurs before the single temporal location in TL_j within the window sliding constraint in each sequence $Ts \in \text{SID}(TL_i \Rightarrow TL_j)$(lines 9 to 11). Next, as a second step, the LEFTEXPAND procedure constructs a rule with left expansion that includes the obtained candidate temporal locations tl_k and computes its frequency to determine whether it is frequent or not (i.e. $frequency(tl_k) \geq minfreq$)(lines 6 to 8). If so, the procedure then checks whether the temporal location used to expand rule (tl_k) also appears after the last occurrence of locations in TL_i within the sliding window denoted as Tli_{last} (line 13). This process is then repeated recursively, to generate larger rules by re-calling the LEFTEXPAND procedure (line 15). Finally, LEFTEXPAND outputs the expanded rules if their confidence values are greater than $minconf$ (lines 16 to 19).
  To summarize, LEFTEXPAND aims at finding and then expanding the antecedent of rules with temporal locations which 1) aren't contained in TL_i and TL_j, 2) occur before the first occurrence of the temporal location in TL_j and after the last occurrence of TL_i within the sliding window and finally outputs the valid expanded rules (frequents and their confidences exceed $minconf$ threshold).

Algorithm 2: LEFTEXPAND($(TL_i \Rightarrow TL_j), SID(TL_i), SID(TL_i \Rightarrow TL_j)$,WINDOW_SIZE)

$TC \leftarrow \emptyset$ /* the set of possible temporal locations to be used for left extension */

$TLi_{last} \leftarrow$ last temporal location in TL_i

foreach $Ts \in SID(TL_i \Rightarrow TL_j)$ **do**

  foreach $tl_k \in Ts$ **do**

    /* If existed a temporal location tl_k that occurs after the last pattern of TL_j */

    if $(Occurence_{Ts}(tl_{j0}) - Occurence_{Ts}(tl_k) > 0$ *and* $Occurence_{Ts}(tl_{j0})$- $Occurence_{Ts}(tl_k) + 1 < window\_size$ *and* $tl_k \notin TL_i \cup TL_j)$ **then**

      $SID(TL_i \cup \{tl_k\}) \leftarrow SID(TL_i \cup \{tl_k\}) \cup Ts$

      $TC \leftarrow TC \cup tl_k$

foreach $tl_k \in TC$ **do**

  if $(\frac{(SID(TL_i \cup \{tl_k\} \Rightarrow TL_j))}{|LH|} \geq minfreq)$ **then**

    $SID(TL_i \cup \{tl_k\}) \leftarrow \emptyset$

    foreach $sid \in SID(TL_i)$ *and* $sid \in SID(tl_k)$ **do**

      /* Check whether the temporal location occurs after the last temporal location TLi_{last} in the antecedent of the rule */

      if $(Occurence_{sid}(tl_k) - Occurence_{sid(TLi_{last})} >$ 0 and $Occurence_{sid}(tl_k) - Occurence_{sid}(Tli_{last}) + 1 <$ $window\_size)$ **then**

        $SID(TL_i \cup \{tl_k\}) \leftarrow SID(\mathrm{TL}_i \cup \{tl_k\}) \cup sid$

    LEFTEXPAND($TL_i \cup \{tl_k\} \Rightarrow$ TL_j,$SID(TL_i \cup \{tl_k\}), SID(TL_i \cup \{tl_k\} \Rightarrow Tl_j)$)

    if $(\frac{|SID(TL_i \cup \{tl_k\} \Rightarrow TL_j)|}{|SID(TL_i \cup \{tl_k\})|} \geq minconf)$ **then**

      $r = TL_i \cup \{tl_k\} \Rightarrow TL_j$

      $RecRules \leftarrow RecRules \cup \{r\}$

      $RecRules \leftarrow RecRules \cup \{r\}$

return *RecRules*

6.2.5.2 Social influence

Based on the assumption that friends have similar preferences, interests and may thus visit the same locations, social influence is an essential factor in POI recommendation.

Definition (Social matrix). Let there be a set of users U. A Social Matrix SM stores the social relationships between users. In this matrix, the entries s_{ij} and s_{ji} are set to 1 if the user u_j appears in the list of friends of u_i. Otherwise, they are set to 0.

To adapt *STS-Rec* so it considers this influence factor, the social matrix SM is used. STS-Rec first extracts the list of friends of a given user u_i by scanning SM and then mines recommendation rules from location histories of u_i and his friends instead of the whole location histories of all users. Considering this factor allows hence to decrease the amount of data that needs to be processed to make recommendations which consequently increases the speed of *STS-Rec*.

6.3 System architecture

In this section, we present the architecture of our *STS-Rec* rule-based recommender. As depicted in Figure 6.1, *STS-Rec* operates in two steps: (1) Offline Modeling and (2) Online Recommendation. The proposed system first generates the set of location histories and then mines sequential rules that are thereafter in recommendation.

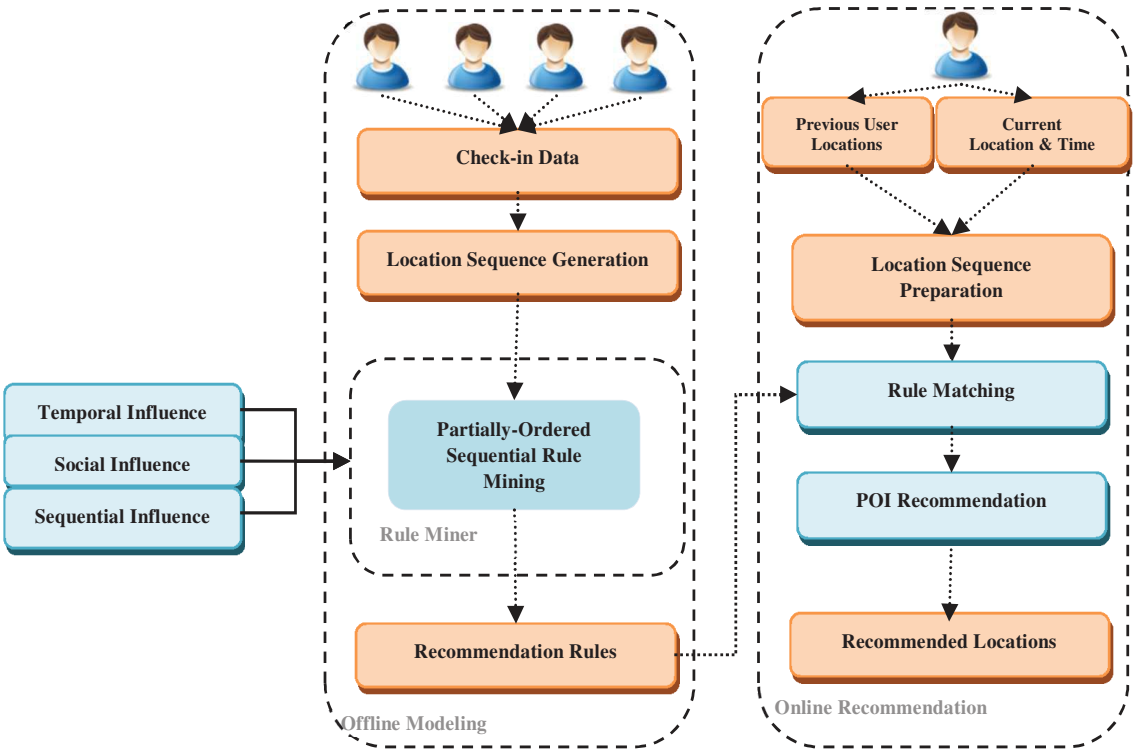


FIGURE 6.1: STS-Rec system architecture.

6.3.1 Offline modeling

It is performed by two modules: a) Location sequence generation and b) Recommendation rule mining.

6.3.1.1 Location sequence generation

Given a set of users $U = \{u_1, u_2, \ldots, u_n\}$ and a set of check-in locations of U, this module generates the location history LH_i of each user. The LH_i of a given user u_i contains the set of location sequences of venues visited each day (i.e. time period T=one day). *STS-Rec* principally adopts this period (a day) since most people exhibit strong periodic mobility behavior throughout this period. To regard the temporal influence, the check-in time of each location is introduced into each location sequence (i.e. time-extended sequence). In this step, the social matrix is also constructed which models user-to-user social relationships and allows the a lookup for the friends of any user.

6.3.1.2 Recommendation rule mining

This module is the core of the proposed system. It considers several factors and mines sequential rules. To do so, this module takes as input the set of location sequences issued from the previous step and the influence factor considered and mines recommendation rules according to the description provided above.

6.3.2 Online Recommendation

The online recommendation module consists of three components 1) Location sequence preparation 2) Rule matching, and 3) POI recommendation.

6.3.2.1 Location sequence preparation

It consists of collecting the current location L_c of a user u_i and its previously visited locations if available, to construct the current location sequence $CL = L_i, L_{i+1}, \ldots, L_c$ of u_i. In the case where the visiting time of location in CL is also available, locations in CL are also annotated with their visiting time to create time-extended locations and CL is written thus as $CL = < t_i > L_i, < t_{i+1} > L_{i+1}, \ldots, < t_c > L_c$.

6.3.2.2 Rule Matching

This component scans the recommendation rules obtained from the offline modeling step to find those that match CL. More formally, a rule R_k is said to match with CL, if and only if CL is a subsequence of the antecedent of R_k

Dataset	Users	POI	Social Links	Period
Gowalla	196.591	1.280.969	950.327	Feb. 2009 - Oct. 2010
Brightkite	58.228	772.965	214.078	Apr. 2008 - Oct. 2010

TABLE 6.6: Datasets' statistics.

($CL \subseteq Antecedent(R_k)$). As mentioned earlier, the proposed system can tolerate order variations. Therefore, the set of locations in CL does not need to appear in the same order in the antecedent of R_k. This is different from Markov-based recommenders such as n-order Markov, which requires a strict ordering. In fact, the proposed matching process offers more flexibility to allow order variation, and tolerate some missing locations (gaps) from a sequence in a rule.

6.3.3 POI Recommendation

Once the set of matching rules is found, the recommendation component selects the top-N rules having the highest confidence. Then, it retrieves their consequents as POIs that the user may be interested in visiting.

6.4 Experimental study

This section describes the evaluation of the proposed *STS-Rec* recommender. The section first illustrates the experimental setting, then presents the conducted experiments, and discusses the obtained results.

6.4.1 Experimental setting

To evaluate $STS - Rec$, our experiments were carried out on a test environment made of an Intel i5- CPU 2.4GHz processor with 6GB of available RAM on a Windows 10 operating system and 500GB of Hard Disk.

Datasets. Experiments were performed on two public large-scale check-in datasets from Gowalla [141] and Brightkite [142]. These two datasets have been widely used for location recommendation and prediction (e.g. see [37]). Table 6.6 summarizes the datasets' characteristics. In these datasets, a check-in is described by a userID, LocationID, Location-Latitude, Location-Longitude, and timestamp. In our experiments, a subset of check-in data of 1000 users was used. Each dataset was split into a training set and a testing set based on a training ratio parameter. From the set of check-in data of users, the location histories were generated, each composed of several daily sequences. In each sequence, the repeated consecutive locations were removed. The reason is that such repetition is not a transition between two locations.

Metrics To evaluate the performance of the proposed recommender, two well-known and well-used measures were employed: accuracy and coverage.

- **Overall Accuracy**. It is the number of POIs recommended and actually visited by users divided by the total number of sequences in the testing set.

$$Accuracy = \frac{\text{Number of successful recommendations}}{\text{Number of testing sequences}}$$

- **Coverage**. It is the number of testing sequences where a recommendation has been done (i.e. at least one matching rule was found), divided by the total number of testing sequences.

$$Coverage = \frac{\text{Number of recommendations}}{\text{Number of testing sequences}}$$

6.4.2 Compared Models

Our proposed *STS-Rec* recommender system which use POSR, denoted as STS-Rec(POSR), was compared with the following state-of-the-art sequential-based models:

- SSR-Rec (Standard Sequential Rules Recommender). This model generates recommendation rules upon sequential patterns proposed in [37].
- SSRW-Rec (Standard Sequential Rules with window Recommender). This recommendation system mines standard sequential rules and applies a window size constraint to only keep rules for a short-term recommendation.
- FOM-Rec (First Order Markov Recommender). This recommender utilizes first-order Markov chain which only uses the last location visited by users to predict the next location [36, 39].
- NOM-Rec (N-Order Markov Recommender). This model employs n-order Markov chains where n is set to 5 [134].

6.4.3 Experiments

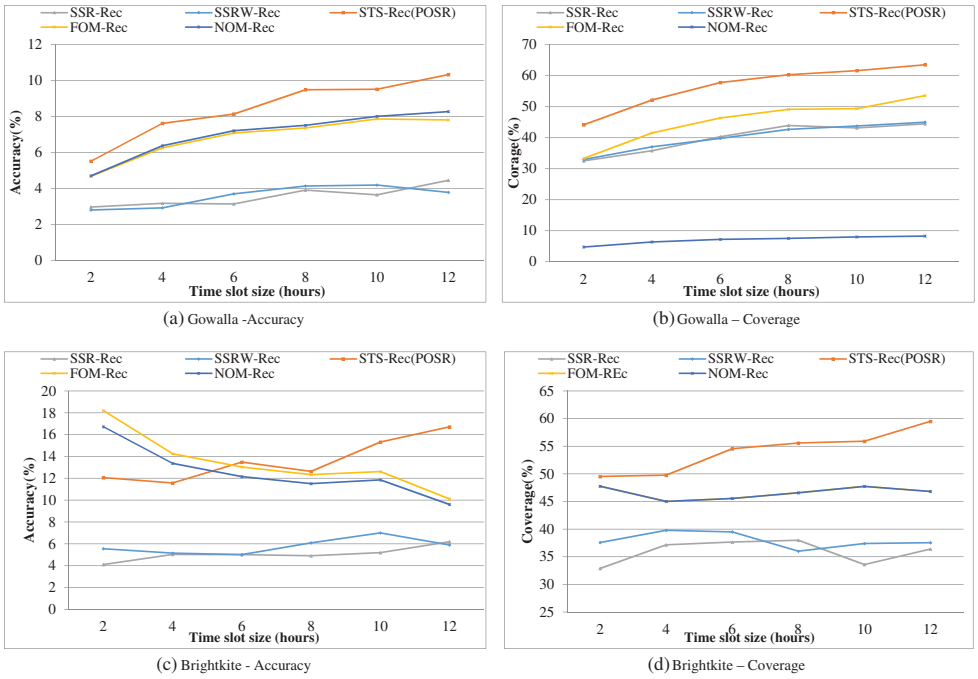
6.4.3.1 Parameter Effect

This set of experiments assesses the influence of varying the parameters: *time slot size*, $minfreq$, *training ratio*, *windowsize*, and $minconf$ on recommendation quality.

6.4.3.1.1 Impact of varying the time slot size

In this experiment, the size of time slots (bins), used to construct temporal sequences, was varied. As mentioned earlier, each day is divided into several time periods of a predefined length. From the results depicted in Figure 6.2, it can be observed, specifically with *Gowalla*, that recommendation quality

is improved when the size of time slots is increased. The reason is that by increasing the time slot length, more temporal flexibility is provided. In other words, the user has more time to visit the location visited by other users and that have been suggested by the recommendation systems. By analyzing and observing the check-in activity of users, it is obvious that each person has his own preferences and habits. Therefore, a user may visit a location, but not necessary in the time slots of recommendation rules. Consequently, the time bin that provides the best accuracy is the largest one (time slot size=12 in the experiment), which provides few restrictions on the time of a user's visits.



(a) Gowalla -Accuracy

(b) Gowalla – Coverage

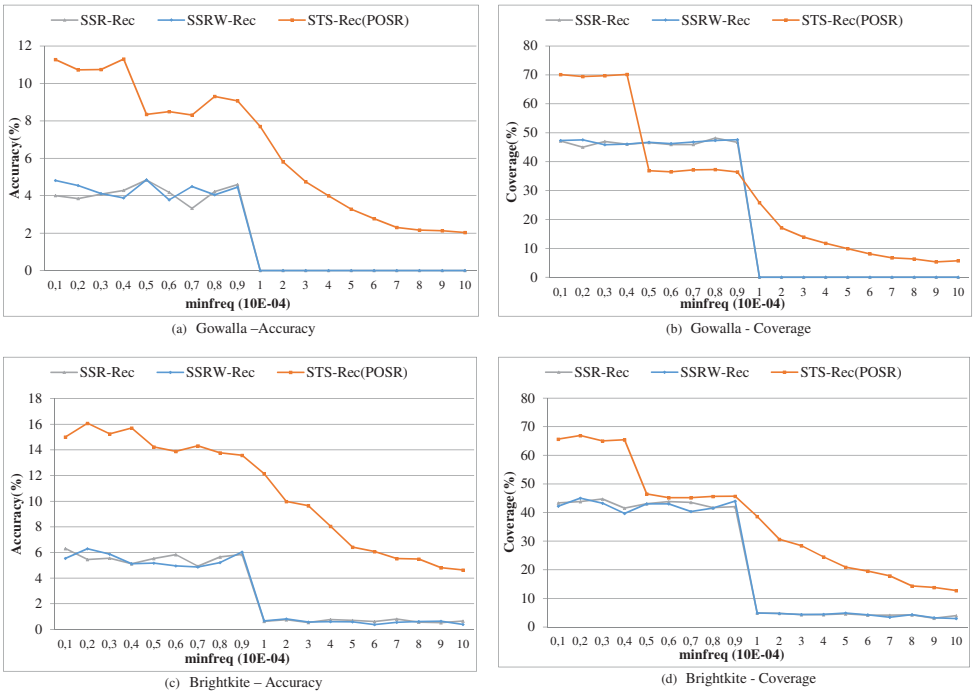
(c) Brightkite - Accuracy

(d) Brightkite – Coverage

FIGURE 6.2: Impact of varying time slot length on recommendation.

6.4.3.1.2 Impact of varying the minfreq threshold

In Figure 6.3, the influence of the $minfreq$ parameter on recommendation quality was evaluated. In this experiment, Markov based models were not evalutated since they are not influenced by this parameter. For all sequential rule mining based models, as $minfreq$ is increased, results become worse. SSRW-Rec and SSR-Rec models are unable to make recommendations for $minfreq$ values greater than 0.001 whereas STS-Rec(POSR) still provides recommendations for that value. The reason is that by increasing $minfreq$, less frequent patterns are found whereas STS-Rec(POSR) still gives good results as POSR tolerates order variations in location patterns since many patterns containing the same set of locations but differently ordered are considered as the same location pattern which boosts the frequency of location patterns.



(a) Gowalla –Accuracy (b) Gowalla - Coverage

(c) Brightkite – Accuracy (d) Brightkite - Coverage

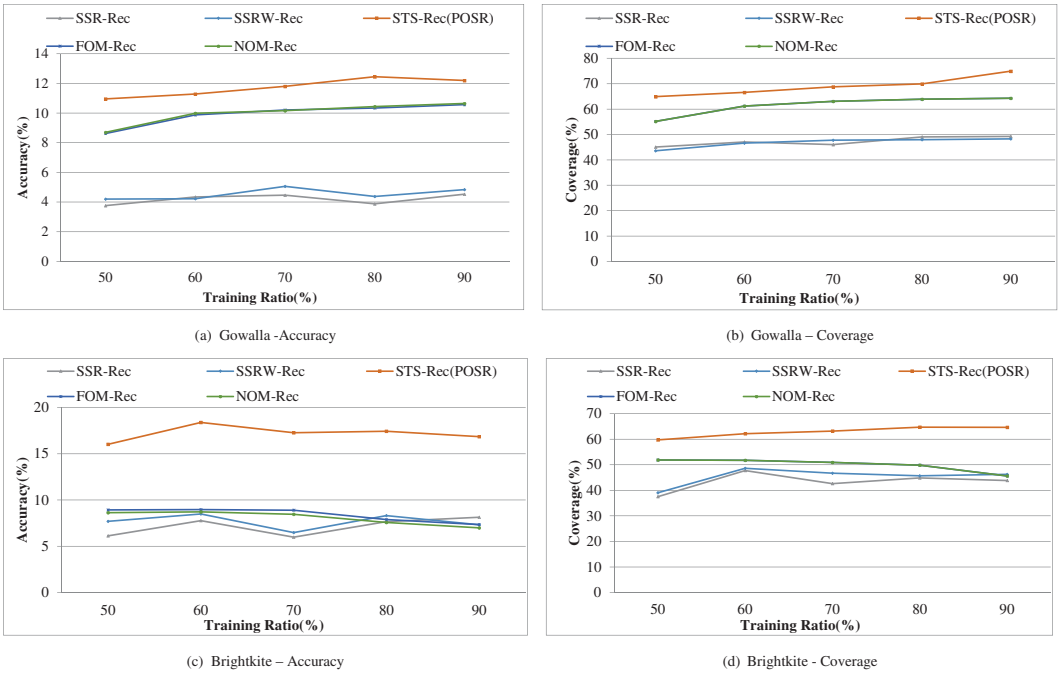
FIGURE 6.3: Impact of varying $minfreq$ threshold on recommendation.

6.4.3.1.3 Impact of varying the training ratio

In Figure 6.4, we examine the effect of varying the training ratio on recommendation. As expected, as the training ratio is increased, more data are used to build the recommender. Consequently, more accurate recommendation are obtained as more information about the check-in activity of users is utilized.

6.4.3.1.4 Impact of varying the window size

In Figure 6.5, we evaluate the impact of the window size on recommendation performance. In this experiment, the window size was varied from 2 to 9. Increasing the window size by 1 means that the recommendation process allows generating patterns that appear in the previous window and also in the next location. Thus, the set of patterns discovered using a window of size w contains those generated using a window size of $w-1$. Results indicate that setting the window size to 2 or 3 gives the best results and that by further increasing the window size, no improvement is obtained. This observation confirms our assumption that short-term recommendation is more relevant to LBSN operators and users than long term recommendations. In fact, a user may be more interested in short-term recommendations as he may want to decide where to go next rather than later. Again, Markov-based recommenders are not evaluated in this experiment as they don't use the window size parameter.



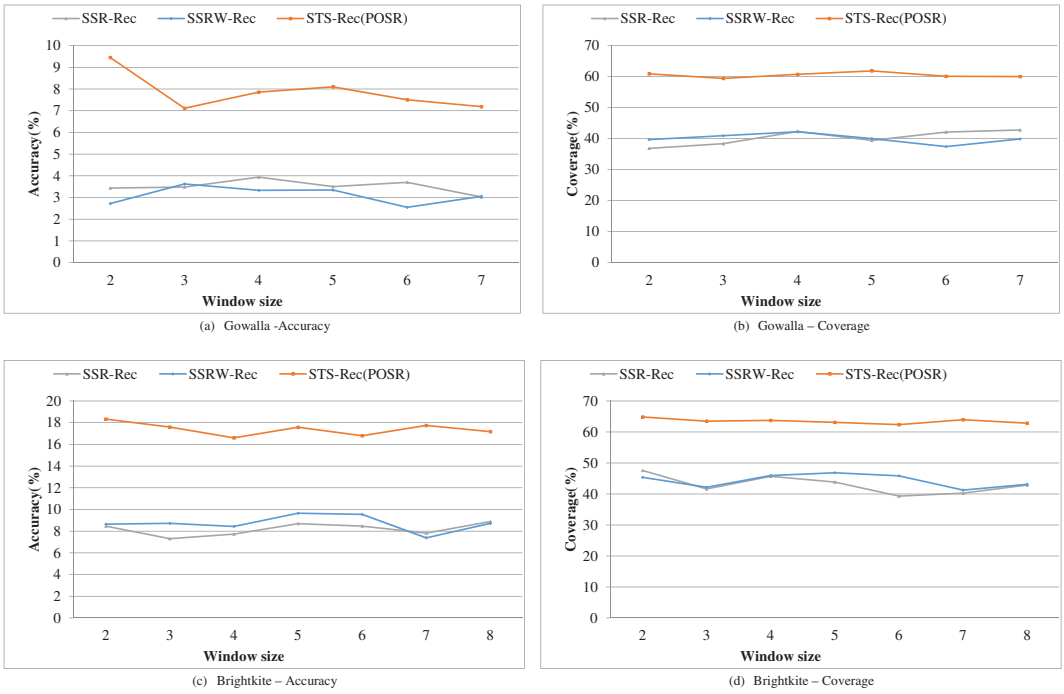
(a) Gowalla -Accuracy

(b) Gowalla – Coverage

(c) Brightkite – Accuracy

(d) Brightkite - Coverage

FIGURE 6.4: Impact of varying training ratio on recommendation.



(a) Gowalla -Accuracy

(b) Gowalla – Coverage

(c) Brightkite – Accuracy

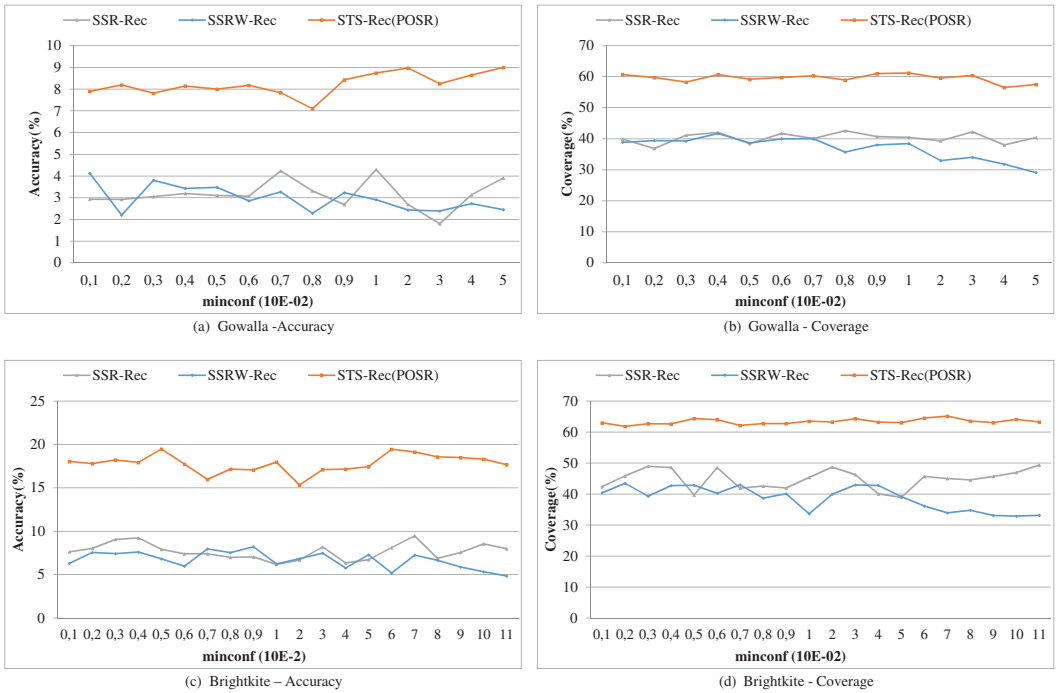
(d) Brightkite – Coverage

FIGURE 6.5: Impact of varying window size on recommendation.

6.4.3.1.5 Impact of varying the *minconf* threshold

Figure 6.6 verifies the influence of varying *minconf* on performance. From the results, it is observed that increasing *minconf* hasn't a significant effect

on the performance for all sequential rule mining recommenders.



(a) Gowalla -Accuracy
(b) Gowalla - Coverage
(c) Brightkite – Accuracy
(d) Brightkite - Coverage

FIGURE 6.6: Impact of varying *minconf* threshold on recommendation.

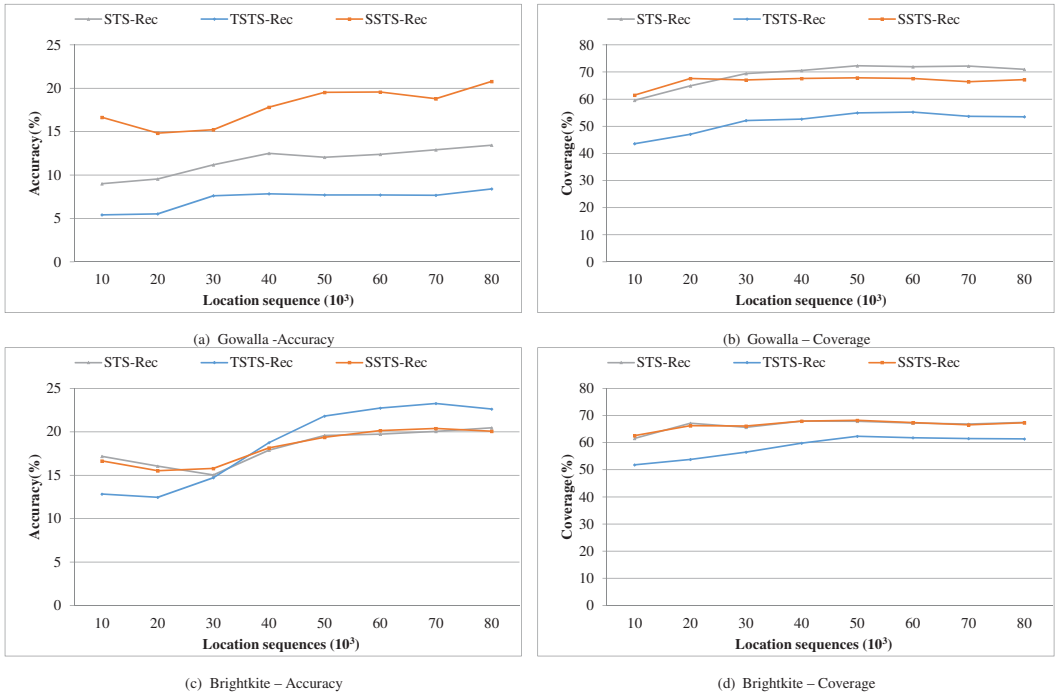
6.4.3.2 Model Comparison

The contribution of social and temporal influence on the performance of the proposed *STS-Rec* recommender was also evaluated. The results are shown in Figure 6.7. Three versions of *STS-Rec* are compared: 1) STS-Rec is the basic version that only considers sequential influence using POSR, 2) TSTS-Rec considers both sequential and temporal influences in the check-in activity, and 3) SSTS-Rec considers sequential and social influence. This experiment was also carried out to study the scalability of the proposed recommenders by increasing the number of location sequences considered.
From the obtained results, it is found that recommendation performance is improved when the number of location sequences for training is increased. This is reasonable because as more location data is used, the more can be learned from this data.
The results also show that SSTS-Rec outperforms TSTS-Rec and STS-Rec (with Gowalla). SSTS-Rec has better performance because it considers social relationships, and persons generally tend to visit venues suggested or previously visited by their friends. Another observation, not shown in Figure 6.7, is that SSTS-Rec is considerably faster and requires less storage than TSTS-Rec and STS-Rec. This is because SSTS-Rec mines recommendation rules using the

location data of subsets of users (the user and his friends) instead of considering the data of all users.
However, even though considering social influence allows obtaining better results, there are some limitations of using social influence in some cases (with Brightkite). In particular, friends on social networks sometimes do not have a real friendship relation (e.g. do not have physical interactions and may even live in different countries). Therefore, visiting locations suggested by friends is not always possible. Furthermore, although friends often share common interests, they may not always have identical behavior and preferences. Hence, differentiation in visited venues may occur.
Besides, results show that TSTS-Rec has limited performance compared to *STS-Rec* and SSTS-Rec (with Gowalla). The reason is that although human mobility is characterized by its temporal regularity, each person still has his own habits. A given person may visit the same set of locations already visited by other persons but not necessarily at the same time or in the same order. For example, a person may visit a popular shopping mall just like some other people but he may not visit it at exactly the same time.
Besides, the lower performance of TSTS-Rec is also due to the frequencies of temporal patterns that are generally less than those of standard location patterns. By extending location sequences to consider temporal data, the frequencies of location patterns decreases and thus, some rules derived from these patterns may not be discovered for the same *minfreq* threshold. This leads to mine fewer recommendation rules and this affects the recommendation coverage. Note that although the number of recommendation rules mined by TSTS-Rec is less than those mined with *STS-Rec*, TSTS-Rec rules are more precise and more relevant. They allow, in addition to recommending future location, to predict the expected visiting time which is not possible with the basic *STS-Rec* system. Furthermore, the lower performance of TSTS-Rec can also be explained by the matching process applied while recommending locations. Matching temporal recommendation rules with a user's previously visited locations and with their corresponding time is too restrictive. It essentially requires that both parts (temporal and visited locations) in the check-in activity of a user appear as the antecedent of at least one temporal rule to make a recommendation. To overcome this situation, it is advisable to use lower *minfreq* threshold values than those used for *STS-Rec* so that more temporal patterns can be learned when using TSTS-Rec or smooth the similarity measure and extend it to also consider patterns from closer time slots.

6.4.3.3 STS-Rec vs other recommenders

Figure 6.8 compares the recommendation quality of the proposed recommender with state-of-the-art-models. Results show that *STS-Rec* outperforms state-of-the-art Markov and sequential patterns mining based recommendation systems for both datasets for all metrics. This demonstrates that our recommender better captures the sequential patterns in check-in data than



(a) Gowalla -Accuracy

(b) Gowalla – Coverage

(c) Brightkite – Accuracy

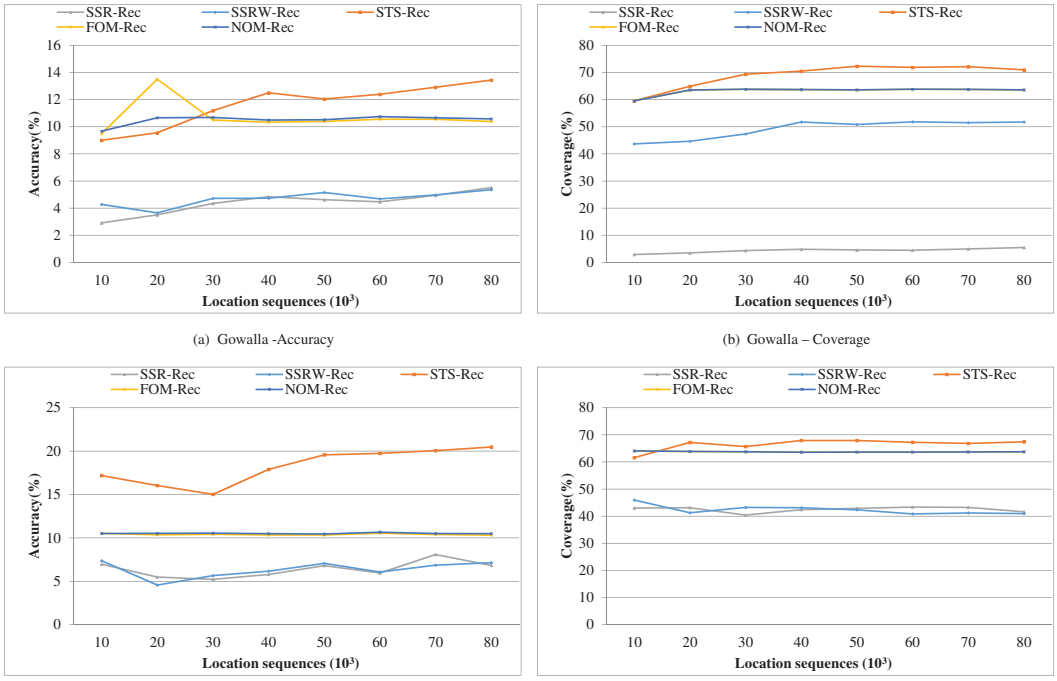
(d) Brightkite – Coverage

FIGURE 6.7: Impact of influence factors on STS-Rec.

other sequential state-of-the-art models. *STS-Rec* can provide the best accuracy (greater than 9%) and coverage (exceeds 60%) in all situations for both datasets. The reason is that order variations are allowed, which 1) removes restrictions on the order of visited locations by users, 2) increases the frequency of sequential patterns, and 3) allows generating fewer but more relevant rules (one POSR can replace many SSR).

Moreover, it can be observed from the obtained results that recommendation accuracy is better for Brightkite compared to Gowalla. This can be explained by the fact that the density of check-in locations in Gowalla is lower than Brightkite. Therefore, the seqeunces generated from this dataset is of smaller size compared to Brightkite which consequently reduces the sequential influence of the visited POIs.

To summarize, the conducted experiments have shown that the proposed recommender exhibits promising results that outperform a set of state-of-the-art models. The *STS-Rec* recommender based on POSR is mainly influenced by the setting of $minfreq$, training ratio and time slot parameters. Moreover, experiments also demonstrate the improvements in accuracy obtained by focusing on short term recommendations due to the changing user preferences and interests. Finally, it is important to mention that the accuracy of location recommendation systems for LBSNs are usually not very high, because of the low density of POIs in check-in data. Therefore, it is advisable to use more dense location data.



(a) Gowalla -Accuracy

(b) Gowalla – Coverage

(c) Brightkite – Accuracy

(d) Brightkite – Coverage

FIGURE 6.8: STS-Rec vs other recommenders.

6.5 Conclusion

This chapter has proposed a recommender named *STS-Rec* that considers the sequential, social and temporal influences on mobility behavior for POI recommendation. *STS-Rec* initially extracts recommendation rules using a new type of sequential rules miner and then utilizes these rules to suggest new venues to users. Unlike standard sequential-based recommenders, the proposed approach does not require strict ordering of locations in learned rules which leads to a more accurate recommendations. An extensive experimental study was carried out using two real LBSN datasets. Results have shown that the proposed recommender achieves higher performance compared to state-of-the-art models. For future work, we plan to extend *STS-Rec* by following two research directions: a) considering the spatial influence in recommendation, b) removing redundancy in recommendation rules, and c)designing an incremental recommender that handles the continuous check-in data stream.

Conclusion

Mobility prediction and POI recommendation are essential location based services that have been widely implemented and integrated in various real-world applications. Mobility prediction is a key feature in mobility management in smart cities, intelligent transportation systems and mobile computing environments. It has been implicitly or explicitly utilized for transportation scheduling, urban regulation, traffic congestion reduction and fuel consumption optimization. As such, POI recommendation, which consists of suggesting locations to a user that he may be interested in visiting, aim at helping mobile users discovering interesting new locations (e.g. restaurants and stores), while on the move. Effective POI Recommendation is beneficial to both users and LBSN providers. It lets users benefit from the previous experiences of other users (in particular, friends) to decide whether a location is interesting and should be visited. In general, location recommendation is most useful to users who are visiting unfamiliar areas. For LBSN providers, an economic profit is made by advertising locations to potential visitors.

In this thesis, we mainly focused on how to build effective models for route prediction and POI recommendation in LBSNs. Therefore, we have proposed three models: two route predictors and one POI recommender. For route prediction, NextRoute and MyRoute predictors have been proposed. Both models are designed to accommodate and fulfill the challenges of noise tolerance, personalized and global prediction, and lossless prediction. While NextRoute mainly aims at building a lossless prediction model and cope with noisy mobility data, MyRoute adopts an efficient graph structure to perform real-time route prediction.

Regarding POI recommendation, STS-Rec model is proposed. This latter considers the sequential, social and temporal influences of the check-in activity for POI recommendation. STS-Rec initially mines recommendation rules using a new type of sequential rules miner and then employs these rules to suggest new venues to users. Unlike other sequential-based recommenders, the proposed approach does not require a strict ordering of locations in learned rules, which leads therefore to more accurate recommendations.

Generally speaking, results of the experimental evaluation of our proposals (NextRoute, MyRoute, STS-Rec), with real-world and synthetic datasets, have shown that the proposed models achieve a much better performance compared to sequential state-of-the-art models in terms of accuracy and prediction coverage. Although, our proposals for route prediction and POI recommendation have shown excellent results and performance, it is worth to insist on some notes and possible enhancements regarding:

1. The consideration of additional contextual data related to person mobility that may also affect the accuracy by including temporal factors (e.g., time-of-day, day-of-week) and other behavioral or traffic details such as traffic volume for upcoming routes where a person may take another route if his usual route is congested. Therefore, it will become possible to distinguish between trips relying not only on spatial data but also time and other contextual data (e.g. weather, traffic congestion level).

2. The extension of STS-Rec by following three research directions:

    - Removing redundancy in recommendation rules and discovering a more compact set of recommendation rules.
    - Considering the spatial influence in recommendation.
    - Designing an incremental recommendation rules miner to handle the continuous check-in data stream, will avoid rules mining from scratch when new check-in data arrive which allows thus performing up-to-date recommendations.

3. The extension of experimental evaluation in order to:

    - Compare our proposals against other approaches such as those based on Context Tree Weighting and Neural Networks.
    - Utilize mobility data of vehicles used for personal purposes rather than those used for work such as taxicabs in order to achieve better realistic insight on people mobility.

4. The pre-processing phase, in route prediction, where prediction accuracy is mainly affected by data collection and preparation steps such as map-matching. This latter is a challenging task and remains an open research axis where no optimal map-matching algorithm could be found. Thus, the concentration on the design or the use of efficient map-matching methods is essential to ensure a reasonable quality of prediction outcome.

Glossary

POI	Point of Interest
LBSN	Location based social network
CRT	Compact rule tree
CPT	Compact Prediction Tree
DG	Dependency graph
SR	Sequential rule
POSR	Partially ordered sequenential rule
LBS	Location based services
ITS	Intelligent Transportation Systems
Minsup	Minimum support
Minconf	Minimum confidence
FSC	Frequent Subsequence Compression
SBC	Simple Branches Compression
GPS	Global Positioning System
GM	Global Model
PM	Personal Model
GMG	Global Mobility Graph
PMG	Personal Mobility Graph
LH	Location History
SUMO	Simulation of Urban MObility
XML	Extended Markup language
PPM	Prediction by Partial Matching
HMM	Hidden Markov model
PST	probabilistic Suffix Trees
RN	Road Netwrok
VMM	Variable Order Markov Model
LZ	Lempel-Ziv
DM	Data mining
CRPM	Continuous Route Pattern Mining
ANN	Artificial neural networks
LDA	landmark discovery algorithm
PT	Prediction Tree
II	Inverted Index
LT	LookupTable
DCF	Subsequence Dictionary
RD	Recursive Divider
PNR	Prediction with improved Noise Reduction
CT	Count Table

LuST	Luxembourg SUMO Traffic
MG	mobility graph
CF	collaborative filtering
MGM	Multi-center Gaussian Model
KDE	kernel density estimation
SM	Social Matrix
TLS	Temporal Location sequence
SRR	Sequential recommendation rules

List of Publications

International Journals

- Amirat, Hanane & Lagraa, Nasreddine & Fournier Viger, Philippe & Ouinten, Youcef. (2019). NextRoute: a lossless model for accurate mobility prediction. Journal of Ambient Intelligence and Humanized Computing. 10.1007/s12652-019-01327-w.
- Amirat, H. & Lagraa, Nasreddine & Fournier Viger, Philippe & Ouinten, Youcef. (2017). Myroute: A graph-dependency based model for real-time route prediction. Journal of Communications. 12. 668-676. 10.12720/jcm.12.12.668-676.

International Conferences

- Amirat, Hanane & Benslimane, Abderrahim & Fournier Viger, Philippe & Lagraa, Nasreddine. (2018). LocRec: Rule-Based Successive Location Recommendation in LBSN. 1-6. 10.1109/ICC.2018.8422183.
- Amirat, H. & Lagraa, Nasreddine & Fournier Viger, Philippe & Ouinten, Youcef. (2017). Myroute: A graph-dependency based model for real-time route prediction, ICNIT 2017.
- Amirat, Hanane & Lagraa, Nasreddine & Kerrache, Chaker & Ouinten, Youcef. (2018). Fuzzy Clustering for Misbehaviour Detection in VANET. 200-204. 10.1109, SaCoNeT.2018.8585454.

Bibliography

[1] Foursquare. L'entreprise de référence dans le domaine des données et services de localisation. https://fr.foursquare.com/. (Accessed on 12/25/2019).

[2] Places. Google maps platform | google cloud. https://cloud.google.com/maps-platform/places/?hl=fr. (Accessed on 23/12/2019).

[3] yelp. Restaurants, dentists, beauty salons, doctors - yelp. https://www.yelp.com/. (Accessed on 12/25/2019).

[4] Shenglin Zhao, Michael R. Lyu, and Irwin King. *Point-of-Interest Recommendation in Location-Based Social Networks*. SpringerBriefs in Computer Science. Springer Singapore, Singapore, 2018.

[5] Yoshitaka Deguchi, Kouichi Kuroda, Makoto Shouji, and Taketoshi Kawabe. HEV Charge / Discharge Control System Based on Navigation Information. In *Convergence International Congress & Exposition On Transportation Electronics*, volume 1, 2004.

[6] Yingzi Wang, Nicholas Jing Yuan, Defu Lian, and Linli Xu. Regularity and Conformity : Location Prediction Using Heterogeneous Mobility Data. 2015.

[7] G Bejerano and G Yona. Variations on probabilistic suffix trees: statistical modeling and prediction of protein families. *Bioinformatics (Oxford, England)*, 17(1):23–43, jan 2001.

[8] Andrew McCallum, Dayne Freitag, and Fernando Pereira. Maximum Entropy Markov Models for Information Extraction and Segmentation. *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 591–598, 2000.

[9] Mukund Deshpande and George Karypis. Selective Markov models for predicting Web page accesses. *ACM Transactions on Internet Technology*, 4(2):163–184, may 2004.

[10] 41+ must know foursquare statistics in 2020. https://review42.com/foursquare-statistics/. (Accessed on 05/30/2020).

[11] Bin Liu, Yanjie Fu, Zijun Yao, and Hui Xiong. Learning geographical preferences for point-of-interest recommendation. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and*

data mining - KDD '13, page 1043, New York, New York, USA, 2013. ACM Press.

[12] Jean-Benoit Griesner, Talel Abdessalem, and Hubert Naacke. POI Recommendation: Towards Fused Matrix Factorization with Geographical and Temporal Influences. *Proceedings of the 9th ACM Conference on Recommender Systems*, pages 301–304, 2015.

[13] Jia-dong Zhang, Chi-yin Chow, and Yanhua Li. iGeoRec : A Personalized and Efficient Geographical Location Recommendation Framework. 1374(c):1–14, 2014.

[14] Josh Jia-ching Ying, Wang-Chien Lee, and Vincent S. Tseng. Mining geographic-temporal-semantic patterns in trajectories for location prediction. *ACM Transactions on Intelligent Systems and Technology*, 5(1):1–33, 2013.

[15] Mao Ye, Peifeng Yin, Wang-Chien Lee, and Dik-Lun Lee. Exploiting geographical influence for collaborative point-of-interest recommendation. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information - SIGIR '11*, page 325, New York, New York, USA, 2011. ACM Press.

[16] Quan Yuan, Gao Cong, Zongyang Ma, Aixin Sun, and Nadia Magnenat Thalmann. Time-aware point-of-interest recommendation. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval - SIGIR '13*, page 363, New York, New York, USA, 2013. ACM Press.

[17] Huiji Gao, Jiliang Tang, Xia Hu, and Huan Liu. Exploring temporal effects for location recommendation on location-based social networks. In *Proceedings of the 7th ACM conference on Recommender systems - RecSys '13*, pages 93–100, New York, New York, USA, 2013. ACM Press.

[18] Jia-dong Zhang and Chi-Yin Chow. TICRec: A Probabilistic Framework to Utilize Temporal Influence Correlations for Time-Aware Location Recommendations. *IEEE Transactions on Services Computing*, 9(4):633–646, jul 2016.

[19] Huayu Li, Yong Ge, Richang Hong, and Hengshu Zhu. Point-of-interest recommendations: Learning potential check-ins from friends. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*, pages 975–984, New York, New York, USA, 2016. ACM Press.

[20] Huiji Gao, Jiliang Tang, and Huan Liu. gscorr: modeling geo-social correlations for new check-ins on location-based social networks. In *Proceedings of the 21st ACM international conference on Information and knowledge management - CIKM '12*, page 1582, New York, New York, USA, 2012. ACM Press.

[21] Jia-Dong Zhang and Chi-Yin Chow. Point-of-interest recommendations in location-based social networks. *SIGSPATIAL Special*, 7(3):26–33, jan 2016.

[22] Jie Bao, Yu Zheng, and Mohamed F. Mokbel. Location-based and preference-aware recommendation using sparse geo-social networking data. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems - SIGSPATIAL '12*, page 199, New York, New York, USA, 2012. ACM Press.

[23] Defu Lian, Xing Xie, Vincent W. Zheng, Nicholas Jing Yuan, Fuzheng Zhang, and Enhong Chen. CEPR: A Collaborative Exploration and Periodically Returning Model for Location Prediction. *ACM Transactions on Intelligent Systems and Technology*, 6(1):1–27, apr 2015.

[24] Kenneth Wai-Ting Leung, Dik Lun Lee, and Wang-Chien Lee. CLR: a collaborative location recommendation framework based on co-clustering. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information - SIGIR '11*, page 305, New York, New York, USA, 2011. ACM Press.

[25] W.Zheng Vincent, Zheng Yu, Xie Xing, and Yang Qiang. Towards mobile intelligence: Learning from GPS history data for collaborative recommendation. *Artificial Intelligence*, 184-185:17–37, jun 2012.

[26] Justin J. Levandoski, Mohamed Sarwat, Ahmed Eldawy, and Mohamed F. Mokbel. LARS: A Location-Aware Recommender System. In *2012 IEEE 28th International Conference on Data Engineering*, pages 450–461. IEEE, apr 2012.

[27] Bin Liu and Hui Xiong. Point-of-Interest Recommendation in Location Based Social Networks with Topic and Location Awareness. In *Proceedings of the 2013 SIAM International Conference on Data Mining*, pages 396–404. Society for Industrial and Applied Mathematics, Philadelphia, PA, may 2013.

[28] Quan Yuan, G A O Cong, Kaiqi Zhao, Zongyang Ma, and Aixin Sun. Who , Where , When , and What : A Nonparametric Bayesian Approach to Context-aware Recommendation and Search for Twitter Users. 33(1), 2015.

[29] Yu Zheng, Lizhu Zhang, Zhengxin Ma, Xing Xie, and Wei-Ying Ma. Recommending friends and locations based on individual location history. *ACM Transactions on the Web*, 5(1):1–44, feb 2011.

[30] Yuki Arase, Xing Xie, Takahiro Hara, and Shojiro Nishio. Mining people's trips from large scale geo-tagged photos. In *Proceedings of the international conference on Multimedia - MM '10*, page 133, New York, New York, USA, 2010. ACM Press.

[31] Defu Lian, Cong Zhao, Xing Xie, Guangzhong Sun, Enhong Chen, and Yong Rui. GeoMF:Joint Geographical Modeling and Matrix Factorization for Point-of-Interest Recommendation . In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '14*, pages 831–840, 2014.

[32] Yi-Liang Zhao, Liqiang Nie, Xiangyu Wang, and Tat-Seng Chua. Personalized Recommendations of Locally Interesting Venues to Tourists via Cross-Region Community Matching. *ACM Transactions on Intelligent Systems and Technology*, 5(3):1–26, jul 2014.

[33] Quan Yuan, Gao Cong, Kaiqi Zhao, Zongyang Ma, and Aixin Sun. Who, where, when and what: A non-parametric Bayesian approach to context-aware recommendation and search for Twitter users. *ACM Transactions on Information Systems*, 33(1):1–33, feb 2015.

[34] Moon-Hee Park, Jin-Hyuk Hong, and Sung-Bae Cho. Location-Based Recommendation System Using Bayesian User's Preference Model in Mobile Devices. In *Ubiquitous Intelligence and Computing*, pages 1130–1139. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[35] Hongzhi Yin, Bin Cui, Yizhou Sun, Zhiting Hu, and Ling Chen. LCARS: A spatial item recommender system. *ACM Transactions on Information Systems*, 32(3):1–37, jul 2014.

[36] Chen Cheng, Haiqin Yang, Michael R Lyu, and Irwin King. Where you like to go next: Successive point-of-interest recommendation. In *IJCAI International Joint Conference on Artificial Intelligence*, pages 2605–2611, 2013.

[37] Jia-dong Zhang and Chi-Yin Chow. Spatiotemporal Sequential Influence Modeling for Location Recommendations. *ACM Transactions on Intelligent Systems and Technology*, 7(1):1–25, 2015.

[38] Shenglin Zhao. STELLAR: Spatial-Temporal Latent Ranking for Successive Point-of-Interest Recommendation. *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 315–322, 2016.

[39] Takeshi Kurashima, Tomoharu Iwata, Go Irie, and Ko Fujimura. Travel route recommendation using geotags in photo sharing sites. In *Proceedings of the 19th ACM international conference on Information and knowledge management - CIKM '10*, page 579, New York, New York, USA, 2010. ACM Press.

[40] Vincent W. Zheng, Yu Zheng, Xing Xie, and Qiang Yang. Towards mobile intelligence: Learning from GPS history data for collaborative recommendation. *Artificial Intelligence*, 184-185:17–37, jun 2012.

[41] Fatima Mourchid, Jalel Ben Othman, Abdellatif Kobbane, Essaid Sabir, and Mohammed El Koutbi. A Markov Chain Model for Integrating Context in Recommender Systems. In *2016 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, dec 2016.

[42] Xin Liu, Yong Liu, Karl Aberer, and Chunyan Miao. Personalized point-of-interest recommendation by mining users' preference transition. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management - CIKM '13*, pages 733–738, New York, New York, USA, 2013. ACM Press.

[43] Ghim-Eng Yap, Xiao-Li Li, and Philip S. Yu. Effective Next-Items Recommendation via Personalized Sequential Pattern Mining. In *Proceedings of the 17th international conference on Database Systems for Advanced Applications - Volume Part II*, pages 48–64. Springer-Verlag, 2012.

[44] Chieh-Yuan Tsai and Lai Bo-Han. A Location-Item-Time sequential pattern mining algorithm for route recommendation. *Knowledge-Based Systems*, 73:97–110, jan 2015.

[45] Ted Gueniche, Philippe Fournier-Viger, and Vincent S. Tseng. Compact prediction tree: A lossless model for accurate sequence prediction. *Lecture Notes in Computer Science*, 8347 LNAI(PART 2):177–188, 2013.

[46] Ted Gueniche, Philippe Fournier-Viger, Rajeev Raman, and Vincent S. Tseng. CPT+: Decreasing the time/space complexity of the compact prediction tree. *Lecture Notes in Computer Science*, 9078:625–636, 2015.

[47] Philippe Fournier-Viger, Cheng-Wei Wu, Vincent S. Tseng, Longbing Cao, and Roger Nkambou. Mining Partially-Ordered Sequential Rules Common to Multiple Sequences. *IEEE Transactions on Knowledge and Data Engineering*, 27(8):2203–2216, aug 2015.

[48] Fernando Terroso-Saenz, Mercedes Valdes-Vela, and Antonio F. Skarmeta-Gomez. Online route prediction based on clustering of meaningful velocity-change areas. *Data Mining and Knowledge Discovery*, pages 1–40, 2016.

[49] C. Song, Z. Qu, N. Blumm, and A.-L. Barabasi. Limits of Predictability in Human Mobility. *Science*, 327(5968):1018–1021, feb 2010.

[50] Marta C. González, César A. Hidalgo, and Albert László Barabási. Understanding individual human mobility patterns. *Nature*, 453(7196):779–782, jun 2008.

[51] John Krumm. A Markov Model for Driver Turn Prediction. *SAE World Congress*, 2008.

[52] SUMO. Simulation of Urban MObility download | SourceForge.net.

[53] Amiya Bhattacharya and Sajal K. Das. LeZi-Update: An Information-Theoretic Framework for Personal Mobility Tracking in PCS Networks. *Wireless Networks*, 8(2/3):121–135, 2002.

[54] Salvatore Scellato, Mirco Musolesi, Cecilia Mascolo, Vito Latora, and Andrew T. Campbell. NextPlace: A spatio-temporal prediction framework for pervasive systems. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 6696 LNCS, pages 152–169. Springer, Berlin, Heidelberg, 2011.

[55] F. Marchal, J. Hackney, and K. Axhausen. Efficient Map Matching of Large Global Positioning System Data Sets: Tests on Speed-Monitoring Experiment in Zürich. *Transportation Research Record: Journal of the Transportation Research Board*, 1935:93–100, jan 2005.

[56] TrackMatching. Cloud based web service for map-matching gps data on the openstreetmap. https://mapmatching.3scale.net/. (Accessed on 01/05/2020).

[57] Alexandre de Brébisson, Étienne Simon, Alex Auvolat, Pascal Vincent, and Yoshua Bengio. Artificial Neural Networks Applied to Taxi Destination Prediction. jul 2015.

[58] Tomáš Mikluščák, Michal Gregor, and Aleš Janota. Using neural networks for route and destination prediction in intelligent transport systems. *Communications in Computer and Information Science*, 329 CCIS:380–387, 2012.

[59] Francisco Dantas N Neto, Cláudio De Souza Baptista, and Cláudio E C Campelo. Prediction of Destinations and Routes in Urban Trips with Automated Identification of Place Types and Stay Points. pages 80–91, 2015.

[60] Francisco Dantas Nobre Neto, Claudio de Souza Baptista, and Claudio E. C. Campelo. A user-personalized model for real time destination and route prediction. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pages 401–407. IEEE, nov 2016.

[61] Attila István Petróczi and Csaba Gáspár-Papanek. Route prediction on tracking data to location-based services. *Lecture Notes in Computer Science*, 5733 LNCS:69–77, 2009.

[62] Reid Simmons, Brett Browning, Yilu Zhang, and Varsha Sadekar. Learning to Predict Driver Route and Destination Intent.

[63] Disheng Qiu, Paolo Papotti, and Lorenzo Blanco. Future locations prediction with uncertain data. *Lecture Notes in Computer Science*, 8188 LNAI(PART 1):417–432, 2013.

[64] Jonathan P. Epperlein, Julien Monteil, Mingming Liu, Yingqi Gu, Sergiy Zhuk, and Robert Shorten. Bayesian classifier for route prediction with markov chains. *CoRR*, abs/1808.10705, 2018.

[65] Guangtao Xue, Zhongwei Li, Hongzi Zhu, and Yunhuai Liu. Traffic-known urban vehicular route prediction based on partial mobility patterns. *Proceedings of the International Conference on Parallel and Distributed Systems - ICPADS*, pages 369–375, 2009.

[66] Meng Chen, Xiaohui Yu, and Yang Liu. Mining moving patterns for predicting next location. *Information Systems*, 54:156–168, 2015.

[67] Xipeng Wang, Yuan Ma, Junru Di, Yi L. Murphey, Shiqi Qiu, Johannes Kristinsson, Jason Meyer, Finn Tseng, and Timothy Feldkamp. Building efficient probability transition matrix using machine learning from big data for personalized route prediction. *Procedia Computer Science*, 53(1):284–291, 2015.

[68] John Krumm. Paper Number 2008-01-0195 A Markov Model for Driver Turn Prediction. 2008.

[69] Guangtao Xue, Yuan Luo, Jiadi Yu, and Minglu Li. A novel vehicular location prediction based on mobility patterns for routing in urban VANET. *EURASIP Journal on Wireless Communications and Networking*, 2012(1):222, dec 2012.

[70] R. Simmons, B. Browning, Yilu Zhang Yilu Zhang, and V. Sadekar. Learning to Predict Driver Route and Destination Intent. *2006 IEEE Intelligent Transportation Systems Conference*, pages 127–132, 2006.

[71] J. Cleary and I. Witten. Data Compression Using Adaptive Coding and Partial String Matching. *IEEE Transactions on Communications*, 32(4):396–402, 1984.

[72] J. Ziv and A. Lempel. Compression of individual sequences via variable-rate coding. *IEEE Transactions on Information Theory*, 24(5):530–536, sep 1978.

[73] M. Feder and N. Merhav. Relations between entropy and error probability. *IEEE Transactions on Information Theory*, 40(1):259–266, Jan 1994.

[74] Dmitry A Shkarin. Improving the efficiency of the ppm algorithm. *Problems of information transmission*, 37(3):226–235, 2001.

[75] Martin Ester, Martin Ester, Hans-peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. pages 226—-231, 1996.

[76] Amar Farouk Merah, Samer Samarah, Azzedine Boukerche, and Abdelhamid Mammeri. A sequential patterns data mining approach towards vehicular route prediction in VANETs. *Mobile Networks and Applications*, 18(6):788–802, 2013.

[77] Qian Ye, Ling Chen, and Gencai Chen. Predict personal continuous route. *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, pages 587–592, 2008.

[78] Ling Chen, Mingqi Lv, and Gencai Chen. A system for destination and future route prediction based on trajectory mining. *Pervasive and Mobile Computing*, 6(6):657–676, 2010.

[79] K. Torkkola, K. Zhang, H. Li, H. Zhang, C. Schreiner, and M. Gardner. Traffic advisories based on route prediction. *Proceedings of Workshop on Mobile Interaction with the Real World*, (September):33–36, 2007.

[80] Jian Jian Pei, Jiawei Jiawei Han, B. Mortazavi-Asl, Jianyong Jianyong Wang, H. Pinto, Qiming Qiming Chen, U. Dayal, and Mei-Chun Mei-Chun Hsu. Mining sequential patterns by pattern-growth: the PrefixSpan approach. *IEEE Transactions on Knowledge and Data Engineering*, 16(11):1424–1440, nov 2004.

[81] Fernando Terroso-Sáenz, Mercedes Valdés-Vela, Francisco Campuzano, Juan A. Botia, and Antonio F. Skarmeta-Gómez. A complex event processing approach to perceive the vehicular context. *Information Fusion*, 21:187–209, 2015.

[82] Anil K. Jain, Jianchang Mao, and K. M. Mohiuddin. Artificial neural networks: A tutorial, mar 1996.

[83] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133, dec 1943.

[84] Jon Froehlich and John Krumm. Route Prediction from Trip Observations. *Proceedings of SAE World Congress*, 2193(2008-01-0201):53, 2008.

[85] Patrick Helmholz, Edgar Ziesmann, and Susanne Robra-Bissantz. Context-awareness in the car: Prediction, evaluation and usage of route trajectories. *Lecture Notes in Computer Science*, 7939 LNCS:413–419, 2013.

[86] Paul Newson and John Krumm. Hidden Markov map matching through noise and sparseness. *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems - GIS '09*, pages 336–343, 2009.

[87] Giovanni Seni and John F. Elder. Ensemble Methods in Data Mining: Improving Accuracy Through Combining Predictions. *Synthesis Lectures on Data Mining and Knowledge Discovery*, 2(1):1–126, jan 2010.

[88] Hanane Amirat, Nasreddine Lagraa, Philippe Fournier-Viger, and Youcef Ouinten. MyRoute: A Graph-Dependency Based Model for Real-Time Route Prediction. *Journal of Communications*, pages 668–676, 2017.

[89] Hanane Amirat, Nasreddine Lagraa, Philippe Fournier-Viger, and Youcef Ouinten. NextRoute: a lossless model for accurate mobility prediction. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–21, may 2019.

[90] James Pitkow and Peter Pirolli. Mining longest repeating subsequences to predict world wide web surfing, 1999.

[91] Philippe Fournier-Viger, Antonio Gomariz, Ted Gueniche, Azadeh Soltani, Cheng-Wei Wu, and Vincent S. Tseng. *SPMF: A Java Open-Source Pattern Mining Library*, volume 15. MIT Press, 2001.

[92] Lara Codeca, Raphael Frank, and Thomas Engel. Luxembourg SUMO Traffic (LuST) Scenario: 24 hours of mobility for vehicular networking research. In *2015 IEEE Vehicular Networking Conference (VNC)*, pages 1–8. IEEE, dec 2015.

[93] Michal Piorkowski, Natasa Sarafijanovic-Djukic, and Matthias Grossglauser. CRAWDAD dataset epfl/mobility (v. 2009-02-24). *CRAWDAD wireless network data archive*, jan 2009.

[94] Yu Zheng, Lizhu Zhang, Xing Xie, and Wei-ying Ma. Mining Interesting Locations and Travel Sequences from GPS Trajectories. (49), 2009.

[95] Venkata N. Padmanabhan and Jeffrey C. Mogul. Using predictive prefetching to improve World Wide Web latency. *ACM SIGCOMM Computer Communication Review*, 26(3):22–36, 1996.

[96] Philippe Fournier-Viger, Antonio Gomariz, Ted Gueniche, Azadeh Soltani, Cheng-Wei Wu, and Vincent S. Tseng. SPMF: A Java Open-Source Pattern Mining Library. *Journal of Machine Learning Research*, 15:3569–3573, 2014.

[97] Vassilis Kostakos. Temporal graphs. *Physica A: Statistical Mechanics and its Applications*, 388(6):1007–1023, mar 2009.

[98] Huiji Gao, Jiliang Tang, Xia Hu, and Huan Liu. Content-aware point of interest recommendation on location-based social networks. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, pages 1721–1727. AAAI Press, 2015.

[99] Shenglin Zhao, Tong Zhao, Haiqin Yang, Michael R Lyu, and Irwin King. STELLAR : Spatial-Temporal Latent Ranking for Successive Point-of-Interest Recommendation. *[AAAI2016]Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 315–321, 2016.

[100] Yanchi Liu, Chuanren Liu, Bin Liu, Meng Qu, and Hui Xiong. Unified Point-of-Interest Recommendation with Temporal Interval Assessment. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*, pages 1015–1024, New York, New York, USA, 2016. ACM Press.

[101] Yue Shi, Pavel Serdyukov, Alan Hanjalic, and Martha Larson. Nontrivial landmark recommendation using geotagged photos. *ACM Transactions on Intelligent Systems and Technology*, 4(3):1, jun 2013.

[102] Yu Zheng, Xing Xie, and Wei-Ying Ma. Geolife: A collaborative social networking service among user, location and trajectory. *IEEE Data(base) Engineering Bulletin*, June 2010.

[103] Eunjoon Cho, Seth A. Myers, and Jure Leskovec. Friendship and mobility: User movement in location-based social networks. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1082–1090, New York, New York, USA, 2011. ACM Press.

[104] Zhiyuan Cheng, James Caverlee, Kyumin Lee, and Daniel Z Sui. Exploring millions of footprints in location sharing services. In *International Conference on Web and Social Media (ICWSM)*, pages 81–88, 2011.

[105] Dingqi Yang, Bingqing Qu, Jie Yang, and Philippe Cudre-Mauroux. Revisiting User Mobility and Social Relationships in LBSNs: A Hypergraph Embedding Approach. In *The World Wide Web Conference on - WWW '19*, pages 2147–2157, New York, New York, USA, may 2019. ACM Press.

[106] Dingqi Yang, Daqing Zhang, and Bingqing Qu. Participatory cultural mapping based on collective behavior data in location-based social networks. *ACM Transactions on Intelligent Systems and Technology*, 7(3):1–23, jan 2016.

[107] Dingqi Yang, Daqing Zhang, Zhiyong Yu, and Zhiwen Yu. Fine-Grained preference-aware location search leveraging crowdsourced digital footprints from LBSNs. In *UbiComp 2013 - Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 479–488, New York, New York, USA, 2013. ACM Press.

[108] Hu Bo and Ester Martin. Social topic modeling for point-of-interest recommendation in location-based social networks. In *2014 IEEE International Conference on Data Mining*, pages 845–850, Dec 2014.

[109] Defu Lian, Yong Ge, Fuzheng Zhang, Nicholas Jing Yuan, Xing Xie, Tao Zhou, and Yong Rui. Content-aware collaborative filtering for location recommendation based on human mobility data. In *Proceedings of the 2015 IEEE International Conference on Data Mining (ICDM)*, ICDM '15, pages 261–270, Washington, DC, USA, 2015. IEEE Computer Society.

[110] Dequan Zhou, Bin Wang, Seyyed Mohammadreza Rahimi, and Xin Wang. A Study of Recommending Locations on Location-Based Social Network by Collaborative Filtering. pages 255–266, 2012.

[111] Michael J. Pazzani and Daniel Billsus. Content-Based Recommendation Systems. In *The Adaptive Web*, pages 325–341. Springer Berlin Heidelberg, Berlin, Heidelberg.

[112] Xiaoyuan Su and Taghi Khoshgoftaar. A survey of collaborative filtering techniques. *Adv. Artificial Intellegence*, 2009, 10 2009.

[113] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web, WWW 2001*, pages 285–295. Association for Computing Machinery, Inc, apr 2001.

[114] Yue Shi, Martha Larson, and Alan Hanjalic. Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges, 2014.

[115] Manos Papagelis, Dimitris Plexousakis, and Themistoklis Kutsuras. Alleviating the sparsity problem of collaborative filtering using trust inferences. In *Lecture Notes in Computer Science*, volume 3477, pages 224–239, 2005.

[116] Hongzhi Yin, Yizhou Sun, Bin Cui, Zhiting Hu, and Ling Chen. LCARS: A location-content-aware recommender system. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '13*, page 221, New York, New York, USA, 2013. ACM Press.

[117] Anastasios Noulas, Salvatore Scellato, Neal Lathia, and Cecilia Mascolo. Mining user mobility features for next place prediction in location-based services. In *Proceedings - IEEE International Conference on Data Mining, ICDM*, pages 1038–1043, 2012.

[118] Quan Yuan, Gao Cong, Zongyang Ma, Aixin Sun, and Nadia Magnenat-thalmann. Time-aware Point-of-interest Recommendation. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval - SIGIR '13*, pages 363–372, 2013.

[119] Chen Cheng, Haiqin Yang, Irwin King, and Michael R. Lyu. Fused matrix factorization with geographical and social influence in location-based social networks, 2012.

[120] Shenglin Zhao, Irwin King, and Michael R. Lyu. Capturing geographical influence in POI recommendations. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 8227 LNCS, pages 530–537, 2013.

[121] Jia-Dong Zhang and Chi-Yin Chow. iGSLR: personalized geo-social location recommendation. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems - SIGSPATIAL'13*, pages 324–333, New York, New York, USA, 2013. ACM Press.

[122] Jia-Dong Zhang and Chi-Yin Chow. GeoSoCa: Exploiting Geographical, Social and Categorical Correlations for Point-of-Interest Recommendations. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval - SIGIR '15*, pages 443–452, New York, New York, USA, 2015. ACM Press.

[123] Jia-Dong Zhang, Chi-Yin Chow, and Yu Zheng. Orec: An opinion-based point-of-interest recommendation framework. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management - CIKM '15*, pages 1641–1650, New York, New York, USA, 2015. ACM Press.

[124] Chen Cheng, Haiqin Yang, Irwin King, and Michael R. Lyu. Fused matrix factorization with geographical and social influence in location-based social networks. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, AAAI'12, pages 17–23. AAAI Press, 2012.

[125] Bernard. W. Silverman. *Density estimation: For statistics and data analysis*. Chapman and Hall, 1986.

[126] Jia-Dong Zhang and Chi-Yin Chow. iGSLR: personalized geo-social location recommendation. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems - SIGSPATIAL'13*, pages 324–333, New York, New York, USA, 2013. ACM Press.

[127] Mao Ye, Peifeng Yin, and Wang-Chien Lee. Location recommendation for location-based social networks. In *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems - GIS '10*, page 458, New York, New York, USA, 2010. ACM Press.

[128] Yi Shu Lu, Wen Yueh Shih, Hung Yi Gau, Kuan Chieh Chung, and Jiun Long Huang. On successive point-of-interest recommendation. *World Wide Web*, 22(3):1151–1173, may 2019.

[129] An-Jung Cheng, Yan-Ying Chen, Yen-Ta Huang, Winston H. Hsu, and Hong-Yuan Mark Liao. Personalized travel recommendation by mining people attributes from community-contributed photos. In *Proceedings of the 19th ACM international conference on Multimedia - MM '11*, page 83, New York, New York, USA, 2011. ACM Press.

[130] Jialiang Chen, Xin Li, William K. Cheung, and Kan Li. Effective successive POI recommendation inferred with individual behavior and group preference. *Neurocomputing*, 210:174–184, oct 2016.

[131] Jia Dong Zhang, Gabriel Ghinita, and Chi Yin Chow. Differentially Private Location Recommendations in Geosocial Networks. In *2014 IEEE 15th International Conference on Mobile Data Management*, pages 59–68. IEEE, jul 2014.

[132] Zaiben Chen, Heng Tao Shen, and Xiaofang Zhou. Discovering popular routes from trajectories. In *2011 IEEE 27th International Conference on Data Engineering*, pages 900–911. IEEE, apr 2011.

[133] Takeshi Kurashima, Tomoharu Iwata, Takahide Hoshide, Noriko Takaya, and Ko Fujimura. Geo topic model. In *Proceedings of the sixth ACM international conference on Web search and data mining - WSDM '13*, page 375, New York, New York, USA, 2013. ACM Press.

[134] Jia-Dong Zhang, Chi-Yin Chow, and Yanhua Li. LORE: Exploiting sequential influence for location recommendations. In *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems - SIGSPATIAL '14*, pages 103–112, New York, New York, USA, 2014. ACM Press.

[135] Thomas Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 1999*, pages 50–57, New York, New York, USA, aug 1999. Association for Computing Machinery, Inc.

[136] Florent Masseglia, Pascal Poncelet, and Maguelonne Teisseire. Incremental mining of sequential patterns in large databases. *Data & Knowledge Engineering*, 46(1):97–121, jul 2003.

[137] Omer Adam, Zailani Abdullah, Amir Ngah, Kasypi Mokhtar, Wan Muhamad Amir Wan Ahmad, Tutut Herawan, Noraziah Ahmad, Mustafa Mat Deris, Abdul Razak Hamdan, and Jemal H. Abawajy. Inc-SPADE: An Incremental Sequential Pattern Mining Algorithm Based on SPADE Property. pages 81–92. Springer, Cham, 2016.

[138] chang Chin-Chen, Li Yu-Chiang, and Lee Jung-San. An Efficient Algorithm for Incremental Mining of Association Rules. In *15th International Workshop on Research Issues in Data Engineering: Stream Data Mining and Applications (RIDE-SDMA'05)*, pages 3–10. IEEE.

[139] B. Nath, D. K. Bhattacharyya, and A. Ghosh. Incremental association rule mining: a survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 3(3):157–169, may 2013.

[140] Hanane Amirat, Abderrahim Benslimane, Philippe Fournier-Viger, and Nasreddine Lagraa. LocRec: Rule-based successive location recommendation in LBSN. In *IEEE International Conference on Communications*, volume 2018-May. Institute of Electrical and Electronics Engineers Inc., jul 2018.

[141] Snap: Network datasets: Gowalla. https://snap.stanford.edu/data/loc-gowalla.html. (Accessed on 06/17/2020).

[142] Snap: Network datasets: Brightkite. https://snap.stanford.edu/data/loc-brightkite.html. (Accessed on 06/17/2020).