Time Aware Location Recommendations in Location Based Social Networks

Doctoral Dissertation submitted to the
Faculty of Informatics of the Università della Svizzera Italiana
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

presented by
Steven Mudda

under the supervision of
Prof. Silvia Giordano

June 2018

Dissertation Committee

Prof. Marc Langheinrich	Università della Svizzera Italiana, Switzerland
Prof. Cecilia Mascolo	University of Cambridge, Cambridge, United Kingdom
Prof. Cesare Pautasso	Università della Svizzera Italiana, Switzerland
Prof. Raffaele Perego	Consiglio Nazionale delle Ricerche, Pisa, Italy

Dissertation accepted on 26 June 2018

Research Advisor	PhD Program Director
Prof. Silvia Giordano	**The PhD program Director**

I certify that except where due acknowledgement has been given, the work presented in this thesis is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; and the content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program.

Steven Mudda
Lugano, 26 June 2018

To my beloved grandparents and my life gurus
Charlie Munger and Warren Buffett.

Abstract

Smartphones have fundamentally changed how people make choices about the products and services they consume and the way they interact with each other. The spread and extensive usage of mobile applications has led to the rise of Location-based Social Network (LBSN) services like Foursquare, Yelp etc, that aim to aim to provide new and novel places of interest to people based on their interests and habits.

A Recommendation system helps users to discover places they may like and also enable them to narrow down their choices. In particular, providing relevant location recommendations to users is essential to drive customer engagement with the mobile application and is also an important research topic. Multiple studies on human mobility patterns and my analysis on different LBSN datasets have shown that the preference of users for different locations changes with time, i.e., type of locations visited in the afternoon are different from those visited in the evening. Majority of recommendation systems in LBSN do not take into account the temporal aspect of recommendation. A recommendation system must be able to provide locations to users by taking into account their 1) stationary preferences that don't change with time and 2) temporal preference that differ with time and recommended locations that are relevant in time to the user.

This thesis first presents a feature based location recommendation model, `REGULA` that exploits the regular mobility behavior of people and incorporates temporal information to provide better location recommendations. `REGULA` outperforms other feature and graph-based location recommendation models. Further, this thesis presents the first model to recommend interesting areas to people based on their Call Detail Records. Finally, this thesis presents two deep neural network based location recommendation models (`DEEPREC` and `DEEPTREC`) that are used to learn the stationary and temporal preferences of users. The combined model (`JOINTDEEPREC`) can be used to provide time-aware location recommendations to people. The model was evaluated on one of the largest check-in dataset collected at Microsoft Research Asia and outperforms state-of-the-art model by a factor of 10.

Acknowledgements

I first to express my gratitude to my advisors Prof. Silvia Giordano and Prof. Fabio Crestani. I would like to thank Prof. Silvia Giordano for always encouraging and supporting me to solve the different research challenges encountered during my Ph.D. and helping me to grow both professionally and personally. I would like to thank Prof. Fabio Crestani for the multiple discussions and his valuable feedback that enabled to understand many things during my research. A special thank you to all the members of my dissertation committee and my research prospectus. The comments and feedback have helped me to better understand the things to focus on.

Many thanks to Defu Lian, Fuzheng Zhang, Xing Xie for providing me an opportunity to do my internship at Microsoft Research Asia, Beijing and multiple discussions that enabled me to design a time-aware deep neural network model. I also thank Cecilia Mascolo for her recommendation. Further, I would like to thank my colleagues during the internship Chao-Chung Wu from NTU, Taiwan and Shonosuke Ishiwatari from The University of Tokyo for helping to understand the fundamentals of recurrent neural networks. Finally, I would like to thank my colleagues at SUPSI for their support. A special thanks go to Alessandro Puiatti, Salvatore Vanini, and Daniele Puccinelli.

I am incredibly grateful to my parents and my siblings for their constant and unconditional love and support. Finally and most importantly, I give my deepest and most special thanks to my beloved wife Kamini for her immense support and love during my Ph.D.

Contents

Figures

Tables

Chapter 1

Introduction

1.1 Motivation

The purpose of any business is to create and keep a customer[1], and in today's world, modern customers have a wide range of choices. Companies offer a vast selection of products and services like entertainment (Netflix, HBO), eCommerce (Amazon, Alibaba) shared economy (Uber, Airbnb) with the aim to meet the various needs of customers. The Internet has not only enabled businesses to interact with their customers directly but has also led to a scenario where their competitor is only one mouse click away[2]. It is a well-known fact that the probability of selling to an existing customer is much higher than compared to a new prospect Farris et al. [2010]. Therefore, making customers happy by providing appropriate products and services is critical to enhance customer satisfaction and create loyalty. A Recommendation system helps users to discover products they may like and is essential to drive customer engagement. Recommendation systems provide a scalable way to offer personalized products and services for a broad set of users in multiple scenarios. Recommendations not only help people to find what they are looking for but also enable them to narrow down their choices.

Recent years has witnessed significant growth in the usage of smartphones, and as of 2017 one in every three people in the world uses it (Global-Smartphone-Penetration). With the increasing usage of smartphones in the daily lives of people, these mobile devices accounted for 33% of the global internet traffic in 2017. These devices have significantly altered the way people interact with each other, do shopping, consume media content, etc. Specifically, smartphone-based social media applications enable users to communicate with each other and share

[1] Quote by Peter Ferdinand Drucker

[2] Quote by Doug Warner

a wide range of information like products/services they use, places and events they visit, etc. The spread and extensive usage of mobile applications has led to the rise of Location-based social network (LBSN) services like Foursquare, Jeipang, Yelp, etc. LBSNs consist of users visiting some places of interests and sharing information about them with friends in their social network. LBSNs are social and physical information-rich networks that incorporate mobility patterns and social ties of humans. At the same time, this usage of mobile phone services has led people to use them for discovering new places of experiences like attractions, restaurants, hotels, etc. This novelty-seeking behavior of people has led to the growth of location recommender services that aim to provide new places to people based on their interests and habits Zheng et al. [2010a]. In particular, providing relevant location recommendations to users of an LBSN is essential to drive customer engagement with the mobile application and is also an important research topic Bao et al. [2015].

Recommending locations entails some unique challenges than compared to recommending movies or items. In traditional recommender systems like recommending movies, or products on a website, the goal is to provide a list of movies/items based on the profiles and previous behavior of the user. Whether a movie/item is of interest to the user is validated by whether the user has clicked that particular online movie/item. So the feedback and behavior of user towards the movie/item is much faster, and the cost of accepting the recommended movie/item by the user is low. In the case of location recommendation, the probability that a user visits a recommended location depends on multiple factors like: (i) what is the current location of user, (ii) how far is the recommended place, (iii) what is the time of recommendation, (iv) if the user is alone or with other people with different interests, etc. Also, the feedback related to whether a recommendation was useful or not is delayed due to the physical distance travel by the user to the recommended place compared to just a click for a recommended movie/item. Therefore, the problem of location recommendation has some unique and different characteristics compared to recommending a movie/item online.

Multiple studies (Gonzalez et al. [2008b]Papandrea et al. [2016]) to understand the mobility patterns of people have corroborated the fact that humans are creatures of habit and frequently visit a specific set of locations like home, office, favorite restaurant, etc. Recommending new locations close to these frequently visited locations increases the possibility that the user will visit the recommended place. Also, since humans are social animals, they are more likely to visit a new location recommended by their friends. Further, humans exhibit a lot of inertia and are unlikely to visit locations geographically far from their current location,

e.g., a place recommended in the afternoon must be close to a person's office to increase the probability that the person will visit the recommended place. Existing recommendation models do not fully exploit these regular mobility habits of humans summarized as a) people have regular visit patterns and explore locations close to their usual places; b) people go to locations recently visited by others, especially friends; c) people prefer to visit locations close to their current location.

Multiple studies on human mobility patterns Noulas et al. [2011]; Cho et al. [2011b] and my analysis on different LBSN datasets have shown that the preference of users for different locations changes with time, i.e., type of locations visited in the afternoon are different from those visited in the evening. Majority of recommendation systems in LBSN do not take into account the temporal aspect of recommendation. A real-world system must be able to provide location recommendations to a wide range of people based on their personal preferences, continuously learn from their time-varying mobility behavior and recommend locations that are relevant in time to the user.

Recent advances in the parallel processing hardware like graphics cards have enabled researchers to perform large-scale computations and has lead to the success of deep neural networks in modeling complex patterns in data Goodfellow et al. [2016]. Deep neural networks can solve complex tasks because theoretically, neural networks can compute any function Goodfellow et al. [2016]. In the past few years, deep learning techniques have achieved great results compared to well-known benchmarks specifically in the domains of computer vision Krizhevsky et al. [2012] and speech recognition Sainath et al. [2015]. Deep neural networks can be used to capture non-linear preferences of users for different locations effectively. Some of the deep neural networks have memory and can be used to model the time-varying personalized preferences of users and locations. A deep neural network based recommendation model can continuously learn the personal preferences of users and provide time-relevant location recommendations.

1.2 Research Challenges

This thesis addresses the problem of recommending new locations to users of a Location Based Social Network (LBSN) and provide novel models that overcome the limitations of existing state-of-the-art location recommendation models.

The thesis will be mainly concerned with the following research questions:

R1 How to exploit the mobility behavior of humans to provide relevant location recommendations to people?

- R1.1 How to capture the geographical preferences of users?
- R1.2 How to capture the temporal preferences of users?
- R1.3 How to capture the influence of social peers on the preferences of users?
- R1.4 How to combine these multiple preferences of users and provide accurate location recommendations to a wide range of users?

R2 Handcrafting features to capture location preferences of users in different application scenarios is difficult to scale. How to overcome this challenge with the help of deep neural networks?

- R2.1 How to design a deep neural network based model that can learn the locations preferences of users from their check-in history and recommend new locations to them?
- R2.2 How to capture the geographical constraints of neighboring locations in a deep neural network based location recommendation model?

R3 Location preferences of users change with time. How to design a deep neural network based model that can learn the time-varying location preferences of users?

- R3.1 How to capture the temporal dimension so that the model learns the day-wise preferences of users to provide time-aware location recommendations?

1.3 Contributions

The main contributions of this thesis can be summarized as follows:

1. I have analyzed and mined large LBSN datasets (Gowalla, Brightkite, etc.) to capture essential aspects for location recommendations like 1) People regularly (or habitually) visit a set of locations, 2) People go to places close to these regularly visited locations, 3) People are more likely to visit places that were recently visited by others like friends.

2. Based on the analysis of human mobility patterns, I designed three key features a) Temporal feature that captures the influence of time in user visits, b) Distance feature that captures the geographical influence of user visits, and c) Friendship feature that captures the influence of the social friends on user visits. I have used these features to build `REGULA` , a location recommendation model. `REGULA` shows very good performance compared to other feature and graph-based recommendation models.

3. Based on insights gained from the analysis of LBSN datasets, I designed a variant of `REGULA` to recommend new cell regions to users based on their Call Detail Records (CDRs). CDRs capture the mobility activities and social ties of a large number of users. Researchers have observed that CDRs are one of the most valuable sources of data to perform user-centric analysis, especially when related to mobility and sociality. I propose the first research work to leverage the rich information in CDRs to construct a CDR-based LBSN and found that they share similar characteristics with LBSNs (refer to Chapter 4). My results show the feasibility of recommendation in CDRs and the importance of taking into account human behavior characteristics. It also opens possibilities to develop novel CDR-based services.

4. Recent years have witnessed a rapid increase in the volume and complexity of data and existing feature/graph based recommendation models are not designed to overcome this continuous information overload. Deep neural networks are best suited to handle large amounts of data and learn continuously. I propose a deep neural network based model (`DEEPREC`) that learns the location preferences of users by understanding their complex visiting patterns at different locations. `DEEPREC` is the first model that incorporates geographic constraints of neighborhood locations to ensure that locations that are within a small geographic region share similar geographical preference than compared to locations that are far away. The `DEEPREC` model uses a Feed Forward Neural Network (FFNN) to learn location preferences of users and another FFNN with a softmax layer to capture the geographical constraints between locations.

5. I propose a time-aware recommendation model (DEEPTREC) that learns the time-varying preferences of people and recommends locations accordingly. The DEEPTREC model learns the mobility patterns of each user using a Recurrent Neural Network and provides day-wise recommendations, i.e., diverse locations for different days of the week.

6. I provide a joint model (JOINTDEEPREC) that collectively learns the time-independent (or personal) locations preferences of users together with geographical constraints using DEEPREC and time-varying locations preferences of users using DEEPTREC . The JOINTDEEPREC model is the first research work to use a neural network to learn geographical constraints and jointly learn time-independent and time-varying location preferences of users.

7. The deep learning based models (DEEPREC and JOINTDEEPREC) were validated on one of the largest check-in dataset collected at Microsoft Research Asia and have outperformed the state-of-the-art by a factor of 10. The dataset was constructed by crawling Sina Weibo (China's Twitter), the largest social networking website in China, that provides location-based services such as check-ins.

8. The results presented in this thesis also demonstrates the usefulness of deep neural networks in the field of LBSNs. Applying deep learning techniques to location recommendation is still in progress and opens new directions for future research works.

1.3.1 Limitations of Models and Data

REGULA is a feature based model where I need to compute the distance, temporal and friendship scores for all locations in the dataset before I recommend a set of new locations to users. Further, every time a user requests a recommendation, I need to compute all the scores, this procedure is not scalable in a real-world system. I overcome these limitations with the help of a deep neural network based recommendation model presented in Chapter 5. Finally, the Brightkite LBSN dataset was collected at a time when users were leaving their service. It has led to a significant drop in the number of active users who were recommended a new location. Similarly, since the CDR-based LBSN dataset was built based on the CDR of users. Therefore, the social ties captured by it is not as strong compared to users who are friends in a social network and visit location together.

In DEEPREC , the predicted preference of a user for location depends only on the current latent factors and the current state of the entire neural network. Further, DEEPREC does not capture the historical interaction between users and locations. Therefore, the set of locations recommended by DEEPREC at different times is the same. DEEPTREC models the time-dependent preferences of users. It utilizes a recurrent neural network based model for each user and location. Therefore, the computational complexity is high as it might require many hours to train all user and location models. Multiple research works are already underway to improve the training procedures of neural networks, as finding the right weights and biases of all hundreds of neurons is important for a neural network model to learn correctly and predict accurate information. Since each user/location model is also provided temporal information, i.e., check-in's day of the visit, there is a need to have enough data available for each user for each day of the week to be able to learn meaningful information about the user's preferences. The Sina Weibo dataset does not have a large number of check-ins for each hour of the day. Therefore, datasets with a higher granularity of data can be used to learn the temporal preferences of users and locations and provide more accurate location recommendations.

1.4 Organization of Thesis

The rest of the thesis is structured as follows:

Chapter 2 will present the State of the Art (SOA) for location recommendation. I give the broad classification of recommendation systems based on their objectives. Later, I present the literature review of location recommendations specifically providing Points of Interest(POI). I mainly focus on the SOA related to POI recommendation instead of a trajectory recommendation. Further, I present the literature on providing time independent and time aware locations recommendations. Finally, I also highlight recent research work on deep neural networks based location recommendations.

Chapter 3 will present the first effort done in the direction of designing a location recommendation model, i.e., REGULA . This chapter presents 5 hypothesis I have formulated about mobility behavior of users along with the experimental analysis done on two LBSN datasets (Gowalla and Brightkite) to test them. This chapter also presents the three features that are used to build REGULA location recommendation model. Finally, this chapter introduces the performance metrics used to measure and compare the performance of REGULA with other location recommendation models.

Chapter 4, will focus on my first work on recommending new cell regions to people based on their Call Detail Records (CDR). This chapter shows how I utilize CDR to construct a new CDR-based LBSN and analytical test conducted to demonstrate that CDR users also exhibit regular mobility behavior. Further, I present a variant of REGULA used to recommend new cell regions to users based on their CDR. Finally, this chapter presents extensive analysis performed to measure an compare the performance of REGULA with state-of-the-art factorization based recommendation models.

In Chapter 5, I will present my novel deep neural network based recommendation model, DEEPREC . Similar to the majority of research works, it provides time-independent location recommendations to users. In this chapter, I will present in detail the basic procedure of how factorization based model aims to capture the preferences of users and locations. Later, I will show why and how a deep neural network can capture the user and location preferences better than factorization technique. In this chapter, I will present the Feed Forward Neural Network (FFNN) used to learn the preferences of users and locations. I will also present the FFNN with a specific output layer called softmax that was used to capture the geographical constraints between neighborhood locations. Further, I will introduce cost functions used to train the two neural networks. Finally, this chapter presents the characteristics of one of the largest dataset used to evaluate DEEPREC along with the performance metrics used to compare its performance with other recommendation models.

In Chapter 6, I will present my novel deep neural network based recommendation model, DEEPTREC that can be used to provide time-aware location recommendations to users. In this chapter, I will discuss the basics of a recurrent neural network specifically a Gated Recurrent Unit (GRU) used to build DEEPTREC . In this chapter, I will present how multiple GRUs can be used to learn the time-varying preferences of users and locations. Further, I will introduce cost function used to train individual GRUs. In this chapter, I present how I combine DEEPREC (described in Chapter 5) and DEEPTREC to form a JOINTDEEPREC model that can be use to provide time-aware location recommendations to users. Finally, this chapter presents the procedure to evaluate JOINTDEEPREC along with the performance metrics used to compare it with other time-aware recommendation models.

Finally, Chapter 7 summarizes the work described in this thesis with an overview of the main findings and results. The Chapter will also provide the future directions derived from this work.

Chapter 2

State of the Art

2.1 Introduction

The worldwide adoption of smartphones has accelerated significantly in recent years. According to a 2017 market report (Global-Smartphone-Penetration), a third of the world's population uses a smartphone. It has led to the growth of wide range of mobile platform based services like entertainment (Netflix, HBO), mobile commerce (Amazon, Alibaba), shared economy (Uber, Airbnb), social media (Twitter, Weibo, Facebook), etc. Specifically, smartphone-based social media applications enable users to communicate with each other and share a wide range of information like products/services they use, places and events they visit, etc. The spread and extensive usage of mobile applications has led to the rise of Location-based social networks (LBSNs) services like Foursquare, Jeipang, Yelp and so on. LBSNs consist of users visiting some places of interests and sharing information about them with friends in their social network. LBSNs are social and physical information-rich networks that incorporate mobility patterns and social ties of humans. At the same time, this usage of mobile phone services has led people to use them for discovering new places of experiences like attractions, restaurants, hotels, etc. This novelty-seeking behavior of people has generated to the growth of location recommender services that aim to provide new places to people based on their interests and habits Zheng et al. [2010a]. In particular, providing relevant location recommendations to users of an LBSN is a popular research topic Ye et al. [2010]Bao et al. [2013]. In this thesis, I also focus on providing better time-aware location recommendations by taking into account human mobility behavior and use deep neural networks to learn complex mobility patterns and preferences of users.

In this Chapter, I will first present a broad classification of recommendation

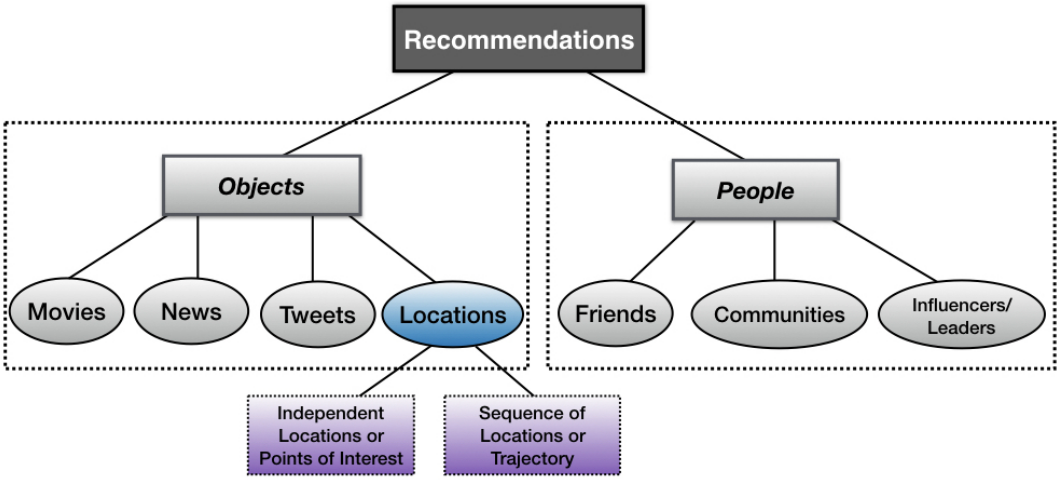


Figure 2.1. Broad classification of recommendation systems based on their objectives.

algorithms based on its objective in Section 2.2. Afterward, in Section 2.3 I describe the process of recommending locations and the different sources of information used to provide location recommendation. Further, in this section, I present the need to provide time-aware location recommendations. In Section 2.4, I present the state-of-the-art location recommendation techniques to provide a time-independent recommendation, i.e., the information provided to the user does not vary with time and time-aware recommendations i.e., the information provided to the user takes into account time at which the recommendation is being offered.

2.2 Objective of Recommendation

The objective of different recommender systems is to provide novel information to people who use the system. For example, a movie recommendation system aims to offer new movies personalized to the user. A user recommendation system provides information about potential friends (e.g., Facebook) or influential people (e.g., Twitter). As shown in Figure 2.1, I broadly classify recommendation systems based on their objective into two groups: 1) Recommend objects like Movies, Tweets, News, Locations, etc., and 2) Recommend people like friends, communities of interest, influential people of interest.

2.2.1 Recommend new objects

The primary focus here is to suggest objects tailored to user's interests. An overview of the different algorithms used in the recommender system of popular entertainment platform Netflix is presented in Gomez-Uribe and Hunt [2016]. In Koren et al. [2009] the authors provide an overview of factorization based methods used to recommend movies as part of the popular Netflix prize contest. In McFee et al. [2012], Wang and Wang [2014] the authors aim to automatically recommend songs that match a user's music preference by utilizing content-based similarity and deep belief networks. In Lu et al. [2015], Zheng et al. [2018] the authors focus to provide personalized dynamically changing news by utilizing content-based collaborative filtering methods and deep reinforcement learning framework. Smith and Linden [2017] presents an overview of recommendation algorithms used to recommend books on Amazon. In Chen et al. [2012a], Yan et al. [2012], Kim and Shim [2014] the authors focus on recommending tweets to users by utilizing graph-based models and collaborative filtering models.

A location recommender system aims to provide a list of new places to visit that are relevant to a querying user. Specifically, a location recommender system exploits both the history of locations visited by all users and social ties among them Bao et al. [2013] Zheng et al. [2010b]. These systems utilize a users' location history and their social network to recommended new places to visit Wang et al. [2013] Zhang et al. [2014]. They also take into account the geographical distance to a recommended place, while providing recommendations to a user. My research work broadly focuses on providing new locations to people. I will present a comparative state-of-the-art in the topic of location recommendation in Section 2.3.

2.2.2 Recommend new people

The objective of these recommender systems is to suggest new people to users based on their interests and social network. In Xie [2010], the author utilizes interest-based features to recommend new people to users. While in Silva et al. [2010] the authors construct a topology of the social network and propose network graph-based techniques to recommend friends and Roth et al. [2010] exploit interactions between users to build a friend suggestion algorithm. In Chen et al. [2008] authors utilize co-occurrences in social data and collaborative filtering based techniques to perform personalized community recommendations. In Sharma and Yan [2013] the authors use a pairwise logistic regression model to build an improved community recommendation model. Further, In Hannon

et al. [2010] and Armentano et al. [2011] the authors focus on recommending influential people to users on Twitter, they utilize feature-based models and content-based filtering strategies to obtain the set of influential people to recommend.

2.3 Location Recommendation

The objective of recommending location(s) to people is to provide novel places that might be of their interests and can be broadly divided into two groups: 1) Recommend Points of Interest (POI) or Independent Locations like restaurants, shopping places, etc., and 2) Recommend a sequence of locations or trajectory like a sequence of popular tourist locations in a city.

A wide range of recent works focuses on POI recommendations that utilize user check-in information to understand their mobility patterns and provide a set of top-K POI that the user is most likely to visit Wang et al. [2013]Liu et al. [2014]. The focus of this thesis is also to provide POI recommendations to users. A trajectory recommendation algorithm utilizes a richer set of information like sequences of locations or routes traveled by a user and recommend a new path or trajectory that could be of interest to users based on their preferences and temporal constraints Yin et al. [2014]Leung et al. [2011]Kong et al. [2017]Ding et al. [2013].

In the next section, I will provide an overview of the state-of-the-art algorithms in POI recommendation and different data sources used to provide relevant POI locations.

2.3.1 POI Recommendations

With the increasing usage of the smartphone in our everyday lives, people are actively using multiple mobile applications to consume a broad variety of information for entertainment and looking for opportunities to have a good experience at multiple points of interest (POIs) like restaurants, hotels, stores, etc. Different location-based social networking services, such as Foursquare, Jeipang and other services like OpenTable, Twitter, and Instagram have a wide repository of interesting POIs gathered using the large base of users who use these applications. POI recommendation addresses the problem of finding the relevant POIs for a given user based on their preferences.

Information utilized for POI recommendation in Location Based Social Network (LBSN)

A Location Based Social Network (LBSN) consists of users who visit different places and share location information with friends in their social network. An LBSN captures the spatial mobility of users by storing their location visits and also captures information related to their social ties. People typically utilize LBSN (like Foursquare) to find popular places of interests close to their current location and also to obtain places popular in their social network. In this section, I present the different types of information used to provide POI recommendations to people in an LBSN and the different datasets used to evaluate the performance of recommendation algorithms in LBSNs.

An LBSN service typically captures the following information about users and locations they visit: 1) User Check-in Information, 2) Friendship network of a user, 3) POI information like descriptive information about the location, 4) Other meta information like ratings of POIs and user reviews.

1. User Check-in Information: The check-in history of a user captures the implicit preference of a user for the locations at different times. Different recommendation algorithms utilize this information to construct user-location graphs where the edge of a graph represents the user's check-in history, and weight of the edge indicates the number of visits. Other algorithms utilize the check-in information to build user and location profiles and employ collaborative filtering techniques.

2. Friendship network of user: It captures the influence of social peers on the mobility preferences of a user. Many recommendation algorithms construct a network, i.e., a User-user graph where edges represent friendship ties in a social network. Further, the edges can also be derived from the user's location history, e.g., two users may be connected if they have visited the same location or similar types of places.

3. POI information: Every POI represents a location like a restaurant, shopping place, museum, etc. Multiple locations can be categorized based on their description and type of services offered, for example, a user who often visits an art museum can probably be recommended a new art museum. Multiple recommendation algorithms utilize this descriptive information about POIs to improve the likelihood that a user accepts the recommended set of POIs.

4. Other meta Information: Many LBSN services also enable their users to rate and provide reviews to the different locations they visit. This meta information about POIs can be used to obtain a collective opinion of people. Many recommendation algorithms utilize this meta information to update the preferences of users with the aim to provide better recommendations.

Majority of the research works evaluate their POI recommendation algorithms on data from location-based social networking websites like Gowalla Cho et al. [2011a], BrightKite Cho et al. [2011a] and Foursquare. Also, multiple research groups have constructed datasets by combining information from social networking websites like Jeipang and Twitter where users also share information about locations visited.

2.3.2 Time-Independent Vs Time-Aware Recommendations

A majority of location recommendation algorithms utilize a long history of location visits of users and recommend locations that are independent of time, i.e., the recommendations do not vary with time or provide recommendations based on the snapshot of check-in history of users.

Humans are creatures of habit and exhibit time-varying mobility patterns Noulas et al. [2011]; Cho et al. [2011b], i.e., type of locations visited in the afternoon is different from those visited in the evening. Therefore, a model that recommends new locations to people must learn the time-varying mobility behavior of a user and recommend new locations accordingly.

Multiple experiments on human mobility patterns Noulas et al. [2011]; Cho et al. [2011b] and my own analysis on different Location Based Social Networking (LBSN) datasets like Gowalla, Brightkite has shown that the preference of users for different locations can be defined as a combination of stationary and temporal preferences. Several research works have shown that the likelihood of a user visiting a particular location can be estimated based on their location history and locations visited by their friends. A Location Based Social Network (LBSN) is one that incorporates both the spatial behavior of all users and their social connections. Recommendation systems designed for LBSN focus on recommending places that are geographically close to the user. These recommendation systems can also be used to obtain a list of nearby potential customers. Majority of recommendation systems in LBSN do not take into account temporal aspect of recommendation, e.g., recommendation for a restaurant has higher priority in the evening compared to a library. Further, existing recommendation systems are not designed to provide continuous recommendations to mobile

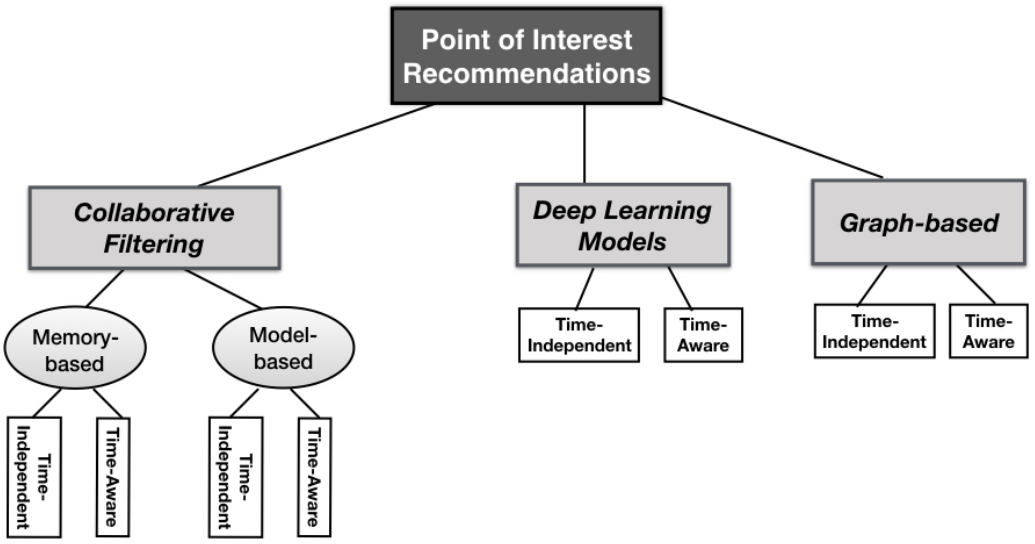


Figure 2.2. Broad classification of techniques to recommend Points of Interests.

users; a real-world need where people want up-to-date recommendations based on their current location.

A real-world model must be able to provide location recommendations to a wide range of people based on their personal preferences and also continuously learn from their temporal mobility patterns. Further, the location provided to people must be relevant to the time at which the user is being recommended.

2.4 Different POI Recommendation Techniques

In this section, I present the literature review of different state-of-the-art techniques to recommend POIs, the information they utilize to provide recommendations and their limitations. Further, I categorize the recommendation techniques into two groups: 1) Time-independent and 2) Time Aware, described in Section 2.3.2. Finally, I describe how the different recommendation models presented in this thesis perform compare to other techniques.

I categorize the major techniques used by POI recommender systems in LBSNs into three groups as shown in Figure 2.2.

2.4.1 Collaborative Filtering Techniques

The most popular POI recommender systems utilize variants of Collaborative Filtering (CF) techniques to recommend places by learning from the location history of users (Adomavicius and Tuzhilin [2005]Linden et al. [2003]Sarwar et al. [2001]Su and Khoshgoftaar [2009]). These systems aim to learn the preferences of users from their location history and provide a recommendation based on the fact that a user is more likely to visit a location preferred by similar users. Collaborative filtering techniques utilize a user-location matrix to represent the preferences of users for POIs. It then matches users with relevant interest and preferences by calculating similarities between their profiles to make recommendations Mooney and Roy [2000].

Further, CF techniques can be broadly grouped into two types: 1) Memory based techniques that construct a profile of each user and location based on the check-in history and utilize this information to find locations visited by similar users, 2) Model based techniques utilize the check-in history of all users and aim to build a model that can capture the mobility preference of all users for known locations visited by the users.

Memory based

Recommendations provided by memory based collaborative filtering algorithms utilize information whether a user has visited a location in the past. Based on the past visits of a user, the similarity of users or locations is used to create new POI recommendations. The two most commonly used memory based filtering methods are User-based and Location-based collaborative filtering Bao et al. [2013]. The idea of User-based collaborative filtering (UserCF) is that similar users have similar preferences on locations. Location-based collaborative filtering (LocCF) assumes that people with similar preferences visit similar locations.

- **Time independent recommendations**:

  Majority of state-of-the-art memory based collaborative filtering techniques are designed to provide recommendations that do not change with time. One of the first work for POI recommendation is proposed by Ye et al. [2010]. The authors consider Friend-based Collaborative Filtering that incorporates the social influence among users under the assumption that people are more likely to visit locations suggested by their friends. The work in Ye et al. [2011] propose a method that also takes into account the geographical influence along with the user preference. They use a power-law

distribution model to capture geographical influence and the user preference derived by a user-based CF approach. Further, Ference et al. [2013] propose a modified collaborative filtering technique to recommend locations for out of town users. Their technique takes into account three things: user preferences, the proximity to recommended place and social relationship with users who have visited the places before. All of these works do not take into account temporal importance of recommended locations i.e., they do not distinguish between old and new (in time domain) popular locations.

- **Time aware recommendations**:

  Very few memory based collaborative filtering techniques incorporate temporal information in the user check-ins at POIs and provide time aware recommendations. In Yuan et al. [2013], authors propose an enhanced user-based collaborative filtering method by considering the similarity between users at different time intervals. If two users have similar temporal behavior, they are likely to visit similar locations at the same time. One of the recent work done by Zhang and Chow [2016] utilize temporal correlations in user visits for weekdays and weekends to provide time-aware location recommendations. These works do not take into account social influence of users.

Memory based techniques heavily rely on similarity measures like Cosine similarity to find potential POIs to recommend. In order to compute similarity, memory based techniques need to process the entire dataset of all user-location pairs every time a recommendation is to be provided; this makes them unreliable and inaccurate for a new user who has not visited any location before. However, the benefit of memory-based techniques is that they are easy to implement.

The REGULA recommendation algorithm proposed in this thesis (described in Chapter 3) also falls under the category of memory based collaborative filtering. REGULA utilizes features based on human mobility to recommend new locations to users. To the best of my knowledge, REGULA is the first model that utilizes the Frequently Visited Locations (FVLs) of a user to recommend the new location and use the concept of regions or bounding boxes around FVLs to capture new locations. Further, REGULA is the first model to capture the temporal visiting patterns of users and their friends with the help of Temporal and Friendship features.

Model based

Memory based techniques aim to compute the relationship between users and locations. Model based methods try to infer the preferences of users and locations by utilizing machine learning algorithms Koren et al. [2009]. Each user and location is represented as a vector of factors that are determined based on the known POI visits of users. Matrix Factorization (MF) is one of the most popular recommendation models that have shown to perform very well Koren et al. [2009]; Lian et al. [2017]. MF based models have become popular after having won the Netflix Prize[1]. The underlying assumption of MF based models is that the interaction between users and locations can be captured by the interaction between their latent factors or feature vectors. MF based model has been shown to perform the best in recommending locations to people.

- **Time independent recommendations**:

  The objective of different MF based models is to approximate the values in a user-location matrix by considering the dot product of user and location factors Koren et al. [2009]. These works only consider the weighted frequency of visits in constructing the user-location matrix. The corresponding factors are the simple interaction between users and locations. Other MF based techniques extend the user-location matrices by considering and building matrices that capture additional information like time of visits and meta information about locations Lian et al. [2017]He et al. [2016].

  LibFM Rendle [2012] is one of the standard and benchmark factorization based model that characterizes users and locations by a vector of factors inferred from the user-location visits. The vectors are in the same latent space and, the preference of a user for locations is modeled as inner products in that space. SVD++ Koren [2008] is another factorization based model that also incorporates the similarities between different items or locations. In IRenMF Liu et al. [2014], authors exploit the distribution of check-ins in a geographical neighborhood and provide recommendations by utilizing Weighted Matrix Factorization Hu et al. [2008]. IRenMF uses the idea of POIs in the same geographical region may share similar user preferences. An excellent piece of work done by Lian et al. [2014] propose GeoMF, a factorization based model that augments the user's and location's latent factors to incorporate the spatial constraints. GeoMF outperforms memory based recommendation models like UCF and other factorization based models.

[1] https://www.netflixprize.com/

Recently, multiple variants of the factorization based model have been proposed that utilize auxiliary information like as texts (user reviews), images and videos (user-created content) to obtain better representations of user and location factors Gao et al. [2015]Zhang et al. [2016b]Lian et al. [2015]Lian et al. [2018].

In this thesis, one of the proposed recommendation model `DEEPREC` (described in Chapter 5) has outperformed factorization based recommendation models like GeoMF, SVD++, and LIBFM.

- **Time aware recommendations**:

  The factorization based models described in the previous section do not consider temporal latent factors to capture the time-varying preferences of users. One of the first factorization based model that incorporates temporal dynamics is TimeSVD++ proposed by Koren [2010] and has shown strong results at Netflix contest. In Gao et al. [2013], the authors utilize a temporal factor matrix to capture the preference of users corresponding to every hour of the day. Wang et al. [2015c] propose a hybrid MF based predictive model that integrates both the regularity and conformity of human mobility. One the recent time-aware factorization model Regularized Content-Aware Tensor Factorization (RCTF) proposed by Lian et al. [2016] augments the user's and location's latent factors to incorporate the spatial constraints.

  In this thesis, the proposed time-aware recommendation model `JOINTDEEPREC` (described in Chapter 6) has outperformed state-of-the-art time-aware factorization based recommendation models RCTF and TimeSVD++.

In contrast to memory-based CF techniques, model-based CF techniques have a time-consuming learning phase. But model-based techniques can quickly recommend a set of locations using a pre-computed model. Some of the common problems highlighted with collaborative filtering techniques are: 1) Cold-start problem, i.e., inability to recommend a location to a new user, 2) Data sparsity problem, i.e., lack of enough information about visited POIs

2.4.2 Graph Based Techniques

Graph based techniques represent the location history of users and their social ties as an LBSN graph where nodes represent users/locations, and edges represent user's preference to a location or social ties among users or similarity

between two locations. The goal of these techniques is to utilize graph based algorithms to process the entire LBSN graph to compute different types of similarity scores Kefalas et al. [2016].

- **Time independent recommendations**: PageRank Page et al. [1999] and Hypertext Induced Topic Search (HITS) Chakrabarti et al. [1998], are widely used link analysis algorithms used to rank web pages. These algorithms extract influential nodes from a complex network by analyzing the structure. Zheng et al. [2009] extend the HITS algorithm for discovering popular users and interesting locations in an LBSN graph that is constructed using user's social network and check-in history of users. Wang et al. [2013] construct an LBSN graph with edges that captures both friendship ties among users and similarity between users based on their location visits. The authors propose a Location-Friendship Bookmark-Coloring Algorithm (LFBCA) to recommend new locations to users. LBSNRank Jin et al. [2012] is another piece of work that incorporates the location history of users and their friends by constructing a personalized PageRank algorithm for each user. The work done by Zhang et al. [2014] represent the sequential mobility patterns of users as a dynamic Location-Location Transition Graph and exploit these sequential movement patterns of users to provide better recommendations in LBSN.

  The REGULA recommendation algorithm proposed in this thesis (described in Chapter 3) has also been compared with LFBCA algorithm proposed by Wang et al. [2013] and provides better location recommendation.

- **Time aware recommendations**:

  There are very few graphs based recommendation algorithms that provide time aware recommendations. In Yuan et al. [2014] the authors present a time aware recommender system (GTAG-BPP) by constructing a geographical-temporal influence graph that encodes both geographical and temporal information of user check-in records. The authors use a breadth-first preference propagation algorithm to provide time-aware POI recommendations.

The major drawback of graph based algorithms is that they are computationally expensive compared to memory based collaborative filtering techniques as they need to process the entire LBSN graph.

2.4.3 Deep Neural Network based Models

Recent advances in the parallel processing hardware like graphic cards have enabled researchers to perform large-scale computations and has lead to the success of deep neural networks in modeling complex patterns in data Goodfellow et al. [2016]. Specifically, in the domain of recommendations, neural network models can learn personalized preferences of users from very large data-set of user check-ins. Recently, MF based models used together with deep neural networks have outperformed state-of-the-art models in recommending new movies/items Zhang et al. [2016a]Wang et al. [2015b]Wu et al. [2016b]. These models utilize deep learning tools to learn features of auxiliary information like textual content, video. The learned features are used as additional features in the overall feature matrix. These additional features are shown to improve in obtaining better latent representation for user and item/locations and thereby able to provide better recommendations.

The list of widely used deep neural networks are as follows:

- Multilayer Perceptron (MLP) is a feed forward neural network with three different layers of artificial neurons: the input layer, one or more hidden layers, and an output layer. The artificial neurons used are different nonlinear activation functions like sigmoid, tanh or perceptron Rosenblatt [1962].

- Auto-Encoder is an unsupervised learning model that aims to learn an encoded representation of data. It learns by compressing data from the input layer into an encoded vector and then un-compresses the vector to produce an output that closely matches the input data. Some variants of autoencoders are denoising autoencoder, marginalized denoising autoencoder and variational autoencoder Vincent et al. [2008]Bengio et al. [2009] Chen et al. [2012b].

- Convolutional Neural Networks (CNNs) are typically used in the field of image processing and are made up of neurons that learn different patterns in images using different convolution filters. The neural network is made up of different convolution layers and pooling operations that capture global and local features. It performs well in processing data with grid-like topology Goodfellow et al. [2016].

- Recurrent Neural Network (RNN) is a class of neural network that allows cyclical connections between input and output Goodfellow et al. [2016]. Unlike feed forward neural network model where the output of neurons in

one layer is given as input to neurons in next layer and neurons in an RNN layer have cyclical connections, i.e., the current output of neuron depends and the historical input and output of the neuron. Further, RNNs have memory that enables them to capture historical states of neurons and is referred to as hidden state of the RNN. It is suitable for modeling sequential data. RNN variants such as Long Short Term Memory (LSTM) Hochreiter and Schmidhuber [1997] and Gated Recurrent Unit (GRU) Chung et al. [2015] network are two widely used model deployed in the real world.

Neural networks have been used to recommend multiple objects like music, news, images, books, tweets Zhang et al. [2017]. They have primarily been used to learn the latent factors of users and items that best represent the ratings of all items due to all users Wu et al. [2016a] Wang et al. [2015a] Li et al. [2015b]. The recommendation model proposed by He et al. [2017] utilize a Feed Forward Neural Network to learn the latent factors of users and items. A majority of these neural network based models were designed and evaluated to recommend items like movies. However, recommending locations entails some unique challenges than compared to recommending items. The problem of location recommendation has some unique and different characteristics compared to other online recommender systems. In traditional recommender systems like recommending movies, or products on a website, the goal is to provide a list of movies/items based on the profiles and previous behavior of the user. Whether a movie/item is of interest to the user is validated by whether the user has clicked that particular online movie/item. So the feedback and behavior of user towards the movie/item are much faster, and the cost of accepting the recommended movie/item by the user is low. In the case of location recommendation, the probability that a user visits a recommended location depends on multiple factors like: (i) what is the current location of the user, (ii) how far is the recommended place, (iii) what is the time of recommendation, (iv) if the user is alone or with other people with different interests, etc... Also, the feedback related to whether a recommendation was useful or not is delayed due to the physical distance travel by the user to the recommended place compared to just a click for a recommended movie/item. Therefore, recommending locations to a user poses some unique challenges compared to recommending a movie/item online.

The application of neural networks to the problem of recommending POIs is a still a growing research area. Therefore, in this Section, I will present recent efforts done in recommending POIs to users that utilize deep neural networks.

The most recent work done by Wang et al. [2017] use the idea that photos reflect user interests and also provide informative descriptions about locations

they visit. The authors utilize images shared by users in Instagram to obtain personalized preferences of users that can be used to enhance the performance of POI recommendations. They propose a visual content enhanced POI recommendation framework that uses a convolutional neural network to extract features from images and use it to guide the learning process of latent user and POI features. Liu et al. [2016] extend a recurrent neural network (called ST-RNN) to incorporate local temporal and spatial contexts in each layer and learn the time-specific and distance-specific transition matrices. Another recent work done by Xia et al. [2017] utilize a feed forward neural network with softmax output that jointly learns the latent factors of users and POIs to predict user preference over POIs under various contexts.

The POI recommendation algorithm `DEEPREC` and `JOINTDEEPREC` proposed in this thesis also utilize the RNN neural network to learn the locations preferences of users to provide time-aware recommendations (described in detail in Chapters 56). To the best of my knowledge, `DEEPREC` and `JOINTDEEPREC` models are the first to incorporate spatial and temporal constraints in the mobility of users. Finally, I summarize different key research works done to address the problem of POI recommendation in Table 2.1.

2.4.4 Evaluation Method

The widely used evaluation method is by dividing the location history into two parts: 1) Location history of users up to a particular point in time and 2) the rest of the user's location history Zhang et al. [2017]. The first part of the data is used to train the location recommendation model and the second part is used to evaluate the performance of the model based on two commonly used metrics: Precision and Recall. I will present the different evaluation metrics in the next chapter.

2.5 Summary

From the above literature review about the goal of recommendation algorithms, the different kinds of objects and people that are recommended and the broad classification of location recommendations, It is evident that there is a need for better time-aware location recommendations that consider different aspects of LBSN. From the above literature review, we also found that deep neural networks can learn time-independent and time-aware preferences of users at scale due to their fundamental strength to learn efficiently and optimally from observational

Table 2.1. Comparison of state of the art POI recommendation techniques.

Research Research Work	Recommendation Recommendation Technique			Type of Information used			Performance based on direct comparison	
	Collaborative Filtering	Deep Neural Networks	Graph Based	Time-Aware	Spatial Information	Social Information	Ranking Based on Precision	Ranking Based on Recall
UCF	✓	×	×	×	×	×	4	4
LCF	✓	×	×	×	×	×	3	3
LFBCA	✓	×	✓	×	×	✓	2	2
REGULA	✓	×	×	×	✓	✓	1	1
SVD++	✓	×	×	×	×	×	4	4
LIBFM	✓	×	×	×	×	×	3	3
GEOMF	✓	×	×	×	✓	×	2	2
DEEPREC	×	✓	×	×	✓	×	1	1
TIMESVD++	✓	×	×	✓	×	×	3	3
RCTF	✓	×	×	✓	✓	×	2	2
JOINTDEEPREC	×	✓	×	✓	✓	×	1	1

data. Therefore, the focus of this thesis is also to utilize a deep neural network to provide a relevant recommendation by learning the time-independent and time-varying locations preferences of users. In the next Chapters, I will present my three recommendation models: A collaborative filtering based recommendation model called REGULA that also takes into account temporal aspects, and two deep learning based recommendation models called DEEPREC and JOINTDEEPREC that provide time-independent and time-aware location recommendations.

Chapter 3

REGULA: A Feature based Recommendation Model

3.1 Introduction

In this chapter, I present my analysis on two standard LBSN datasets to show that humans exhibit regular mobility. I describe in detail the 5 analytical tests conducted to confirm such regular behavior. Utilizing the finding from these tests, I present 3 features that capture the regularity in human mobility. Later, I describe how I utilize these 3 features to build my recommendation model: REGULA , that can be used to recommend locations to users in an LBSN.

The recommendation process broadly follows two steps: 1) I categorize all users of an LBSN into 4 groups based on the availability of their mobility and social information; 2) for each group, I use different variants of REGULA to provide recommendations.

In this chapter, I will also present the evaluation of REGULA on two standard LBSN datasets and measure its performance based on two standard metrics: Precision and Recall. I will also present a comparative performance of REGULA with other state-of-the-art recommendation models.

This chapter is structured as follows: In Section 3.2, I present the characteristics of the two standard LBSN datasets used in my evaluation. Section 3.3 describes the five different analytic tests I conducted to show that humans exhibit regular mobility behavior. Section 3.4 presents the 3 features that REGULA recommendation model is built on. Section 3.5 presents the evaluation of REGULA based on two performance metrics and compare it with other recommendation models. Section 3.7 concludes the Chapter.

3.2 LBSN Datasets

A Location Based Social Network (LBSN) consists of users who visit different places and share information about such places with friends in their social network. An LBSN captures the spatial mobility of users by storing their location visits and also captures information related to their social ties. In this thesis, I have evaluated my REGULA recommendation model on two standard LBSN datasets.

3.2.1 Gowalla

Gowalla was a popular location-based social networking website that was acquired by Facebook in 2011. It had a user base of more than 600,000 and was popular between 2007 to 2012. Users were able to check into locations using a dedicated mobile application or through the mobile website. The website enabled users to share their location check-ins with friends.

The Gowalla dataset contains 6.4 million check-ins and was collected by Cho et al. [2011a] between February 2009 and October 2010. The friendship network is un-directed and was collected using a public API, and consists of 196,591 nodes with 950,327 edges.

3.2.2 Brighkite

Brighkite was a location-based social networking service provider that operated between 2007 and 2011. The service enabled users to "check in" at places using a mobile application or a text message. The service also enabled users to see who is nearby and who visited the check-in location recently.

The Brightkite dataset contains 4.4 million check-ins and was also collected by Cho et al. [2011a] between April 2008 and October 2010. Brighkite's public API was used to collect an un-directed friendship network, and consists of 58,228 nodes with 214,078 edges.

Table 3.1 presents the characteristics of above two standard LBSN datasets: Gowalla and Brightkite

Table 3.1. LBSN Dataset Characteristics

	Gowalla	**Brightkite**
Time period of Check-ins	569 Days	929 Days
Number of Users	196,591	58,228
Number of Locations	1,280,956	772,933
Number of Check-ins	6,264,203	2,627,870
Number of Friendship links	950,327	214,078

3.3 An Experimental Analysis to understand human mobility

Multiple experiments conducted by several groups e.g., Gonzalez et al. [2008a]; Papandrea et al. [2013] have captured the real world mobility patterns of humans using the GPS in smart phones carried by people. These experiments show that humans regularly visit same set of places and often at the same time i.e., humans follow simple reproducible patterns. Based on these findings, I formulate five hypothesis and test whether users of an Location Based Social Network (LSBN) also exhibit regular mobility behavior.

H1: Regularity Users regularly (or habitually) visit a set of locations i.e., their Frequently Visited Locations (FVLs).

H2: Vicinity Users visit places in the vicinity of their FVLs.

H3: Recency Users are more likely to go places that were visited recently by others.

H4: Sociality Users are more likely to go places that were visited by their friends.

H5: Inertia Users are more likely to go places geographically close to their present location.

I have conducted analytic tests to check the validity of these hypothesis for users in Gowalla and Brightkite datasets.

3.3.1 Hypothesis 1: Regularity

Large scale studies Gonzalez et al. [2008b] on human mobility patterns have shown that humans exhibit regular mobility patterns i.e., they visit a few set of

locations repetitively like favorite pizzeria, McDonald's near home etc. I refer to any location that a user has visited more than once as one of their relevant Frequently Visited Location (FVL).

The two datasets described in Section 3.2 contains check-ins collected over a long period ranging from 569 Days to 929 Days. In order to test the validity of my hypothesis, I conducted tests at different days in a sequential manner. For example, a test conducted on 30th day will take into account all check-ins during the first 30 days. For each user with a check-in in the first 30 days, I find out if they have any FVL. The goal of this test is to measure what fraction of users have one or more FVLs.

For Gowalla and Brightkite datasets the tests were conducted in an interval of 30 days i.e., for Gowalla, the analysis was conducted at days $t = \{90, 120, .., 510\}$ and for Brightkite at $t = \{90, 120, .., 870\}$. Figure 3.1 shows the fraction of users with atleast one FVL and their distribution at different testing times in Gowalla and Brighkite. I observe that a significant fraction of users regularly visit atleast one or more set of locations. Further, in Gowalla and Brightkite I observe that the fraction of users with atleast one FVL increases with time. This suggests that even as more check-in information is obtained, still the users tend to regularity visit same set of locations.

3.3.2 Hypothesis 2: Vicinity

Several human mobilty experiments (e.g., Gonzalez et al. [2008a]; Papandrea et al. [2013]) has found that people are unwilling to travel long distances from their current location. These studies show that majority of next locations visited are within a range of 10 km from the present location. Based on these studies, I hypothesize that people usually tend to explore new locations that are close to their frequently visited locations like, places close to their favorite pizzeria etc.

Similar to the analysis done in Section 3.3.1, I test this hypothesis in a sequential manner at different times (t). For each test, I measure the fraction of users that have a new check-in within a distance of 1 km / 2.5km / 5 km from their FVLs. Figures 3.2, 3.3, 3.4 present the fraction of users who have visited atleast one new location within a range of 1 km / 2.5km / 5 km of their FVLs in Gowalla and Brightkite. The fraction of users that visit locations close to their FVLs keeps growing with time. Therefore, I can utilize this observation to provide correct recommendations to users by suggesting them new locations around their FVLs. Further, If a place is popular then for the majority of users, the popular location would not be part of the new locations around FVL. Only for a small fraction of overall users will the popular location could be part of the new location around

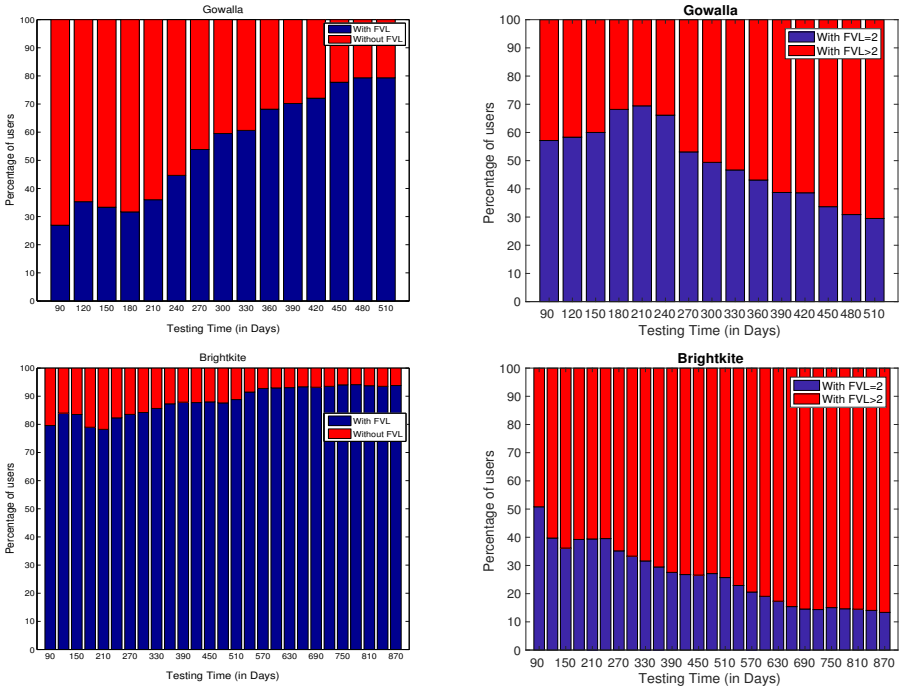


Figure 3.1. Hypothesis 1: Regularity - Fraction of users with atleast one FVL at different times in Gowalla and Brightkite LBSN datasets

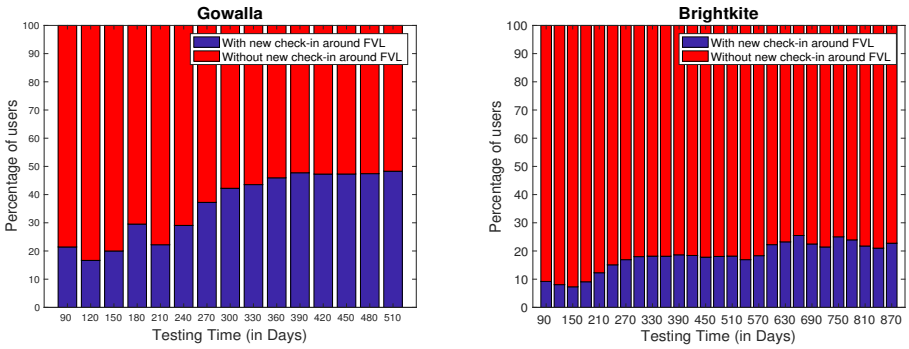


Figure 3.2. Hypothesis 2: Vicinity - Fraction of users with new check-ins within a distance of 1km from their FVLs at different times (Days) in Gowalla and Brightkite LBSN datasets.

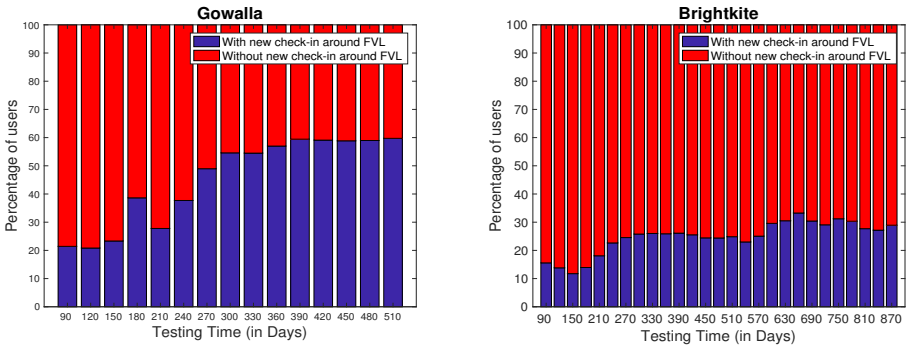


Figure 3.3. Hypothesis 2: Vicinity - Fraction of users with new check-ins within a distance of 2.5km from their FVLs at different times (Days) in Gowalla and Brightkite LBSN datasets.

FVL. The overall impact of this small fraction of users on the distribution of users that have a new check-in within a distance of 1 km / 2.5km / 5 km from their FVLs is low. Therefore it is not a confounding factor.

3.3.3 Hypothesis 3: Recency

I hypothesize that people go to new places that were recently visited by others. I test this hypothesis in three steps: a) For each location visited by a user, I find out the first time the user visited it, say t_1, b) For this location, I find out when did any other user visit it, say t_2, c) I state the user visited a new location that was most recently visited by somebody else m $(= t_1 - t_2)$ days back. The goal of this test is to measure the influence across time. The maximum possible value of

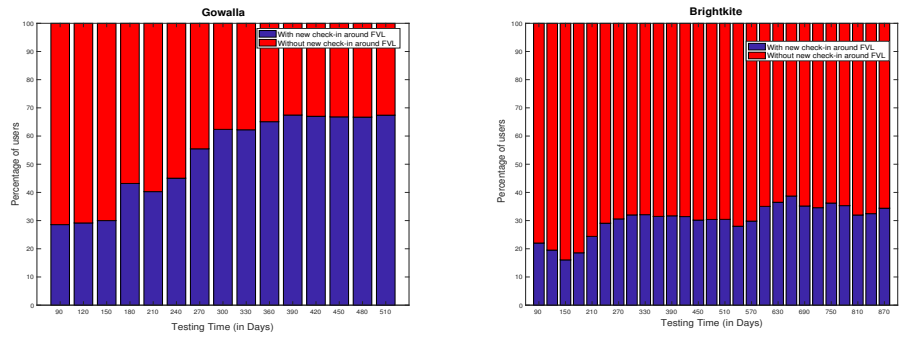


Figure 3.4. Hypothesis 2: Vicinity - Fraction of users with new check-ins within a distance of 5km from their FVLs at different times (Days) in Gowalla and Brightkite LBSN datasets.

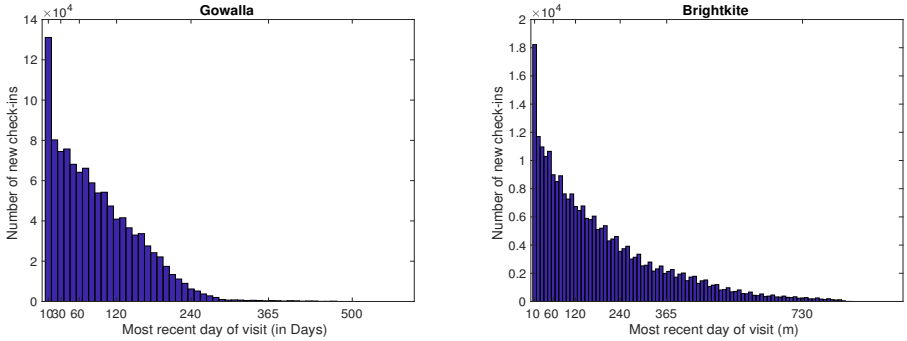


Figure 3.5. Hypothesis 3: Recency - Most recent day of visit for all new check-ins

m for users of Gowalla and Brightkite is their time period of check-ins i.e., 569 days and 929 days respectively (see Table 3.1).

Figure 3.5 presents the most recent day of visit for all new check-ins in Gowalla and Brightkite datasets. I observe that a significant number of users go to places that were visited by others in the last 30 days. Therefore I can say that users mostly ignore locations that were visited a long time back. This analysis shows the importance of time while providing more precise recommendations.

3.3.4 Hypothesis 4: Sociality

Humans are social animals and heavily influenced by their friends and family. I hypothesize that people go to new places were their friends have been recently. I conducted tests to check whether a user is affected by their social network

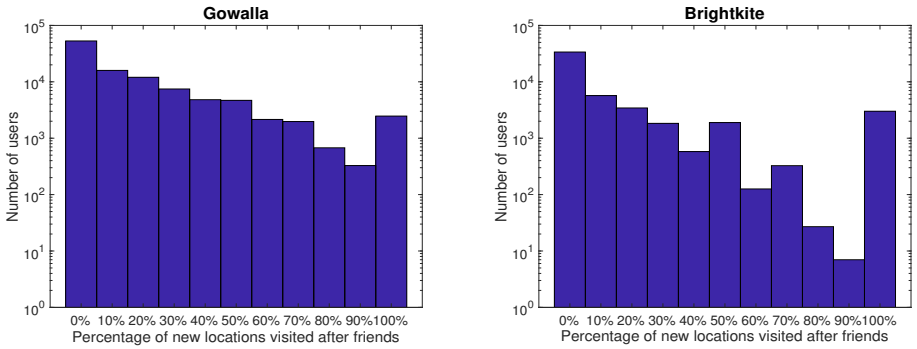


Figure 3.6. Hypothesis 4: Sociality - Distribution of users who have visited a new location after their social friends.

friends. For each user, I find out the total number of unique new location visits and then measure what percentage of them were recently visited by their social friends. Figure 3.6 presents how friends influence new places visited by a user in Gowalla and Brightkite datasets. I observe that a significantly large number of users have at least one new location that was previously visited by their social friends.

3.3.5 Hypothesis 5: Inertia

My final hypothesis is that people exhibit a lot of inertia and often tend to go nearby places. For example, tourists are more likely to eat close to the monuments or other attractions they visit. Thus, in addition to my hypothesis that people move closer to their FVL (Hypothesis 2: Vicinity), I argue that, when they receive a recommendation, they are likely to go there if the place is geographically close to their present location. I conducted tests where, for each new location visited by a user in our CDR-based LBSN dataset, I measure its geographical distance from locations visited before to see whether they exhibit inertia or not.

Figure 3.7 presents a distribution of distance traveled for a new location visit by all users of Gowalla and Brightkite. We observe that more than 84% of new locations visited by users in Gowalla and Brightkite are within a range of less than 10km. It confirms my hypothesis that people prefer going to places geographically close to their present location.

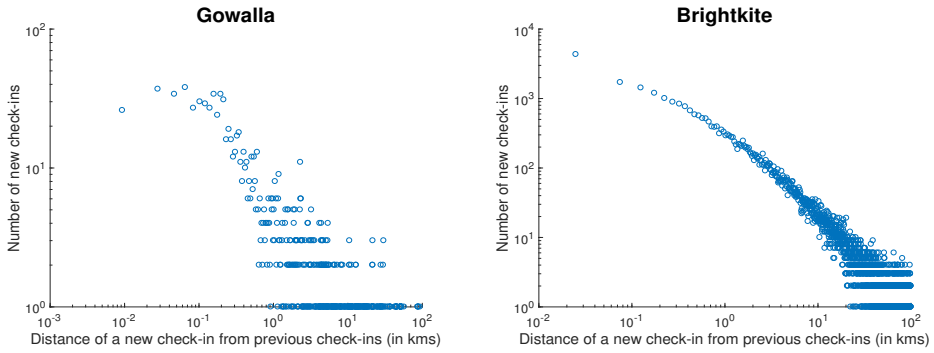


Figure 3.7. Hypothesis 5: Inertia - Distribution of how far do people travel to visit new locations.

3.3.6 Observations based on Tests

Having validated my hypothesis on Gowalla and Brightkite LBSN datasets. I formulate, from the five hypothesis, the following observations.

Table 3.2. Observations based on Tests

Observation 1	People regularly visit a certain set of places (i.e., frequently visited locations)
Observation 2	People usually visit places in the vicinity of their frequently visited locations.
Observation 3	People usually visit places recently visited by others.
Observation 4	People usually go to places visited by their friends.
Observation 5	People usually go to places close to their own recently visited places.

3.4 REGULA Recommendation Model

Using the five observations presented in Section 3.3.6, I developed 3 features that are used by my recommendation algorithm `REGULA` . Table 3.3 presents the symbols that will be used to describe `REGULA` model and different features.

Table 3.3. Symbols used to define REGULA

Symbol	Description
U, L, T	user set, location set, check-in time set (in days)
u, l	user $u \in U$, location $l \in L$
$t_{(u,l)}$	time at which u visited location l, $t_{(u,l)} \in T$
C	set of all check-ins $\{< u, l, t_{(u,l)} >\}$
E_f	set of all friendship links between U users
G	friendship graph of U users and E_f edges
VL_u	unique locations visited by user u, $VL_u \subset L$
VL_u^N	last N locations visited by user u, $VL_u^K \subset VL_u$
F_u	friends of user u, $F_u \subset U$
RT	time of recommendation (in days)
K	number of recommended locations
$ts(u,l)$	temporal feature value assigned by user u to location l
$ts(l)$	aggregated temporal feature value of location l
TS	set of aggregated temporal feature value for all L
$ds(u,l)$	distance feature value assigned by user u to location l
$fs(u,l)$	friendship feature value assigned by user u to location l
$rs(l)$	recommendation feature value assigned to location l

3.4.1 Features of REGULA model

Given the check-in history C of U users at L locations, the goal of REGULA is to recommend a list of N new locations (out of L) to any user u. Every location $l \in L$ is assigned a feature vector that is a combination of three features: 1) Temporal Feature, 2) Distance Feature, and 3) Friendship Feature. These features are described in detail in the following sections.

Temporal Feature

The temporal feature value assigned to a location l due to the visit of user u is given by the following equation:

$$ts(u,l) = \begin{cases} \frac{t_{(u,l)}}{RT} & l \in VL_u \\ 0 & l \in L \setminus VL_u \end{cases} \tag{3.1}$$

Let us assume that we have the check-in history of a user $v \in U$ and we know that v had first visited location $p \in L$ at time $t_{(v,p)}$ and later visited $q \in L$ at time $t_{(v,q)}$ ($t_{(v,p)} < t_{(v,q)}$). Based on the *Observation 3* (refer to Section 3.3.6),

the recently visited location q must be assigned a higher temporal feature value compared to location p. Temporal Feature value assigned using Equation 3.1 ensures that the value assigned to location q is greater than location p because $t_{(v,q)} > t_{(v,p)}$. Therefore, my generalized temporal feature value assignment ensures that for each user, the locations that were visited earlier are assigned smaller values compared to recently visited locations.

I compute the aggregated temporal feature value of a location l due to visits of all users using the following equation:

$$ts(l) = \sum_{\forall u \in U} ts(u,l) \tag{3.2}$$

The idea for aggregated temporal feature was derived from the tests conducted on Recency (H3 3.3.3) . The goal was to ensure that locations that were visited in the recent past must be assigned a higher preference that compared to locations that were visited in the distant past.

Distance Feature

Utilizing *Observation 5*, I designed a feature that captures distance from the last few check-in of a user. Let VL_u^N be the set of *last N* locations visited by user u. The distance feature value assigned to location l by user u is given by the following equation:

$$ds(u,l) = \begin{cases} 0 & l \in VL_u \\ \frac{1}{\min_{l' \in VL_u^N} \text{dist}(l',l)} & l \in L \setminus VL_u \end{cases} \tag{3.3}$$

Where, dist$(l',l)$is the Euclidean distance between locations l' and l

Let $p \in L \setminus VL_u$ and $q \in L \setminus VL_u$ be two un-visited locations of user u. Let $dist(lastN, p)$ and $dist(lastN, q)$ be the closest distance from the set of *last N* locations visited by user u (or VL_u^N) to locations p and q. Based on tests conducted to measure how far do people travel (H5: Inertia 3.3.5), we know that people travel to locations geographically close their current location. If $dist(lastK, p) < dist(lastK, q)$ then Equation 3.3 ensures that distance feature assigned to location p is higher than compared to location q.

My distance feature assignment ensures that new locations closer to the *last K* visited locations of user u are assigned higher distance feature values compared to other locations.

Friendship Feature

Based on *Observation 4*, we know that people are more likely to visit places that were recently visited by their friends. I designed a feature that captures the check-in history of friends. The friendship feature value assigned to a location l by friends of user u is given by the following equation:

$$fs(u,l) = \begin{cases} 0 & l \in VL_u \\ \sum_{v \in F_u} (\frac{t_{(v,l)}}{RT} \cdot \alpha^{\frac{t(v,l)}{RT}}) & l \in L \setminus VL_u \end{cases} \tag{3.4}$$

Where, $\alpha > 0$, is a constant weighting factor

The goal of the friendship feature is to capture the influence of friends in assigning importance to a potential un-visited location. For example, let's assume that user A has 3 friends who have visited a location X, 5 days ago, and the same 3 friends have also visited a location Y, 50 days ago. Assigning a value based only on number of friends will not work because both locations X and Y have been visited by the same 3 friends. However, when we include the time dimension we know that since location X was recently visited compared to location Y. The friendship feature value assigned to location X must be higher than compared to location Y. Equation 3.4 ensures that friendship feature value assigned to locations recently visited by friends of a user is higher than compared to locations that were visited by friends in the distant past. Finally, my friendship feature value computation ensures that locations not visited by user u, but visited by its friends are assigned higher scores compared to locations that were not visited by the user.

Finally, the combined feature value (or recommendation score) assigned to each un-visited location l of user u is an accumulation of Temporal, Distance and Friendship features and is given by the following equation:

$$rs(u,l) = \ ts(l) + fs(u,l) + ds(u,l) \qquad l \in L \setminus VL_u \tag{3.5}$$

3.4.2 Overall Procedure to recommend locations using REGULA

The overall process of recommending new locations to a user using `REGULA` can be broken down into 5 broad steps and is presented in Figure 3.8.

Step 1: All users are divided into four different types based on the availability of their check-in history and friendship information. Section 3.4.3 present the details related to this categorization.

Step 2: For each user, I obtain a list of their Frequently Visited Locations. The FVLs captures the locations that are important to the user. The idea of utilizing FVLs was derived based on the insights from test on *Regularity* (Section 3.3.1).

Step 3: Fix a bounding box around each one of the FVLs. The box represent regions of interest around each FVL. Later, from each bounding box region obtain the list of un-visited locations of the user. This idea of bounding box was derived based on the insights from test on *Vicinity* (Section 3.3.2).

Step 4: For each un-visited location:

- a Compute its Temporal feature value (Section 3.4.1). I designed this feature based on insights from test on *Recency* (Section 3.3.3)
- b Compute its Distance feature value (Section 3.4.1). I designed this feature based on insights from test on *Inertia* (Section 3.3.5).
- c Compute its Friendship feature value (Section 3.4.1). I designed this feature based on insights from test on *Sociality* (Section 3.3.4).

Step 5: The combined feature value assigned to each un-visited location is the sum of Temporal, Distance and Friendship feature values. Finally, un-visited locations with highest combined feature value are given as a recommendation to the user.

In next section, I will present how all users are categorized into four different types and how different variants of `REGULA` is used to recommend locations to each type of user.

3.4.3 Variants of REGULA

Table 3.4 presents how each user is categorized into one of the four types based on the availability of prior check-in information and friendship information. In order to provide recommendation to a user we might or might not have prior check-in information. Also, for a user we might or might not have information about their friends. Therefore, depending on the availability of a user's prior check-in or friendship information, each user is divided into one of the four types.

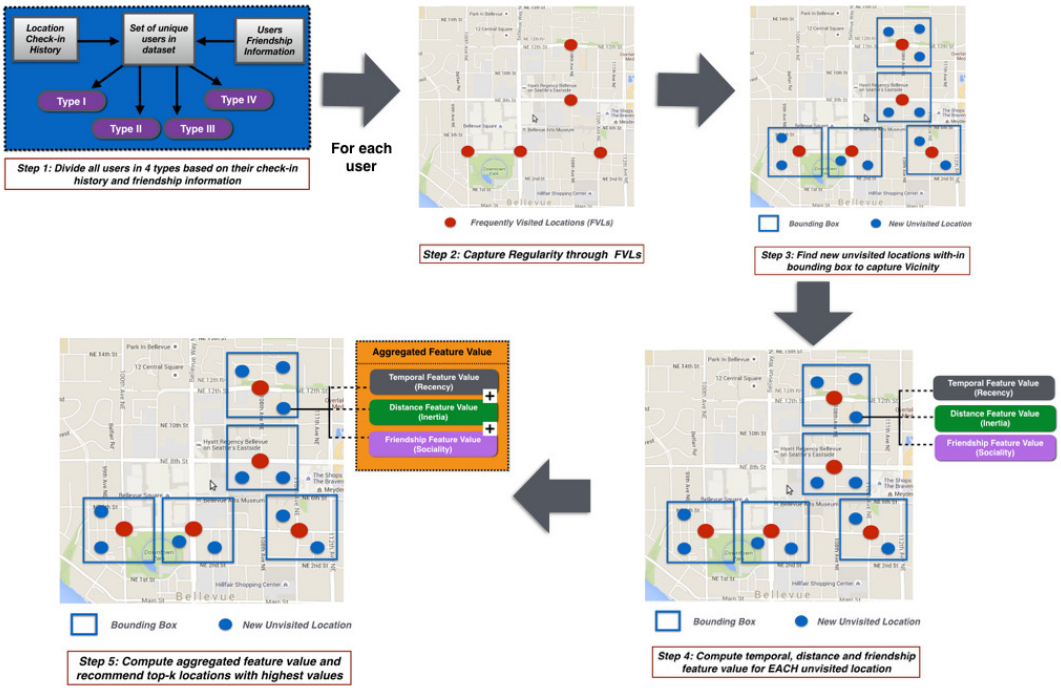


Figure 3.8. The overall process to recommend locations using REGULA

Table 3.4. Type of users in LBSN

	Is prior check-in info available?	Is prior friendship info available?
Type-I	NO	NO
Type-II	NO	YES
Type-III	YES	NO
Type-IV	YES	YES

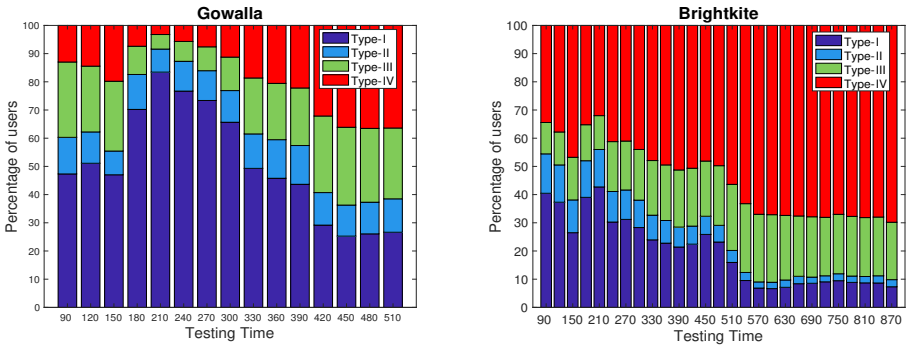


Figure 3.9. Fraction of user types at different times in Gowalla and Brightkite

REGULA utilizes different combination of features based on the type of user (refer Table 3.5). Since for Type-I and Type-II users we do not have their prior check-in information, REGULA does not compute distance feature. For Type-I and Type-III users, since we do not have information about their friends, REGULA does not compute friendship feature. However, it is important to note that REGULA utilizes aggregated temporal feature of all locations to provide recommendations to all user types.

Table 3.5. Features used to provide recommendations for different user types

	Temporal	**Distance**	**Friendship**
Type-I users	YES	NO	NO
Type-II users	YES	NO	YES
Type-III users	YES	YES	NO
Type-IV users	YES	YES	YES

I also present the evolution in number of users per type with time for Gowalla and Brighkite LBSN datasets in Figure 3.9. It captures the transitions of users from Type-I to Type-IV with time as we obtain additional check-in and friendship information.

REGULA for Type -I users

For any Type-I user, since we do not have any prior check-in (i.e., $VL_u^N = \emptyset$) and friendship (i.e., $F_u = \emptyset$) information the distance and friendship feature values are zero. Therefore, I only utilize the aggregated temporal feature (Equation 3.2) of all locations to provide recommendations to all Type-I users. The pseudo-code to provide recommendations to Type-I users is depicted in Algorithm 1

The algorithm takes as input the aggregated temporal feature values of all L locations. Top K locations with highest aggregated temporal feature values are recommended to Type-I users (Line 2, Algorithm 1). Since Type-I users have no prior check-in and friendship information, the sames locations are recommended to all Type-I users

Algorithm 1: Recommend Type-I Users

REGULA -I(TS, K)

Input : Set of all temporal feature value $TS := \{(l, ts(l)\}, l \in L$; K is number of locations to recommended

Output: A set of K recommended locations, R

1: $R = \emptyset$
2: Sort all locations in descending order of temporal feature values in TS
3: $R \leftarrow$ Get top K locations in TS
4: return R

REGULA for Type -II users

For any Type-II user, since we do not have any prior check-in information (i.e, $VL_u^N = \emptyset$) the distance feature value is zero. While, we do have information related to their direct friends and utilize it to compute friendship features of all L locations. Recommendations are given by combining the aggregated temporal feature and friendship feature values. The pseudo-code to provide recommendations to Type-II users is depicted in Algorithm 2

I first find out the direct friends of each Type-II user u (Line 3). Later, I iterate over all un-visited locations (l') and compute the friendship feature value assigned to each one of them (Line 5). The sum of aggregated temporal feature value and friendship feature value is the final combined feature or recommendation score ($rs(l')$) assigned to each l' (Line 6). Top K locations with highest scores are recommended to Type-II user.

REGULA for Type -III and Type -IV users

The difference in Type-III and Type-IV users is only in the non-availability of friendship information for Type-III users. Therefore, REGULA utilizes all three features to provide recommendations to Type-IV users and only utilizes aggregated temporal and distance features to provide recommendations to Type-III

Algorithm 2: Recommend Type-II Users

REGULA -II(u, TS, C, G, K)

Input : user $u \in U$; Set of all temporal features $TS := \{(l, ts(l)\}, l \in L$; Set of all check-ins C; $G = (U, E_f)$ Friendship Graph; K is number of locations to recommended

Output: A set of K recommended locations, R

1: $R = \emptyset$
2: Get unique locations visited by user u using C, VL_u
3: Get direct friends of user u using G, F_u
4: **foreach** $l' \in L \setminus VL_u$ **do**
5: Compute Friendship feature for location l', $fs(u, l')$ (using Equation 3.4)
`/* recommendation score of location` l' `*/`
6: $rs(l') := fs(u, l') + TS.ts(l')$
7: $R \leftarrow R \cup \{(l', rs(l'))\}$
8: **end**
9: Sort R in descending order of scores $rs(l')$
10: $R \leftarrow$ Get top K locations in R
11: return R

users. The pseudo-code to provide recommendations to Type-III Type-IV users is depicted in Algorithm 3

Based on *Observation 1*, I first find out the top 50 most frequently visited locations (FVLs) of a user u (Line 2). Later, for each FVL, I obtain a list of all un-visited locations within a bounded box of 5km (Line 8) (Utilizing *Observation 2*). Further, for every un-visited location l_k of u I compute the distance score (Line 11) and friendship feature (Line 13) (For Type-III users friendship feature value is zero). The final combined feature ($rs(l_k)$) assigned to l_k is the sum of four features: 1) aggregated temporal feature of l_k 2) aggregated temporal feature of corresponding FVL 3) distance feature and 4) friendship feature (Lines 10-14)

If an un-visited location l_k is in the bounded box of many FVLs of user u, then the FVL with largest aggregated temporal feature is used to compute the final combined feature (Line 16). Top N locations with highest combined feature values ($rs(l_k)$ are recommended to Type-III and Type-IV users.

3.5 Evaluation of REGULA model

In this section, I describe the procedure to evaluate my REGULA algorithm based on different performance metrics. I compared REGULA with state-of-the-art feature based algorithm Location-Friendship Bookmark-Coloring Algorithm (LFBCA) Wang et al. [2013]. Further I also compare with other traditional filtering based recommendation algorithms like UserCF and LocCF Bao et al. [2013]. REGULA was implemented in MATLAB and tested on a Linux based server.

3.5.1 Evaluation procedure

Similar to the procedures designed to test my Hypothesis described in Section 3.3, the evaluation was conducted in a sequential manner at different times in intervals of 30 days for Gowalla & Brightkite. For Gowalla, the I evaluated (or recommendations were provided) REGULA and other models at days $t = \{90, 120, .., 510\}$; for Brightkite at days $t = \{90, 120, .., 870\}$. The training data considered for an evaluation at time t are all check-ins in the interval of $(0, t)$. All check-ins in the interval of $[t, t + 30)$ were considered as testing data for Gowalla and Brightkite datasets. In each evaluation, recommendations were provided only to active users (who have atleast one new check-in in the testing data). Further depending on the type of active user (refer Table 3.4), I utilize different variants of REGULA algorithm to provide location recommendations.

In literature, the separation between the training and the testing dataset is

Algorithm 3: Recommend Type-III & IV Users

REGULA -III,IV(u, TS, C, G, K)

Input : user $u \in U$; Set of all temporal scores $TS := \{(l, ts(l)\}, l \in L$; Set of all check-ins C; $G = (U, E_f)$ Friendship Graph; K is number of locations to recommended

Output: A set of K recommended locations, R

Set Bounded Box size $D = 5$ (in km)
Get 50 most frequently visited locations of u using C, L_{fvl}
Get direct friends of u using G, F_u
Get last 10 locations visited by u using C, VL_u^{10}
$R = \emptyset$
foreach $l' \in L_{fvl}$ **do**
    if $l' \exists\ TS$ **then**
        Get un-visited locations in bounded box of size D around l', $Box(l', D)$
        foreach $l_k \in Box(l', D)$ **do**
            /* Assign temporal score of l_k */
            $rs(l_k) = TS.ts(l_k)$
            /* Add temporal score of FVL l' */
            $rs(l_k) = rs(l_k) + TS.ts(l')$
            /* Compute and add distance score of l_k using VL_u^{10} (using Equation 3.3) */
            $rs(l_k) = rs(l_k) + ds(u, l_k)$
            /* Compute and add friendship score of l_k using F_u (using Equation 3.4) */
            if $F_u \neq \emptyset$ **then**
                /* Type-IV users only */
                $rs(l_k) = rs(l_k) + fs(u, l_k)$
            end
            if $\{(l_k, rs(l_k))\} \in R$ **then**
                Update R if new $rs(l_k)$ is larger
            else
                $R \leftarrow R \cup \{(l_k, rs(l_k))\}$
            end
        end
    end
end
Sort R in descending order of scores $rs(l_k)$
$R \leftarrow$ Get top K locations in R
return R

not always fully coherent. While such separation is applied strictly on users, e.g. to identify the active users at time t. The separation is not clear for social ties and locations. By neglecting this distinction for social ties, it might happen that a user u, who is not present in the training dataset, but has a tie with an active user, could be considered for recommendation. This is not an issue when only direct friends recommendations are used, like in REGULA , but it can generate an unrealistic situation when more degrees of ties are used. In this case, we can receive a recommendation from friends of u, who does not exist at the recommendation time t. Similarly, if we do not limit the recommended locations to only those present in the training dataset, we are going to recommend locations that do not exist at the recommendation time t. For this reason, I clearly state that:

- Only users with check-ins in the training dataset can be included in friendship graph.
- Any new location l will be recommended to u only if $l \in L_{train}$, where L_{train} is the set of unique locations in training dataset.

I also apply above conditions to other algorithms used in my evaluation.

3.5.2 Performance metrics

I evaluated the performance of REGULA using two widely used metrics i.e., precision at K ($p@K$) and recall at K ($r@K$) and are defined as,

$$\text{p@K} = \frac{1}{N}\sum_{u=1}^{N}\frac{|S_u(K) \cap V_u|}{K}, \text{r@K} = \frac{1}{N}\sum_{u=1}^{N}\frac{|S_u(K) \cap V_u|}{|V_u|}$$

$S_u(k)$ is the set of top K locations recommended to a user u and V_u is the set of locations visited by user in the testing data. $p@K$ metric measures how many locations (out of K recommendations) were visited by users. $r@K$ metric captures how many locations visited by the user were part of the recommendation.

3.5.3 Performance of REGULA

I vary the independent parameters of my REGULA model and measure its impact on the performance metrics. The different independent parameters of REGULA are:

1. FVL — The number of Frequently Visited Locations control the regions from where candidate locations are selected. I perform a grid search over FVL $\in \{10, 20, 30, 40, 50\}$ and measure its impact on the performance metrics.

2. $bbox$ — The Bounding box (bbox) represents a rectangular region of interest. Bounding boxes placed around every FVL capture the set of candidate locations. Therefore, the total number of candidate locations to rank depends on the size of Bounding Box. A larger bounding box will capture more candidate locations than compared to a smaller bounding box. I perform a grid search of bounding box size over $bbox \in \{1km, 2.5km, 5km\}$ and measure its impact on the performance metrics. The bounding box size is the diagonal length of the rectangle.

3. $lastk$ — This parameter controls how many of the locations visited in the recent past contribute to the distance feature value of a candidate location (refer to Section 3.4.1). I perform a grid search over $lastk \in \{10, 20, 30, 40, 50\}$ and measure its impact on the performance metrics.

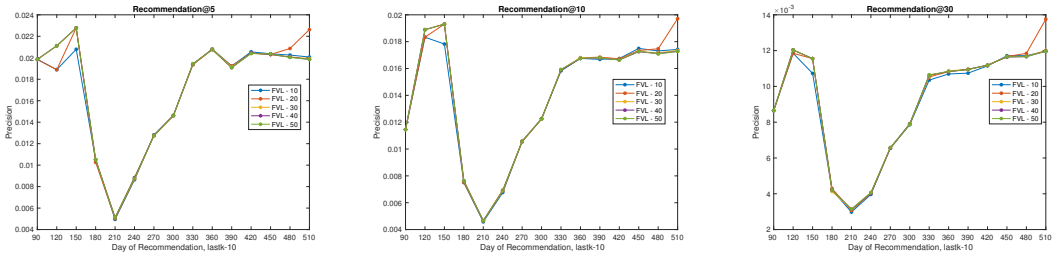
4. alpha (α) - It controls the impact of friendship ties between users on the ranking of candidate locations. I perform a grid search over $\alpha \in \{10, 50, 100, 200\}$ and measure its impact on the performance metrics.

For each experiment, I vary any of the above four independent parameters, together with K - the number of locations recommended to a user u - and measure its impact on the performance metrics. Results of experiments performed for Gowalla and Brightkite datasets are shown in Figures 3.10 to 3.29 respectively.

In the following sections I demonstrate that REGULA is able to utilize information about the frequently visited locations of a user to provide better recommendations compared to the baseline algorithms. I also show that considering new locations within a small distance from the FVLs lead to better recommendations.

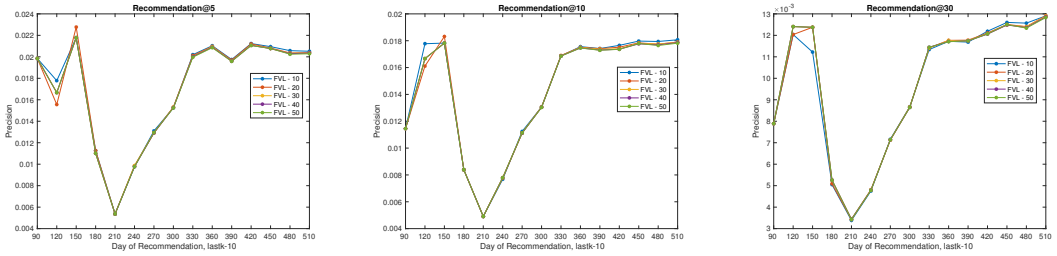
Impact of Frequently Visited Locations on Precision and Recall

Figures 3.10 to 3.21, show how the performance of REGULA varies for different FVLs and a fixed bounding box of $1km/2.5km/5km$ respectively. They also show the difference in performance when FVL is varied from 10 to 50. I only show the performance when $lastk$ is 10 because the performance doesn't not vary significantly when $lastk$ is increased beyond 10. For brevity, I only show the performance when K is 5 or 10 or 30 (for all intermediate values of K refer to Figures in Appendix 8) We observe that as the bounding box size is increased



(a) (b) (c)

Figure 3.10. Performance ($p@K$) of REGULA on Gowalla for different FVLs, α=100, $lastk$=10, Box=1km.



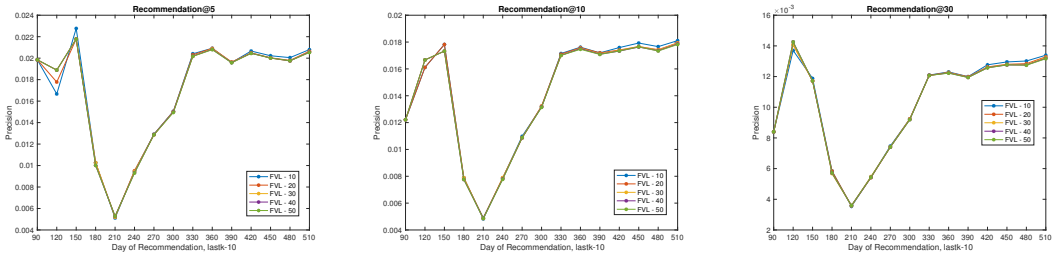
(a) (b) (c)

Figure 3.11. Performance ($p@K$) of REGULA on Gowalla for different FVLs, α=100, $lastk$=10, Box=2.5km.

the performance improves because REGULA is able to obtain more unvisited locations within the bounding box. Further, when the bounding box is set to 1km, we observe that the best performance is obtained when 50 FVLs are used to recommend locations. When bounding box size is increased from 1km to 2.5km or 5km there is no difference in performance for different FVLs as almost all unvisited locations are captured within 2.5km bounding box. Finally, when I increase the number of location recommended K, the total number of locations that need to be ranked correctly also increases. Therefore, as expected, we observe that with increasing K the performance decreases, independently from the other parameters.

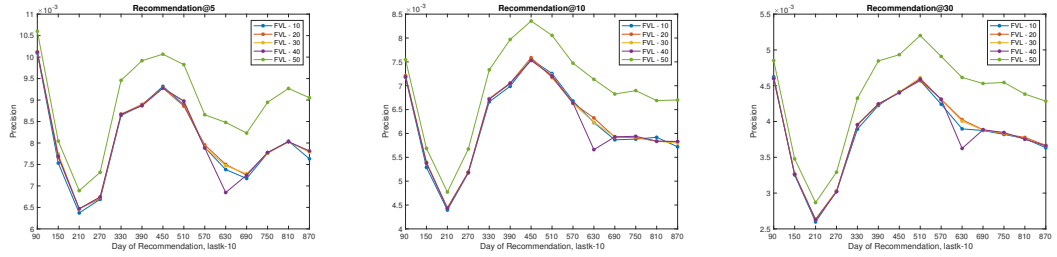
Impact of bounding box size on Precision and Recall

I highlight here the impact of distance that is reflected by the bounding box parameter ($bbox$) already seen in the previous subsection by fixing $\alpha = 100$ and



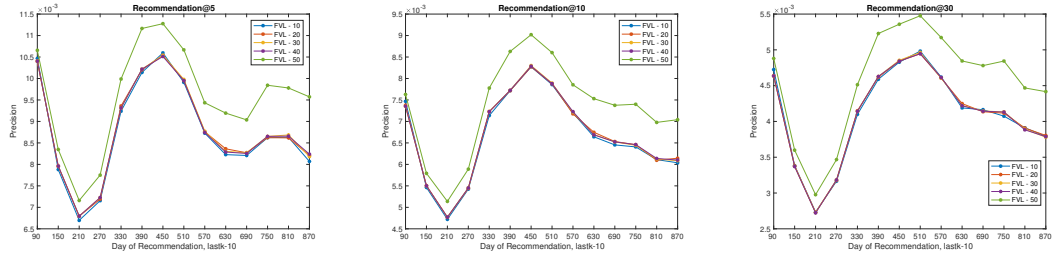
(a) (b) (c)

Figure 3.12. Performance ($p@K$) of REGULA on Gowalla for different FVLs, α=100, $lastk$=10, Box=5km.



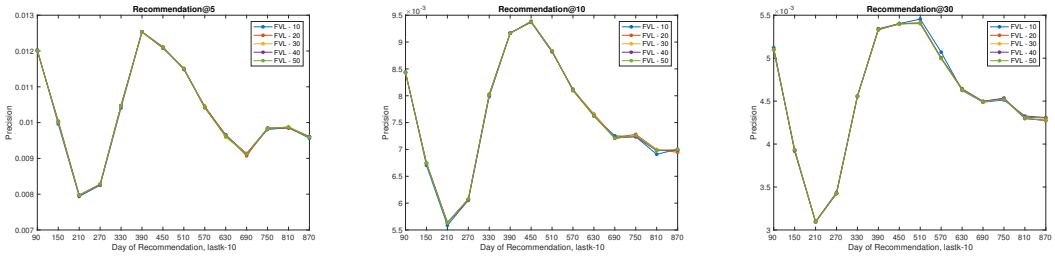
(a) (b) (c)

Figure 3.13. Performance ($p@K$) of REGULA on Brightkite for different FVLs, α=100, $lastk$=10, Box=1km.



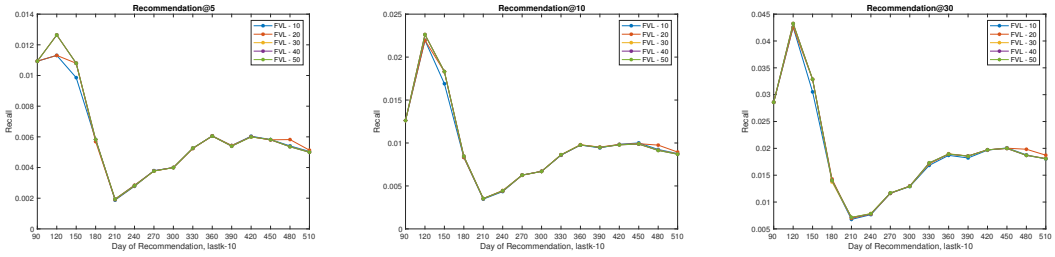
(a) (b) (c)

Figure 3.14. Performance ($p@K$) of REGULA on Brightkite for different FVLs, α=100, $lastk$=10, Box=2.5km.



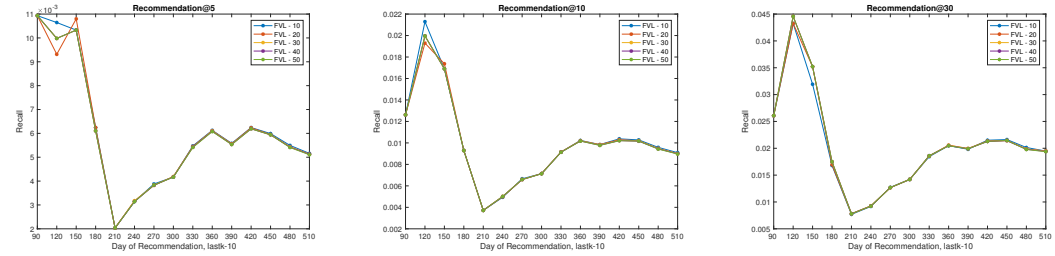
(a) (b) (c)

Figure 3.15. Performance ($p@K$) of REGULA on Brightkite for different FVLs, α=100, $lastk$=10, Box=5km.



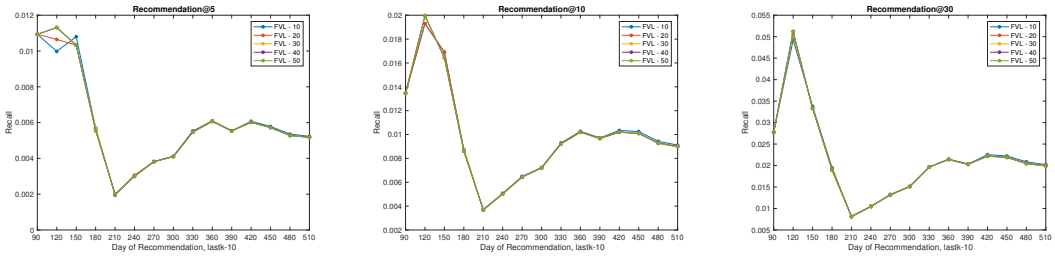
(a) (b) (c)

Figure 3.16. Performance ($r@K$) of REGULA on Gowalla for different FVLs, α=100, $lastk$=10, Box=1km.



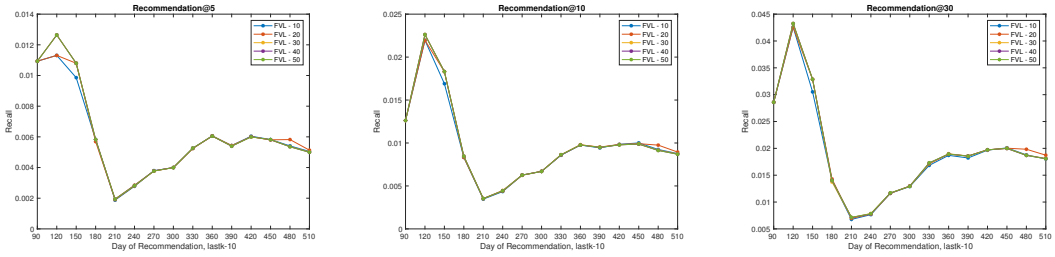
(a) (b) (c)

Figure 3.17. Performance ($r@K$) of REGULA on Gowalla for different FVLs, α=100, $lastk$=10, Box=2.5km.



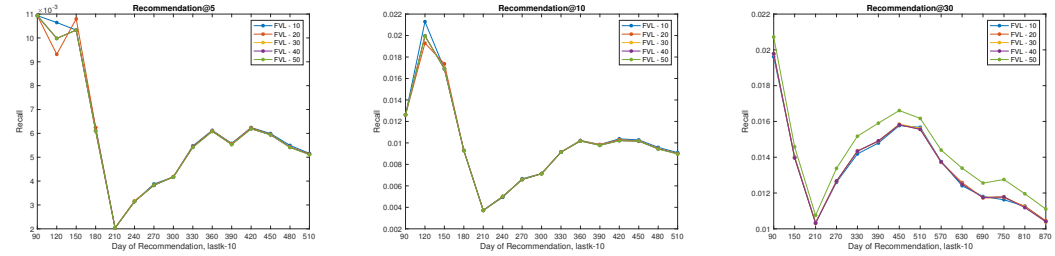
(a) (b) (c)

Figure 3.18. Performance ($r@K$) of REGULA on Gowalla for different FVLs, α=100, $lastk$=10, Box=5km.



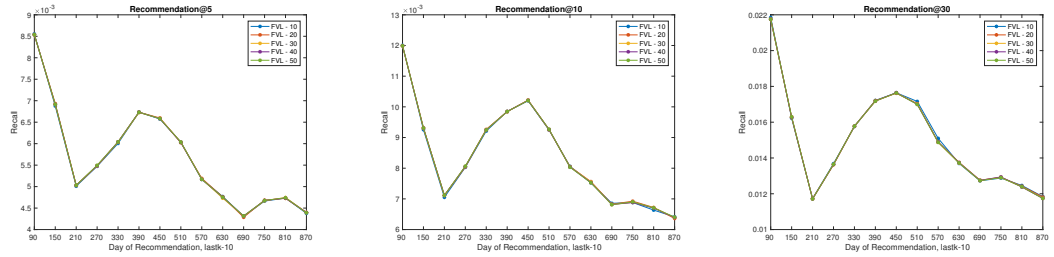
(a) (b) (c)

Figure 3.19. Performance ($r@K$) of REGULA on Brightkite for different FVLs, α=100, $lastk$=10, Box=1km.



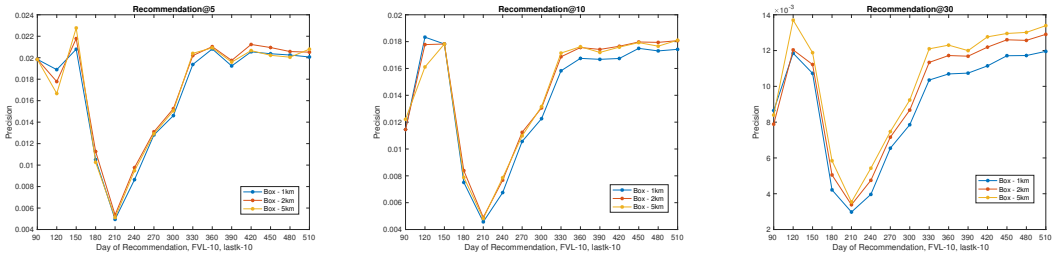
(a) (b) (c)

Figure 3.20. Performance ($r@K$) of REGULA on Brightkite for different FVLs, α=100, $lastk$=10, Box=2.5km.



(a) (b) (c)

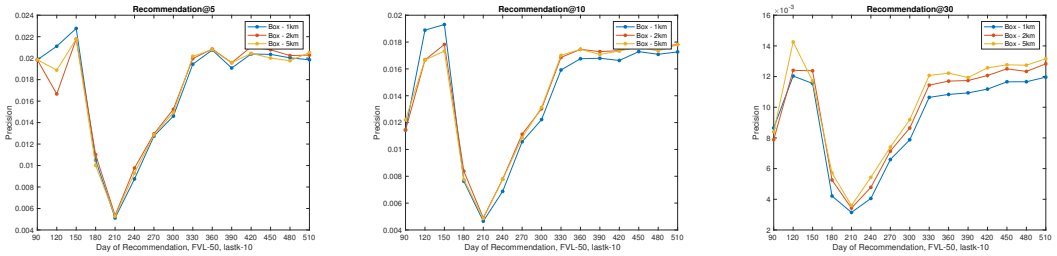
Figure 3.21. Performance ($r@K$) of REGULA on Brightkite for different FVLs, α=100, $lastk$=10, Box=5km.



(a) (b) (c)

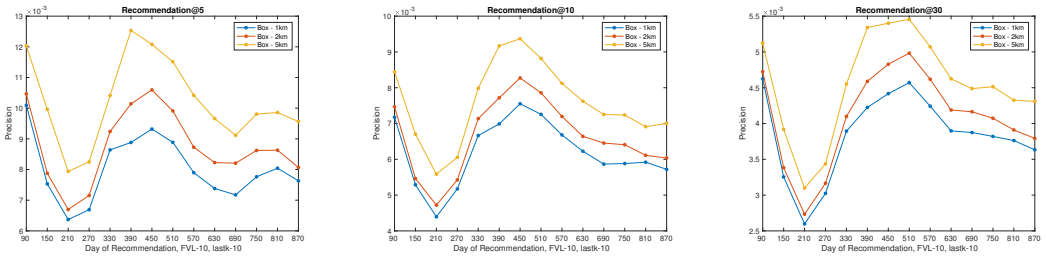
Figure 3.22. Performance ($p@K$) of REGULA on Gowalla for different size of Bounding Box, $\alpha = 100$, $FVL = 10$, $lastk$=10.

$lastk = 10$, which are the values for which I observed best results. For brevity, I only show the performance when FVL is 10 or 50 and K is 5 or 10 or 30 (for all intermediate values of FVL and K refer to Figures in Appendix 8). Figures 3.22 to 3.29 shows how the Precision of REGULA varies with the size of the bounding box. As the size of the bounding box ($bbox$) is increased from $1km$ to $5km$ the total number of candidate locations within the bounding box increases. It is also evident from my analysis (refer to Section 3.3.2) related to Vicinity hypothesis (H2) that shows that as the size of the bounding box is increased the number of users with a new location around their FVLs increases. An increase in the size of the bounding box also increases the percentage of un-related locations in the candidate set that are very far from the places regularly visited by the user. Thus, we observe best performance for the biggest bounding box i.e., 5km.



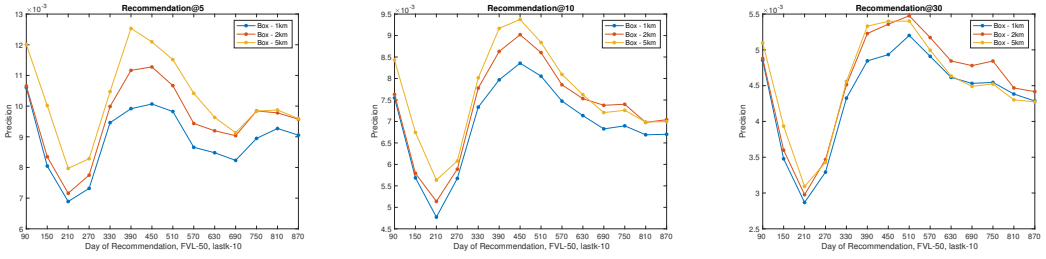
(a) (b) (c)

Figure 3.23. Performance ($p@K$) of REGULA on Gowalla for different size of Bounding Box, $\alpha = 100$, $FVL = 50$, $lastk$=10.



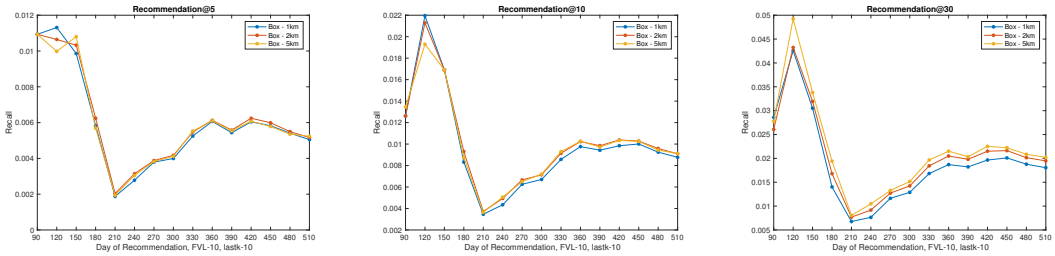
(a) (b) (c)

Figure 3.24. Performance ($p@K$) of REGULA on Brightkite for different size of Bounding Box, $\alpha = 100$, $FVL = 10$, $lastk$=10.



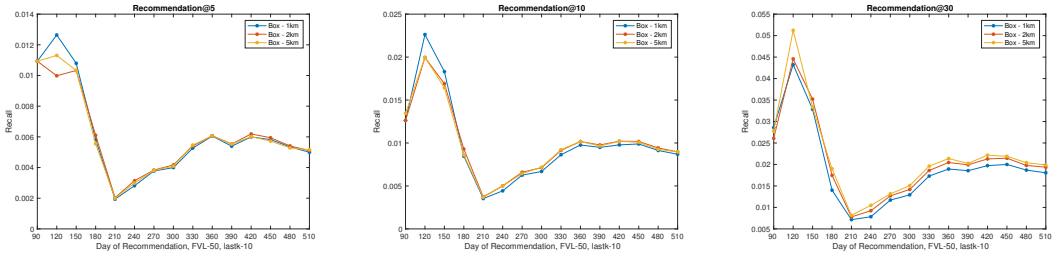
(a) (b) (c)

Figure 3.25. Performance ($p@K$) of REGULA on Brightkite for different size of Bounding Box, $\alpha = 100$, $FVL = 50$, $lastk$=10.



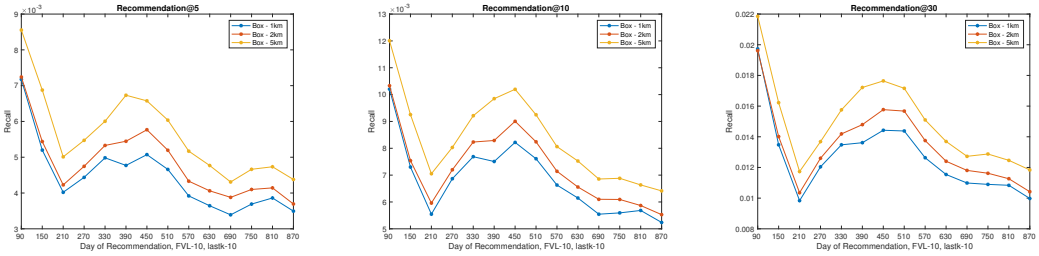
(a) (b) (c)

Figure 3.26. Performance ($r@K$) of REGULA on Gowalla for different size of Bounding Box, $\alpha = 100$, $FVL = 10$, $lastk$=10.



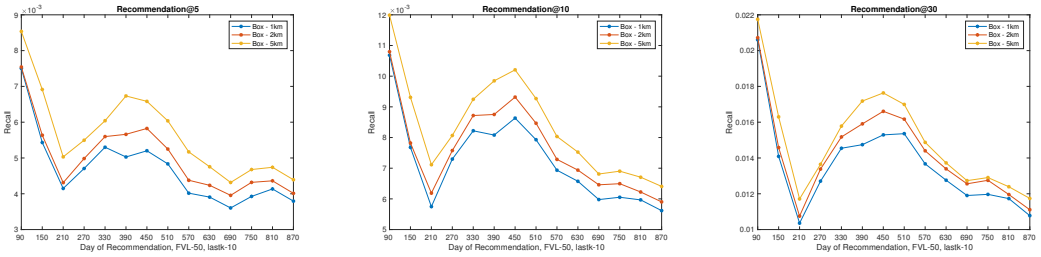
(a) (b) (c)

Figure 3.27. Performance ($r@K$) of REGULA on Gowalla for different size of Bounding Box, $\alpha = 100$, $FVL = 50$, $lastk$=10.



(a) (b) (c)

Figure 3.28. Performance ($r@K$) of REGULA on Brightkite for different size of Bounding Box, $\alpha = 100$, $FVL = 10$, $lastk$=10.



(a) (b) (c)

Figure 3.29. Performance ($r@K$) of REGULA on Brightkite for different size of Bounding Box, $\alpha = 100$, $FVL = 50$, $lastk$=10.

Impact of Last Visited Locations lastK on Precision and Recall

I conducted multiple experiments on Gowalla and Brightkite to study the impact of recently visited locations $lastk$ on the performance on REGULA . I perform a grid search of $lastk$ over $lastk \in \{10, 20, 30, 40, 50\}$. Based on my tests I observer that the impact of $lastk$ locations on the distance score does not significantly change when $lastk$ is varied from 10 to 50. Therefore, I fix the total number of recently visited locations i.e., $lastk$ to 10 in our evaluations.

Impact of alpha on Precision and Recall

I conducted multiple experiments on Gowalla and Brightkite to study the impact of friendship ties by varying α (refer to Equation 3.4). I perform a grid search of α over $\alpha \in \{10, 50, 100, 200\}$. Based on my evaluations, I set the constant weighting factor α to 100 so that REGULA assigns a significantly higher weight to locations visited by friends of a user.

3.6 Performance Comparison

3.6.1 Parameters fixed in Comparative evaluation

Based on my tests to measure the performance of REGULA for Gowalla and Brightkite datasets (Section 3.5.3), I conclude that REGULA performs the best when most of the users move withing a range of 5km from their FVLs. Therefore, I set the bounding box size to 5km in all my evaluations of REGULA . I also fix the total number of recently visited locations considered in computing the distance score to 10 i.e., $N = 10$.

Finally, in order to compare with other recommendation models for each user I recommend 10 locations i.e., $K = 10$. Therefore, I only measure p@10 and r@10 for REGULA and all other baseline algorithms considered.

3.6.2 Performance based on Precision

Figure 3.30 presents *Precision* of different algorithms at different testing times in Gowalla and Brightkite datasets. I observe that REGULA outperforms all state of the art algorithms. It happens because REGULA is able to recommend locations to all four types of users (refer Figure 3.9). The two algorithms UserCF and LocCF are designed to provide recommendations only to users with prior check-in information i.e., only Type-III and Type-IV users. Therefore, their overall *Precision* is lesser than REGULA . My recommendation algorithm REGULA also achieves higher *Precision* than LFBCA due to better recommendations for Type-III and Type-IV users as it utilizes the regularity in human behavior. Further REGULA also improves with time due to an increase in the set of un-visited locations within the bounded box of the FVLs of users.

For Gowalla dataset, the *Precision* for all algorithms reduces in the beginning from 90 to 210 days because there is significant drop in the number of users with prior check-in or friendship information i.e., users of Type-I, Type-II and Type-III (refer Figure 3.9) even though the total number of users increases. From 240 to 510 days REGULA was able to provide recommendations with higher *Precision* because the number of Type-III and Type-IV users increases significantly. For Brightkite dataset, I observe a gradual reduction in the *Precision* for all algorithms as the number of users with un-visited locations reduces with time. This happens because after some days users start leaving the Brighkite LBSN.

REGULA was evaluated at different points in time and for more than 100K users visiting more than 1 millions locations. Therefore, I can claim that REGULA significantly outperforms baselines. The difference looks small based on precision because of the significant number of users in the dataset. A 1% difference over 1 Million users leads to better recommendations for 10,000 users. I can therefore conclude that the *Precision* of REGULA is better than other existing algorithms.

3.6.3 Performance based on Recall

Figure 3.31 presents the *Recall* of recommendations in Gowalla and Brightkite dataset for different algorithms at different testing times. Once again I observe that REGULA outperforms against other algorithms. REGULA is able to provide a

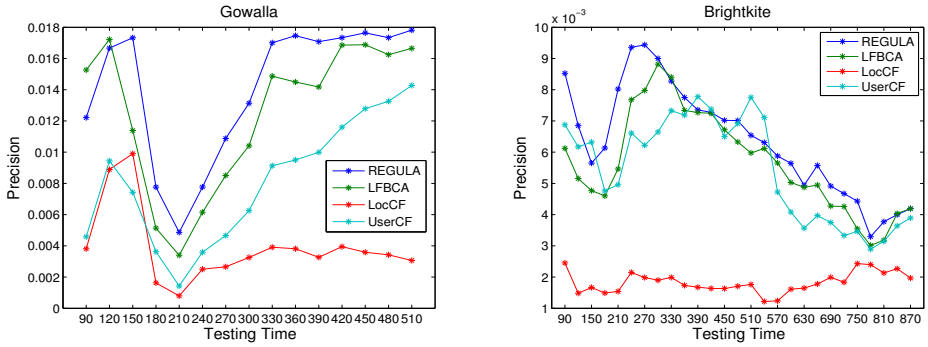


Figure 3.30. Precision ($p@10$) of recommendations provided at different days for Gowalla and Brightkite datasets

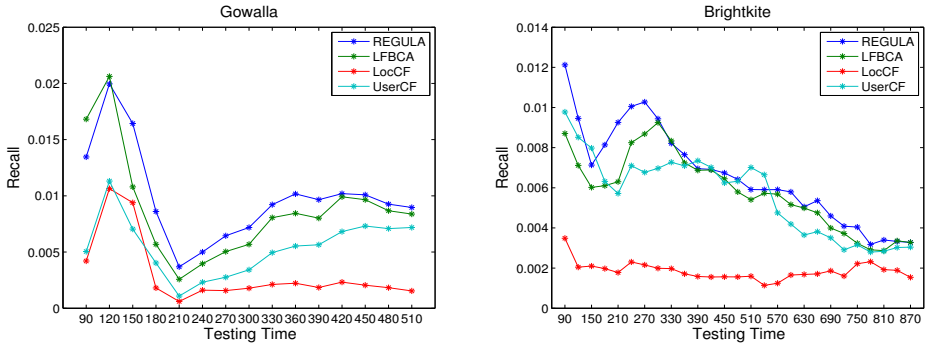


Figure 3.31. Recall ($r@10$) of recommendations provided at different days for Gowalla and Brightkite datasets

larger number of correct recommendations because it utilizes the mobility patterns of a user (i.e., regularity and recency) and is able to select candidate locations from a larger set of un-visited locations around FVLs specific to every user.

3.6.4 Significance of REGULA performance

The standard statistical test to compare the performance of the two algorithms is McNemar's test Dietterich [1998]. The test measures the difference in error rate or the number of users who were not given a correct recommendation (i.e., the user has visited a recommended location). To apply McNemar's test, I categorized the users into four groups: 1) users who were given a correct recommendation by REGULA and LFBCA (the baseline), 2) users who were given a correct recommendation by REGULA but not by LFBCA, 3) users who were given

a correct recommendation by LFBCA but not by REGULA and 4) users who were not given a correct recommendation by LFBCA or REGULA . The null hypothesis is that the two algorithms should have the same error rate. The McNemar's test is based on the χ^2 test for goodness-of-fit that compares the distribution of users under the null hypothesis to the observed count of users. Based on McNemar test I found that for Gowalla dataset REGULA is statistically better ($p<0.05$) from 240 days because there is enough data to capture regularity. Further for Brightkite since the users in the dataset are joining and leaving the network, REGULA is statistically better ($p<0.05$) between 240 to 780 days and is not for the last three tests, i.e on 810, 840, 870 because the number of active users is drastically reduced. Therefore, REGULA statistically performs better than the best baseline, i.e., LFBCA for Gowalla and Brightkite datasets.

Further, concerning the contribution of different features (i.e., temporal score, distance score and friendship score), the temporal score contributes the most in providing relevant locations to users because it captures the global time-varying preferences of users for different locations. The next important feature is the friendship score that captures the impact of time-varying preferences of friends. Since in LBSNs the driver of going to new places is time and friends, therefore, the distance score contributes the least to the overall recommendation performance.

3.7 Conclusion and Limitations

In this chapter, I presented the characteristics related to users and locations of the two standard LBSN datasets: Gowalla and Brightkite. I presented my five hypothesis related to the mobility behavior of these users i.e., *Regularity*, *Vicinity*, *Recency*, *Sociality* and, *Inertia*. I describe in detail the tests I conducted to validate each of these five hypothesis and present observations based on them. Utilizing these observations, I present three features designed to capture the regularity in human mobility. The three features are: 1) Temporal feature that captures the influence of time in user visits, 2) Distance feature that captures the geographical influence of user visits, and 3) Friendship feature that captures the influence of the social friends on user visits.

Later, I presented my recommendation model REGULA that utilizes Temporal, Distance and Friendship features to assign value to each new location of a user. Every user is recommend locations with topK aggregated feature values. I described the overall process of my REGULA model in 5 broad steps, shown in Figure 3.8. I also presented, how REGULA categorizes each user based on the

availability of their prior check-in and friendship information into four different types and utilized different variants of the algorithm to recommend new locations. By using the behavior patterns of human mobility REGULA is able to reduce the set of candidate locations to rank and thereby lowers the complexity to compute a feature value for each candidate location.

I presented the evaluation of REGULA based on two performance metrics: Precision and Recall. I presented the comparative performance of REGULA with three baseline algorithms and show that it outperforms them (at the time it was published) due to the features that capture the regularity in mobility of users. The key findings of this chapter are:

- I present my novel REGULA model that utilizes features based on human mobility to recommend new locations to users of a Location Based Social Network (LBSN). To the best of my knowledge REGULA is the first model that utilizes the Frequently Visited Locations (FVLs) of a user to recommend new location. Its one of the first model to utilize the concept of regions or bounding boxes around FVLs to capture new locations.
- I designed three features based on the analytical tests I conducted to capture the mobility behavior of users.
- REGULA outperforms others recommendation models as it is better able to model the regular mobility patters inherent in the two standard LBSN datasets.
- To the best of my knowledge, REGULA is the first model to capture the temporal visiting patterns of users and their friends with the help of Temporal and Friendship features.

The work presented in this chapter has resulted in a publication at ACM SIGSPATIAL 2015. In the next chapter, I present how we can utilize my REGULA model to recommend regions of interests in the context of a Call Detail Records(CDR) based LBSN.

3.7.1 Limitations

REGULA is a feature based model where I need to compute the distance, temporal and friendship scores for all locations in the dataset before I recommend a set of new locations to users. Further, every time a user requests a recommendation, I need to compute all the scores, this procedure is not scalable in

a real-world system. I overcome these limitations with the help of a deep neural network based recommendation model presented in Chapter5. Finally, the Brightkite LBSN dataset was collected at a time when users were leaving their service. It has led to a significant drop in the number of active users who could be recommended a new location.

Chapter 4

REGULA for CDR-based LBSN

4.1 Introduction

In this chapter, I show how I utilize Call Detail Records (CDR) to construct a new CDR-based LBSN. I show that users of CDR-based LBSN also exhibit regular mobility behavior through the 5 analytical tests. I describe how a recommendation model can be used in the context of CDR. Finally, I compare the performance of REGULA with other models in recommending new regions to users of a CDR-based LBSN

This chapter is structured as follows: In Section 4.2, I will present how I construct a CDR-based LBSN and its characteristics. Section 4.3 presents the results of five analytic tests to show that users of CDR-based LBSN also exhibit regular mobility behavior. Section 4.4 presents the evaluation of REGULA based on two performance metrics and compare it with other recommendation models. Section 4.5 concludes the Chapter.

4.2 CDR-based LBSN Dataset

Quadri et al. [2014] leverage a large anonymized dataset of Call Detail Records (CDRs) Naboulsi et al. [2015] capturing voice calls, short text messages (SMS) and Internet traffic of about 1 million subscribers of an international mobile operator. They operate on a dataset restricted to the metropolitan area of Milan for a period of 67 days, from March 26 to May 31, 2012. This two months of dataset contains a total of 63 millions phone-call records and 20 million SMS records.

Call Detail Records(CDRs) collection and pre-processing

When a user makes a call or sends a text message the following information gets recorded: 1) user ID of sender, 2) user ID of receiver, 3) cell ID of tower sender is connected to, 4) cell ID of tower receiver is connected to, 5) date and time of contact. In Figure 4.1 shows a small sample of the recorded information.

CALL RECORDS
"source","destination","date","time","start_cell","end_cell","dir","duration"
574864,574865,"2012-03-27","13:36:54",47615,47615,"O",0
574864,574867,"2012-03-27","13:55:59",15824,15825,"O",46
574870,574864,"2012-04-02","22:37:41",16677,16677,"I",14

SMS RECORDS
"source","destination","date","time","cell","dir"
1916062,574864,"2012-03-27","21:48:53",16676,"I"
2267867,574864,"2012-03-30","21:59:05",16676,"I"

Figure 4.1. Sample of CDRs reporting call and text message.

SMS and call records are the primary data source to understand the complex structure of the interactions mediated by on-phone communications. Such a complex structure is often represented by a network whose nodes are users and links connect two users who call/text each other. However the choice of drawing a link depends on the purpose of communication. In fact, call or text messages have not all the same social value; this is especially the case of advertisements and commercial messages or communications issued by call centers. Moreover, in the dataset observe a curious behavior: nearly 40% of calls have a duration equal to 0. Besides missed or unanswered calls, such a large amount of rings is echoing a common practice in Italy to use rings for meaning "“Call me back soon"' or "'I'm just arriving"', for instance, to get synchronized at a meeting. Due to the difficulty to discriminate 0-duration calls on the base of their social meaning, in this analysis we decided to remove these records from the dataset, which finally turns out to be composed of 41 million calls and 20 million SMS. Furthermore, according to the literature on mobile phone cleansing Lambiotte et al. [2008]; Blondel et al. [2015]; Karsai et al. [2012]; Li et al. [2015a], we filtered out calls involving other mobile operators, both incoming and outgoing, thus maintaining only activities involving subscribers of the same operator. This way we eliminate the bias between operators.

CDR-based Social Network

On the basis of the obtained CDR dataset, they construct two preliminary on-phone social networks one for each communication channel, that are then processed to extract only those interactions with social relevance. For the in call graph, they consider the pairs of users whose sum of call duration exceeds the minute and whose total number of interactions is higher than 3, while in the SMS graph, the only relevant pairs are those with a total number of interactions higher than 3. After filtering and pre-processing, the processed data contains 7 millions calls, 317,000 hours of conversations and 4 million SMSs generated by about 420,000 people.

CDR-based LBSN

Call Detail Records represent a valuable data source which enables us to capture the social relationships among the operator's customers. In fact, they are considered an essential tool to analyze social dynamics of a large population. Also, CDRs provide the location of the users, enhancing the connection between communications mediated by mobile devices and habits and relationships in the real world. So, mobile phone data are the most valuable form of data to perform user-centric analysis, especially when related to mobility and sociality.

The combination of social networking and geographic information is also typical of location-based social networks (LBSN), so we can report our CDRs dataset into the LBSN modeling framework. The geographic information is represented as a graph where users and cells/locations represent nodes and edges indicate whether a person has been attached to the cell at least once in two months. The social information is also represented as a user-user graph where, edges represent the number of calls and text messages exchanged between users.

The final processed dataset contain 118 million visits to 901 locations (or cell regions). The un-directed friendship network consists of 688,302 nodes with 1,432,938 edges. Table 4.1 presents the characteristics of our CDR-based LBSN dataset.

4.3 Experiments to understand mobility of users in CDR-based LBSN

I conduct the same analytic tests described in Section 3.3 on users of our CDR-based LBSN to check if the five hypothesis presented for standard LBSN datasets

Table 4.1. Dataset Characteristics of CDR-based LBSN

	CDR-based LBSN
Time period of Check-ins	68 Days
Number of Users	688,302
Number of Locations	901
Number of Check-ins	118,752,518
Number of Friendship links	1,432,938

are also applicable these users. The five hypothesis described in Section 3.3 are as follows:

H1: Regularity- Users regularly (or habitually) visit a set of locations i.e., their Frequently Visited Locations (FVLs).

H2: Vicinity- Users visit places in the vicinity of their FVLs.

H3: Recency- Users are more likely to go places that were visited recently by others.

H4: Sociality- Users are more likely to go places that were visited by their friends.

H5: Inertia- Users are more likely to go places geographically close to their present location.

4.3.1 Hypothesis 1: Regularity

Similar to tests on standard LBSN datasets, the test to validate my hypothesis was conducted in a sequential manner. For example, a test conducted on 30th day will take into account all check-ins during the first 30 days. For each user with check-in data in the first 30 days, I find out if they have any FVLs. The goal of this test is to measure what fraction of users have one or more FVLs.

Since CDR-based LBSN data was collected in 68 Days, the tests were conduced in an interval of 15 days i.e., at days $t = \{30, 45\}$. Figure 4.2 shows the fraction of users with atleast one FVL at different testing times in CDR-based LBSN dataset. We observe that more than 80% of users regularly visit atleast one location.

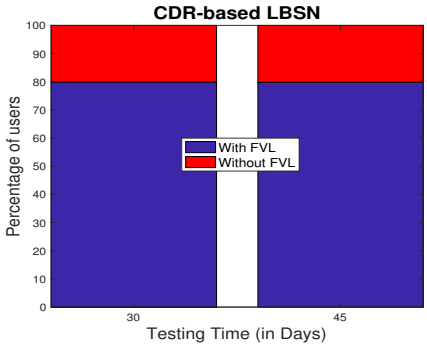


Figure 4.2. Fraction of users with atleast one FVL at different times in Gowalla, Brightkite and CDR-based LBSN datasets

4.3.2 Hypothesis 2: Vicinity

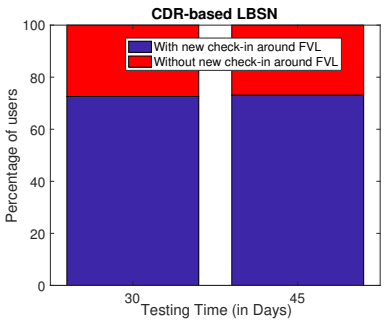
Similar to the analysis done in Section 4.3.1, I test this hypothesis in a sequential manner at different times (t). For each test, I measure the fraction of users that have a new check-in within a distance of 1 km / 2.5km / 5 km from their FVLs. Figure 4.3 presents the fraction of users who have visited atleast one new location within a range of 1 km / 2.5km / 5 km of their FVLs in CDR-based LBSN dataset. Once again, we observe that more than 80% of users visit locations close to their FVLs. This test confirms my hypothesis is also applicable to users of a CDR-based LBSN.

4.3.3 Hypothesis 3: Recency

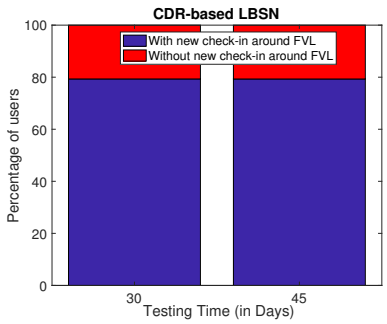
I conduct similar analysis described in Section 3.3.3 to test Recency Hypothesis on users of our CDR-based LBSN dataset. From my tests, I found that all users go to places that were visited by others in the previous day. Since, the number of users in our CDR-based LBSN dataset is ∼3.5 times greater than users in Gowalla dataset the probability that somebody had visited a given location on the previous day is very high.

4.3.4 Hypothesis 4: Sociality

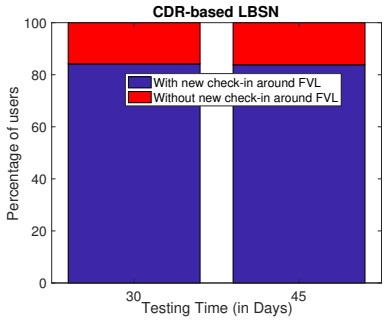
Like in tests described in Section 3.3.4, I conducted tests to check whether a user is affected by their social network friends. For each user, I find out the total number of unique new location visits and then measure what percentage of them were recently visited by their social friends. Figure 4.4 presents how friends influence new places visited by a user in our CDR-based LBSN dataset. We



(a) within a distance of 1km.



(b) within a distance of 2.5km.



(c) within a distance of 5km.

Figure 4.3. Fraction of users with new check-ins within a certain distance from their FVLs at different times (Days) in CDR-based LBSN dataset.

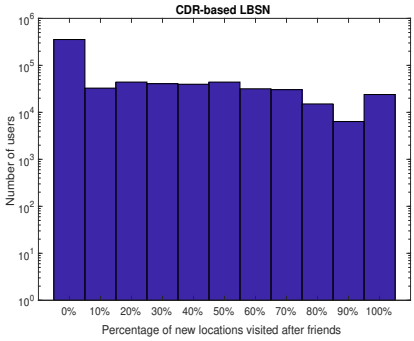


Figure 4.4. Distribution of users who have visited a new location after their social friends.

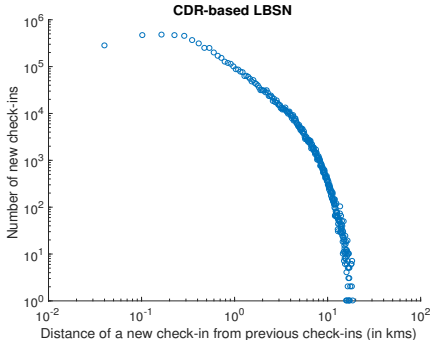


Figure 4.5. Distribution of how far do people travel between two consecutive locations.

observe that a significantly large number of users have at least one new location that was previously visited by their social friends.

4.3.5 Hypothesis 5: Inertia

My final hypothesis is that people exhibit a lot of inertia and often tend to go nearby places. I conducted tests similar to the one described in Section 3.3.5 where for each new location visited by a user, I measure its geographical distance from locations visited before to see whether they exhibit inertia or not. Figure 4.5 presents a distribution of distance traveled for a new location visit by all users in our CDR-based LBSN dataset. We observe that almost 90% of new locations visited by users are within a range of less than 1km. It confirms my hypothesis that people prefer going to places geographically close to their present location.

4.4 Recommendation in CDR-based LBSN

In this section, I describe the procedure to evaluate recommendation algorithms on CDR-based LBSN using the Precision performance metric described in Section 3.5.2. Since, I apply REGULA to a new kind of LBSN i.e., CDR-based LBSN I conduct extensive evaluation by varying all the parameters of REGULA and measure its impact on the recommendation performance.

In traditional item recommendation datasets like Movies (Gomez-Uribe and Hunt [2016]), Factorization based models are found to perform better than collaborative filtering algorithms Bobadilla et al. [2013]. These models have become popular after outperforming filtering based methods in the NetFlix competition Koren et al. [2009]. Therefore, in this chapter I compare the performance of REGULA with LibFM Rendle [2012], a standard factorization based models and GeoMF Lian et al. [2014] a state-of-the-art factorization model (refer in Section 2.4.1) that incorporates spatial constraints because people's mobility patterns tend to cluster around specific locations .

Similar to the test procedure described in Section 4.3.1, the evaluation was conducted in a sequential manner at different times in intervals 15 days. REGULA and other algorithms were evaluated (or recommendations were provided) at days $t = \{30, 45\}$. The training data considered for an evaluation at time t are all check-ins in the interval of $(0, t)$. All check-ins in the interval of $[t, t+15)$ were considered as testing data. In each evaluation, recommendations were provided only to active users (who have atleast one new check-in in the testing data).

4.4.1 Performance of REGULA

I vary the independent parameters of my REGULA model and measure its impact on the performance metrics. The different independent parameters of REGULA are:

1. FVL — The number of Frequently Visited Locations control the regions from where candidate locations are selected. I perform a grid search over FVL $\in \{10, 20, 30, 40, 50\}$ and measure its impact on the performance metrics.

2. $bbox$ — The Bounding box (bbox) represents a rectangular region of interest. Bounding boxes placed around every FVL capture the set of candidate locations. Therefore, the total number of candidate locations to rank depends on the size of Bounding Box. A larger bounding box will capture

more candidate locations than compared to a smaller bounding box. I perform a grid search of bounding box size over $bbox \in \{1km, 2.5km, 5km\}$ and measure its impact on the performance metrics. The bounding box size is the diagonal length of the rectangle.

3. $lastk$ — This parameter controls how many of the locations visited in the recent past contribute to the distance feature value of a candidate location (refer to Section 3.4.1). I perform a grid search over $lastk \in \{10, 20, 30, 40, 50\}$ and measure its impact on the performance metrics.

4. alpha (α) - It controls the impact of friendship ties between users on the ranking of candidate locations. I perform a grid search over $\alpha \in \{10, 50, 100, 200\}$ and measure its impact on the performance metrics.

For each experiment, I vary any of the above four independent parameters, together with K - the number of locations recommended to a user u - and measure its impact on the performance metrics. Results of experiments performed on the 30th day and 45th day are shown in Figures 4.6, 4.8, 4.10 and Figures 4.7, 4.9, 4.11 respectively.

In the following sections I demonstrate that REGULA is able to utilize information about the frequently visited locations of a user to provide better recommendations compared to the baseline algorithms. I also show that considering new locations within a small distance from the FVLs lead to better recommendations.

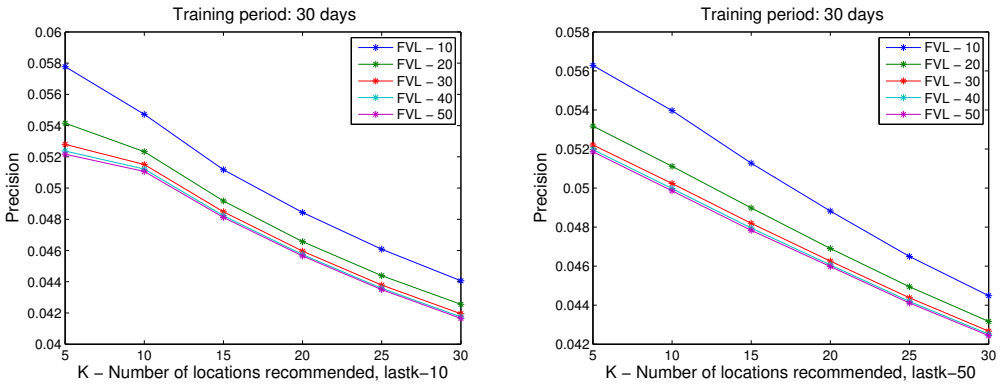
4.4.2 Impact of Frequently Visited Locations on Precision

Figures 4.6 and 4.7, show how the performance of REGULA varies for different FVLs and a fixed bounding box of $1km$. They also show the difference in performance when $lastk$ is varied from 10 to 50. For brevity, I only show the performance when $lastk$ is 10 or 50 (for all intermediate values of $lastk$ refer to Figures in Appendix 8). We observe that for a fixed K as the number of FVLs is increased from 10 to 50, the set of candidate locations obtained within a fixed bounding box increases along with difficulty to correctly rank them. Adding more FVLs increases the noise to rank candidate locations. The phenomena behind this effect has been presented in Barabasi's work Gonzalez et al. [2008b]: people tend to be inertial. Therefore, they will hardly visit places far away from current location. Thus, places that are close to far away FVLs are unlikely to be visited. Further, when I increase the number of location recommended K, the total number of locations that need to be ranked correctly also increases. There-

fore, as expected, we observe that with increasing K the performance decreases, independently from the other parameters.

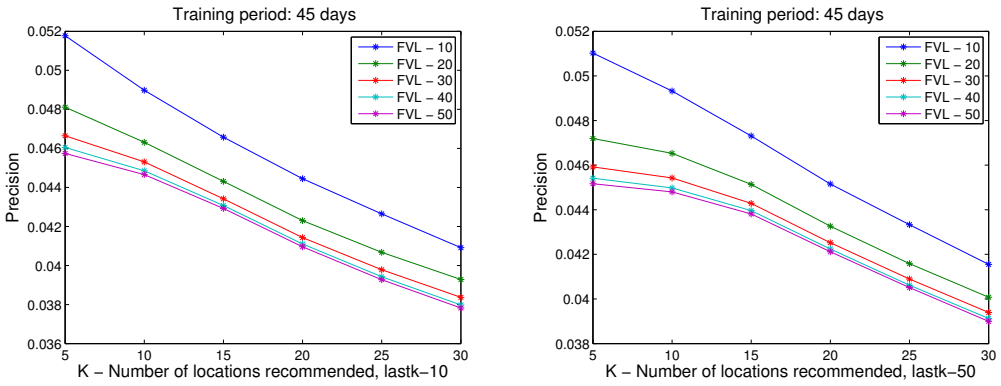
In Figures 4.8, 4.9 I show the performance of REGULA for different FVLs and a fixed bounding box of size $2.5km$. When K is varied from 5 to 30, we observe a significant change in the pattern of performance of REGULA compared to the $1km$ bounding box. For a user with a certain number of FVLs, when the bounding box size is increased from $1km$ to $2.5km$ the number of potential un-visited locations of user increases because a bigger bounding box around FVL will contain more number of candidate locations than compared to a smaller one. Analyzing our CDR-based LBSN dataset, we observe that, on an average, each user in the tests conducted on 30th and 45th day have visited 15 locations, that is: the average value of $|V_u|$ is 15. When K increases from 5 to 10, REGULA 's performance improves because it can correctly rank the top-k locations. However, when k is increased beyond 10, the performance reduces because, as I said above, with an higher value of K the total number of locations that need to be correctly ranked increases, but the $2.5km$ bounding box is unable to capture all potential candidate locations.

In Figures 4.10, 4.11 I show the performance of REGULA for different FVLs and a fixed bounding box of size $5km$. We observe that there is no difference in performance for different FVLs because the $5km$ bounding box captures all the potential candidate locations for all users. Therefore, whether we consider 10 or 50 FVLs of a user the total number of candidate locations to rank is same, as the large $5km$ bounding box already includes all un-visited locations. When K is varied from 5 to 15 the performance of REGULA increases because it can correctly rank the top-K locations. Since each user in our CDR-based LBSN dataset has visited on an average 15 locations the theoretically best performance will be at $K = 15$, where the set of recommended locations and visited locations could be same. Therefore I observe the best performance of REGULA when K is 15. Beyond 15, the performance of REGULA drops because the intersection between the set of recommended locations and visited locations is maximum at 15, while the increase in K or number of recommended locations negatively impacts the performance.



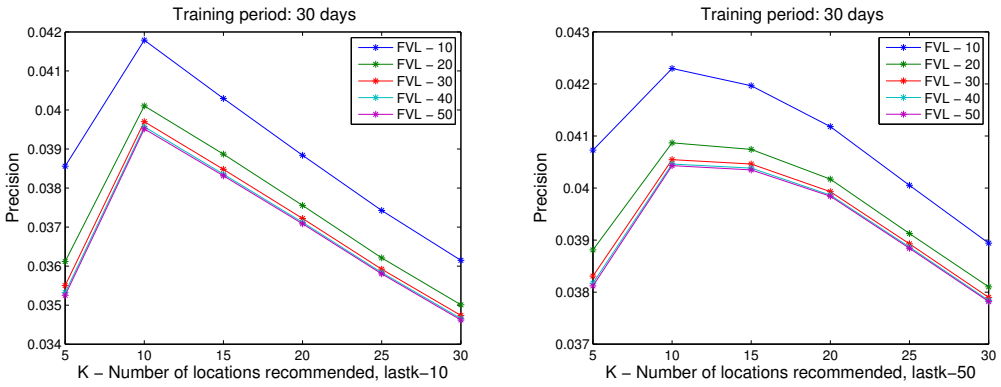
(a) (b)

Figure 4.6. Performance ($p@K$) of REGULA on CDR-based LBSN for different FVLs, α=10, Box=1km, and 30 days for training.



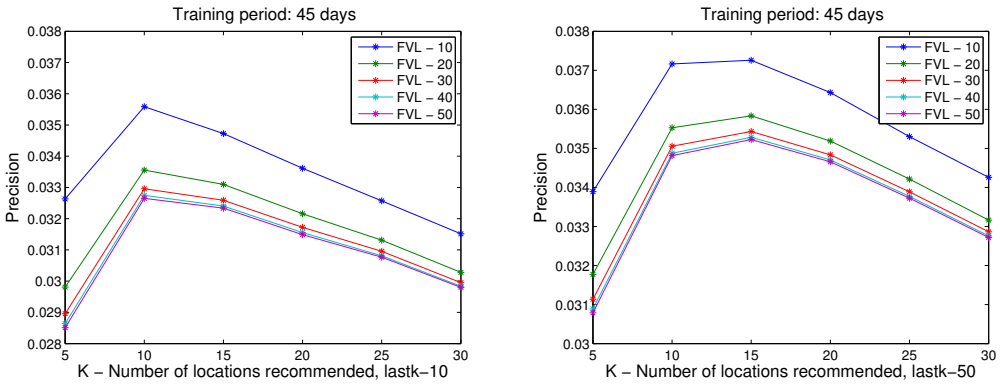
(a) (b)

Figure 4.7. Performance ($p@K$) of REGULA on CDR-based LBSN for different FVLs, α=10, Box=1km, and 45 days for training.



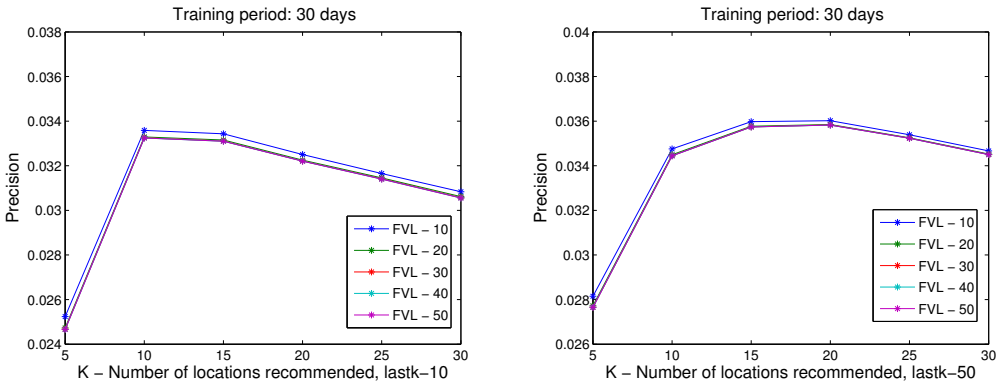
(a) (b)

Figure 4.8. Performance ($p@K$) of REGULA on CDR-based LBSN for different FVLs, α=10, Box=2.5km, and 30 days for training.



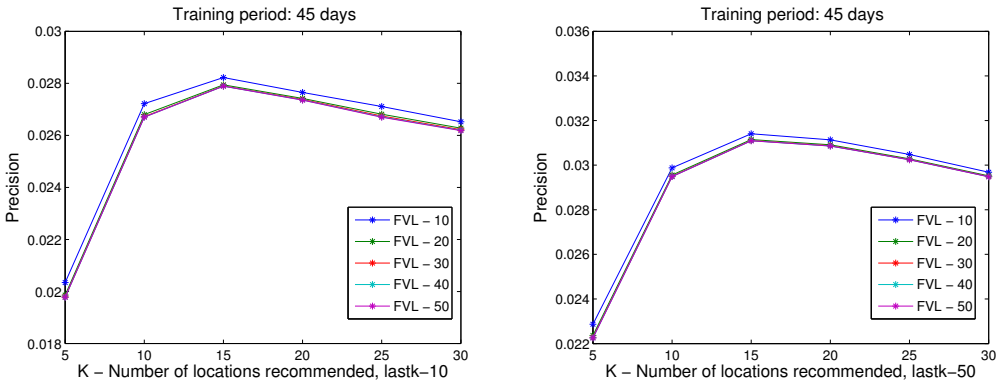
(a) (b)

Figure 4.9. Performance ($p@K$) of REGULA on CDR-based LBSN for different FVLs, α=10, Box=2.5km, and 45 days for training.



(a) (b)

Figure 4.10. Performance ($p@K$) of REGULA on CDR-based LBSN for different FVLs, α=10, Box=5km, and 30 days for training.

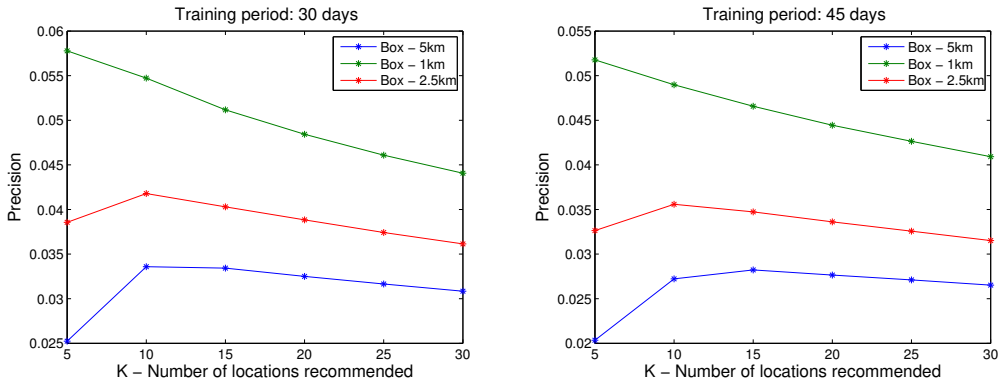


(a) (b)

Figure 4.11. Performance ($p@K$) of REGULA on CDR-based LBSN for different FVLs, α=10, Box=5km, and 45 days for training.

4.4.3 Impact of bounding box size on Precision

I highlight here the impact of distance that is reflected by the bounding box parameter ($bbox$) already seen in the previous subsection by fixing $FVL = 10$, $\alpha = 10$ and $lastk = 10$, which are the values for which I observed best results. Figure 4.12 shows how the performance of REGULA varies with the size of the bounding box for experiments performed on 30th and 45th day. As the size of the bounding box ($bbox$) is increased from $1km$ to $5km$ the total number of candidate locations within the bounding box increases. It is also evident from my analysis (refer to Section 3.3.2) related to Vicinity hypothesis (H2) that shows that as the size of the bounding box is increased the number of users with a new location around their FVLs increases. An increase in the size of the bounding box also increases the percentage of un-related locations in the candidate set that are very far from the places regularly visited by the user. Locations that are far away from the FVLs might be closer to the $lastk$ recently visited locations that lead to a rise in the distance score (Equation 3.3) and an increase in the overall recommendation score (Equation 3.5) assigned to it. An increase in the bounding box increases the noise in the overall ranking of candidate locations. Thus, a smaller bounding box will lead to a reasonable number of candidate locations that can be ranked efficiently.



(a) (b)

Figure 4.12. Performance ($p@K$) of REGULA on CDR-based LBSN for different size of Bounding Box, $\alpha = 10$, $FVL = 10$, $lastk$=10, and 30/45 days for training.

4.4.4 Impact of Last Visited Locations lastK on Precision

From Figures 4.6-4.11, we observe that REGULA performs best when we consider a small number of recently visited locations $lastk$. Increase in the parameter $lastk$ adversely impacts the performance because locations visited in the distant past reduces the impact of locations that were visited most recently. Therefore the performance of REGULA reduces as $lastk$ is increased from 10 to 50.

4.4.5 Impact of alpha on Precision

I performed multiple experiments on 30th and 45th day to study the impact of friendship ties by varying α. I perform a grid search of α over $\alpha \in \{10, 50, 100, 200\}$ and found that the performance of REGULA does not vary meaningfully. Therefore I conclude that the variable α used to compute the impact of friendship score on the recommendation does not have a significant influence on the overall performance.

4.4.6 Performance Comparison

Performance of Baseline Algorithm: LibFM

Figure 4.13 shows the performance of LibFM for different number of recommended locations K. Based on my experiments we observe that LibFM assigns very similar scores to all the candidate locations of a user. Since there are only 900 locations in our CDR-based LBSN dataset, LibFM is unable to utilize the limited number of interactions between user and locations in training data, to correctly score the candidate locations in testing data. As K is varied from 5 to 30 the probability for a visited location to be part of the top k increases that leads to an increase in precision. However, the performance of LibFM is still very low than compared to REGULA . Based on my additional experiments we observe that $p@K$ of LibFM reduces beyond $K = 50$.

Performance of Baseline Algorithm: GeoMF

In Figure 4.14, I show the performance of GeoMF for different number of recommended locations K. Similar to standard recommendation algorithms that rank all un-visited locations in the training data of a user, as K increases the precision reduces because of the increased in difficulty to correctly rank the top K locations. As K is increased from 5 to 30, we do not observe a proportional increase in the number of visited locations that are part of the top K recommendations.

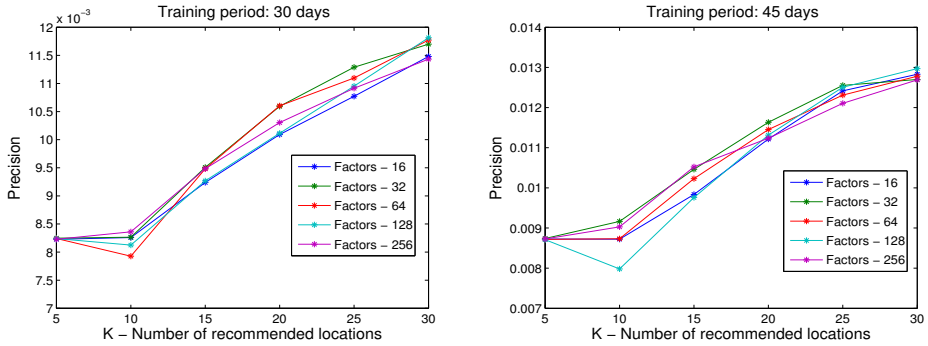


Figure 4.13. Performance ($p@K$) of LibFM on CDR-based LBSN.

We find that GeoMF performs best when every user and location are modeled by a latent factor of size 32.

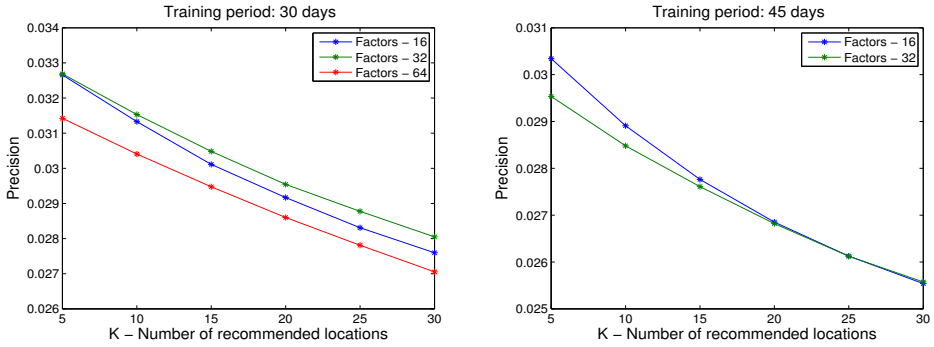


Figure 4.14. Performance ($p@K$) of GeoMF on CDR-based LBSN.

Comparison of REGULA with Baseline Algorithms

In Figures 4.15 4.16, I compare the performance of REGULA , LibFM, and GeoMF for experiments performed on 30th and 45th day. For true comparison, I select the best parameters for each algorithm i.e., for REGULA $\{bbox = 1km, FVL = 10$ and $lastK = 10\}$; for GeoMF $\{Factors = 32\}$; for LibFM $\{Factors = 128\}$. When we recommend 5 locations to each user, i.e., $K = 5$, REGULA performs six times better than libFM and two times better than GeoMF. When K is increased from 5 to 30, REGULA still outperforms GeoMF, but we observe a reduction in the performance gap because $1km$ bounding box of REGULA does not capture all potential candidate locations.

Further, concerning the contribution of different features (i.e., temporal score, distance score and friendship score), once again the temporal score contributes

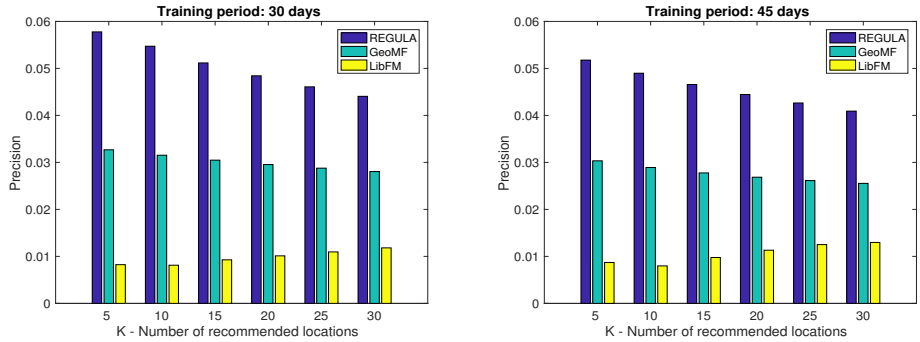


Figure 4.15. Comparison between the performance ($p@K$) of REGULA, LibFM and GeoMF for their best parameters i.e., for REGULA $bbox$ = 1km, $FVL = 10$ and $lastk = 10$; for GeoMF $Factors = 32$; for LibFM $Factors = 128$

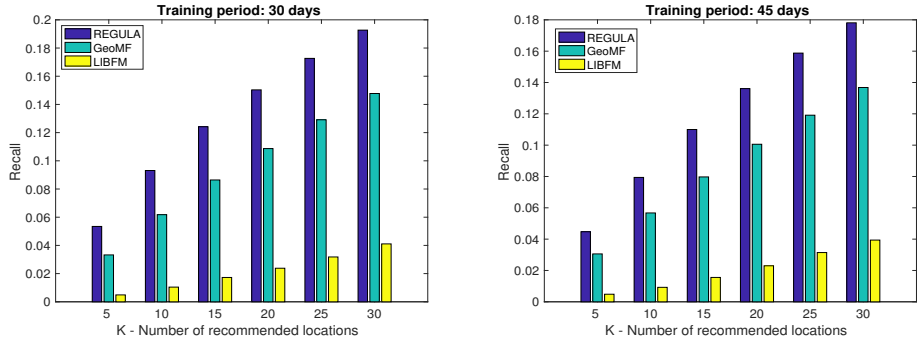


Figure 4.16. Comparison between the performance ($r@K$) of REGULA, LibFM and GeoMF for their best parameters i.e., for REGULA $bbox$ = 1km, $FVL = 10$ and $lastk = 10$; for GeoMF $Factors = 32$; for LibFM $Factors = 128$

the most in providing relevant locations to users in CDR-based LBSN because it captures the global time-varying preferences of users for different locations. Different from standard LBSNs the friendship score contributes the least because the social ties captured by CDR is not that strong compared to social ties in LBSN. The second most important feature in CDR-based LBSN is the distance score because people visit places close to their homes/office and the CDR-based LBSN is constructed based on users in the Milan metropolitan area.

4.5 Conclusion and Limitations

In this chapter, I introduced a CDR-based LBSN and how we can construct it using CDR data. I presented the analytical tests I conducted on users of our CDR-

based LBSN to prove that they also exhibit the five behavior patterns: *Regularity*, *Vicinity* and *Recency*, *Sociality* and, *Inertia*.

Later, I presented a comprehensive evaluation of REGULA based on Precision performance metric. I measured the impact of the four independent variables of REGULA i.e., 1) Frequently Visited Location (FVL), 2) bounding box size, 3) $lastk$ and 4) alpha, on precision. I presented the comparative performance of REGULA with two state-of-the-art factorization based algorithms and show that it outperforms them due to the features that capture the regularity in mobility of users. The key findings of this chapter are:

- I present a new direction of research, where CDR data can be used by mobile operators to recommend regions of interests to their customers. The results of a recommendation algorithm can also be used by mobile operators to proactively allocate network resources to handle mobile connections.
- To the best of my knowledge, REGULA is the first model to recommend regions of interests to users based on their Call Detail Records (CDR).
- I present the unique characteristics of a CDR-based LBSN compared to standard LBSNs. Since people always carry their mobile phones with them, a CDR-based LBSN is closest to capture the real mobility of people and my analysis shows that they exhibit a very high degree of regular mobility.
- REGULA performs 6 times better than standard LIBFM recommendation models as it better able to model the regular mobility patters of a large number of users at comparatively small set of locations.

The work presented in this chapter has resulted in a publication at COMPLENET 2018 and one journal paper is under submission to Pervasive and Mobile Computing. The model presented in this chapter only takes into account the precise temporal aspects like when was the location visited. However, it still does not handle time-aware recommendations i.e., recommended locations do not vary with time. In next chapter, I will present my deep learning based model called DEEPREC which provides time-aware recommendations.

4.5.1 Limitations

REGULA is a feature based model where I need to compute the distance, temporal and friendship scores for all locations in the dataset before I recommend a set of

new locations to users. Further, every time a user requests a recommendation, I need to compute all the scores, this procedure is not scalable in a real-world system. I overcome these limitations with the help of a deep neural network based recommendation model presented in Chapter5. Finally, the CDR-based LBSN dataset was built based on the CDR of users. Therefore, the social ties captured by it is not as strong compared to users who are friends in a social network and visit location together.

4.6 Remarks

The work done in this Chapter is done in collaboration with Dr. Matteo Zignani, Prof. Sabrina Gaito, Prof. Gian Paolo Rossi of Universitá degli Studi di Milano, Milan, Italy and it is under submission to Pervasive and Mobile Computing Journal.

Chapter 5

Deep Neural Network based Recommendation Model

5.1 Introduction

In this chapter, I present a neural network based model that can learn the preferences of users by understanding their complex visiting patterns at different locations. Recent advances in the parallel processing hardware like graphic cards has enabled us to perform large scale computations, that is critical to train a neural network model. This has enabled researchers to design complex neural networks and train them to learn complex patterns in data. For example, neural network models have outperformed state-of-the art models in recognizing complex pictures like handwritten digits, real world objects like flowers, car number plates and human faces. Specifically, in the domain of recommendations, neural network models are able to learn personalized preferences of users from very large data-set of user check-ins.

In this chapter, I first present a Feed Forward Neural Network (FFNN) model that is able to learn time-independent location preferences of people based on their visits to multiple locations. I will show why a FFNN is able to learn location preferences of people better than one of the most successful recommendation model i.e., a matrix factorization based model. I will also present a FFNN model that incorporates geographical constraints of neighborhood locations to ensure that locations that are within a small geographical region share similar geographical preference than compared to locations that are far away. I refer to the combination of two FFNNs that learn time-independent location preferences of users along with spatial constraints as `DEEPREC` model.

This chapter is structured as follows: In Section 5.2, I present the recommen-

dation process and basic assumptions of a standard matrix factorization based model. I also present how a neural network model overcomes the limitations of matrix factorization based model and performs better than it. Section 5.3, presents a FFNN model can be trained to learn the time-independent preferences of users and locations. Section 5.4, presents how I capture the geographical constraints between locations using a different type of feed forward neural network model. Section 5.5, presents my joint neural network model that I refer to as DEEPREC along with the procedure to train it. Section 6.5, presents the evaluation of DEEPREC and its comparative performance with state-of-the-art recommendation models along with REGULA on a large dataset that contains check-ins of users located in the city of Beijing, China. Finally, Section 5.7 concludes the Chapter.

5.2 Location Recommendation Model: Matrix Factorization vs Neural Network

The problem of location recommendation has some unique and different characteristics compared to other online recommender systems. In traditional recommender systems like recommending movies, or products in a website, the goal is provide a list of movies/items based on the profiles and previous behavior of the user. Whether a movie/item is of interest to the user is validated by whether the user has clicked that particular online movie/item. So the feedback and behavior of user towards the movie/item is much faster and the cost of accepting the recommended movie/item by the user is low. In the case of location recommendation, the possibility that a user visits a recommended location depends also on other real world factors like physical distance to the recommended place from the user's present location.

Collaborative filtering based models are one of the most successful recommendation models, especially matrix factorization based models. I will first describe how a basic matrix factorization based model recommend locations to users. What are the underlying assumptions and limitations of factorization based models?. Later I present how a neural network model is able to overcome the limitations of factorization based models to provide better recommendations to users.

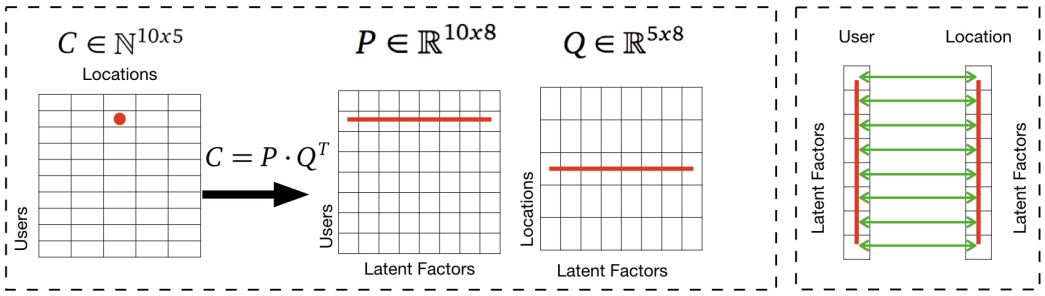


Figure 5.1. Factorization methods aim to decompose a sparse check-in matrix C into user factor matrix P and location factor matrix Q. The preference of a user for a location is given by the dot product of their latent factors i.e., user and location factors interact independently.

5.2.1 Matrix Factorization Model

Matrix Factorization (MF) based models are the most popular recommendation models that have shown to perform very well Koren et al. [2009]; Lian et al. [2017]. MF based models characterize the preferences of each user and location by vector of factors and the interaction between users and locations can be captured by the interaction between their latent factors or feature vectors.

Let a sparse matrix $C \in \mathbb{N}^{MxN}$ capture the mobility interaction between M users and N locations. Every element $c_{i,j} \in C$ represents the visiting frequency of user u_i to location l_j. The matrix C captures the implicit feedback of user's preference for locations. The matrix is sparse because it contains information only about known user visits. MF based models assume that each user and location can be mapped to joint latent factor space and the preference of a user for a location can be approximated by their dot product in the latent factor space. If every user and location is mapped to a latent factor space of dimension d then, the goal of any MF based model is to decompose the sparse matrix $C \in \mathbb{N}^{MxN}$ into two matrices $P \in \mathbb{R}^{Mxd}$ and $Q \in \mathbb{R}^{Nxd}$ that represent the latent factors of M users and N locations respectively. Figure 5.1, shows the general procedure to decompose matrix C into two matrices P and Q. The latent factor of user u_i can be represented as $p_i \in \mathbb{R}^{1xd}$ which is the i^{th} row of matrix P. The latent factor of location l_j can be represented as $q_j \in \mathbb{R}^{1xd}$ which is the j^{th} row of matrix Q. The elements of vector q_j measure the extent to which location l_j posses the d factors. While the elements of vector p_i measure the interest of user u_i for such d factors or features. The predicted preference of user u_i for location l_j is given by the following equation:

$$\hat{c}_{i,j} = p_i.q_j^T \tag{5.1}$$

The goal of MF based methods is to find the set of user and location latent factors that minimizes the squared error in the predicted preference of known user check-ins.

$$\min_{P,Q} \sum_{(u_i,l_j)\in C} (c_{i,j} - \hat{c}_{i,j})^2 \tag{5.2}$$

Optimization algorithms like Stochastic Gradient descent (SGD) Bottou [2012]; Ruder [2016] are used to minimize Equation 5.2 for the best set of user and location factors. Once we obtain the latent factors of all users and locations then for each user we compute the predicted preference for all its un-visited or new locations using Equation 6.18. Top-k new locations with highest predicted score are recommended to the user.

The goal of different MF based models is to obtain the optimal set of user and location factors that best represent the interactions between users and locations. The basic assumption of MF based models is that each one of the d latent factors of a user/location are independent of each other. Also, the k^{th} factor of user interacts only with the k^{th} factor of location as shown in Figure 5.1. Further, MF based models assume equal weighted interaction of user and location factors. We can relax the assumption that latent factors interact independently by using a neural network model that can learn the interaction between all latent factors of a user and location.

In this next section, I will first present the basic component of a neural network based model i.e., an artificial neuron. Later, I will present how a neural network can be built using artificial neurons. Then, I will present how a neural network model can be trained to learn a specific task. Finally, I will describe how a neural network model can be used to learn the interaction between all latent factors of a user and location.

5.2.2 Neural Network Model

Artificial Neuron

An artificial neuron is a function that maps a given input to a desired output. Perceptron, is one of the first artificial neuron developed that takes multiple real

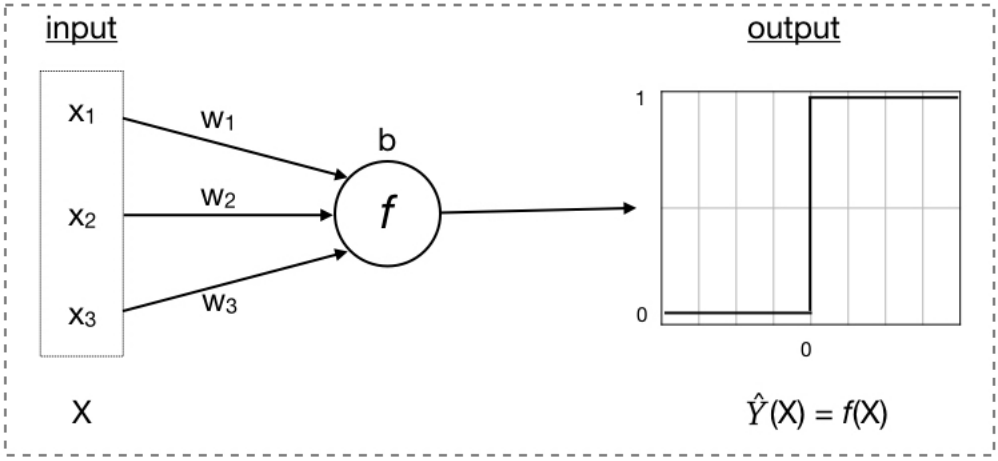


Figure 5.2. Perceptron neuron that operates on 3 inputs (X) and produces an output ($\hat{Y}_X$) with a certain bias b.

valued inputs and produces a binary output Rosenblatt [1962]. The perceptron also assign's importance to different inputs with the help of different weights. Figure 5.2 show a simple perceptron that assigns weights w_1, w_2 and w_3 to its three inputs x_1, x_2 and x_3 respectively. The output of the perceptron with a certain bias b is given by the following function:

$$f(X) = \begin{cases} 1 & \sum_i x_i w_i + b > 0 \\ 0 & \text{Otherwise} \end{cases} \tag{5.3}$$

$$\text{Input vector } X = [x_1 x_2 x_3]^T \tag{5.4}$$

The weights and bias of a perceptron control how it responds to the input and produces the desired output. Since, the perceptron's output is binary, it can be trained to learn a binary classifier.

A Sigmoid neuron is another artificial neuron that outputs a range of values between 0 and 1 instead of just 0 or 1. The output of sigmoid neuron with the same input vector X and bias b is given by the following sigmoid function (σ):

$$\sigma(X) = \frac{1}{1 + exp(-\sum_i x_i w_i - b)} \tag{5.5}$$

$$\text{Input vector } X = [x_1 x_2 x_3]^T \tag{5.6}$$

Figure 5.3 presents the shape of output of sigmoid neuron. The primary benefit of using a sigmoid neuron is that small changes in the input leads to small

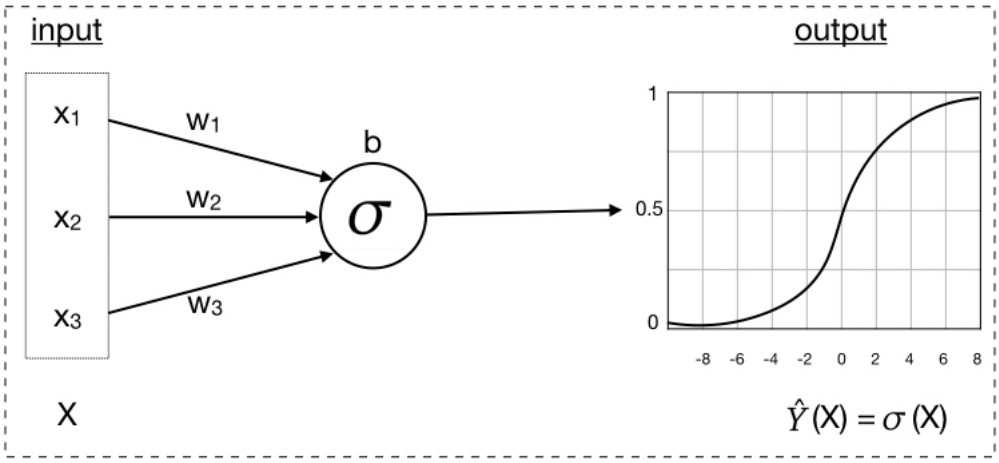


Figure 5.3. Sigmoid neuron that operates on 3 inputs (X) and produces an output ($\hat{Y}_X$) with a certain bias b.

changes in the output. Where as in the case of a perceptron small changes in the input leads to large variance in the output.

Neural Network Model

When multiple artificial neurons are connected they form an artificial neural network. Figure 5.4 shows a simple neural network with different layer of neurons. The leftmost layer is called input layer because it takes the input data. The rightmost layer is called output layer because it outputs data processed by all neurons. The middle layers are called hidden layers since neurons in these layers don't directly receive the input nor do they directly provide the output.

A Feed Forward Neural Network (FFNN) is a special form of neural network where all neurons in a layer feed information to neurons in the next layer only. Figure 5.4 is an example of a FFNN because all the neurons in a layer receive input only from the previous layer. My `DEEPREC` recommendation model also utilizes a FFNN.

I represent the output of a neural network model as a function of input and model parameters that can be represented as follows:

$$\hat{Y}(X) = f(X, \theta) \tag{5.7}$$

Where X is the input given to the neural network, $\hat{Y}(X)$ is the predicted output of the neural network, and θ represents the neural network parameters i.e., collection of weights and biases of all neurons in the network.

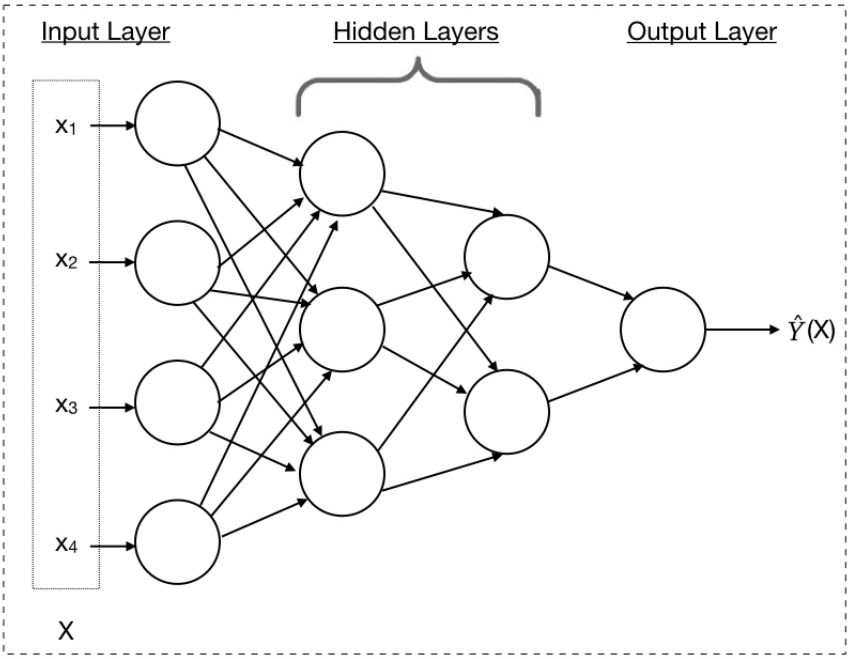


Figure 5.4. A Feed Forward Neural Network (FFNN) with input layer of 4 neurons, 2 hidden layer of neurons and an output layer with 1 neuron.

Training a Neural Network Model

The purpose of training a neural network model is to teach it a certain task. We can train a neural network model to classify input data. For example, if we want to train a neural network model to recognize pictures of cats then, the input is an image vector and the expected output is 1, if there is a cat in the image or 0 otherwise. By providing multiple pictures with and without cats we can train the neural network model as a binary classifier of images with cats. The Neural network model presented in figure 5.4 can be trained as a binary classifier for 4 dimensional input data.

The overall process to train a neural network model can be divided into 3 steps:

1. For each training sample pair i.e., input X and expected output $Y(X)$, feed the input X to neural network model and obtain the predicted output $\hat{Y}(X)$.

2. Define a cost function that will measure how far is the predicted output from the expected output; for example a quadratic cost function for the

neural network model presented in Figure 5.4 is defined as:

$$cost(\theta) = \sum_{\forall X} ||\hat{Y}(X) - Y(X)||^2 \tag{5.8}$$

Where θ is model parameters or collection of weights and biases associated with all neurons.

3. Utilize an optimization algorithm like gradient descent that minimizes the cost function by finding the optimal set of weights and biases of all neurons in the network.

$$\min_{\theta} cost(\theta) \tag{5.9}$$

The cost function and the optimization algorithms used to train my DEEPREC model is presented in Section 5.5.1.

5.3 A Neural Network model to learn time-independent preferences

As discussed in Section 5.2.1, matrix factorization based methods aim to obtain the optimal set of user and location factors that best capture the mobility behavior of users (refer to Figure 5.1). The primary drawback of these methods is that they assume independent interaction between the latent factors of users and location. We over come these limitations, by utilizing a neural network to learn the interaction between latent factors of all users and locations.

Let set $C^{'}$ be set of all check-ins of M users at N locations. Each element of set $C^{'}$ can be represented as $< u_i, l_j, t_{u_i,l_j} >$, where $u_i \in M$ and $l_j \in N$ and t_{u_i,l_j} represents the time stamp of check-in. Also, let $P \in \mathbb{R}^{Mxd}$ and $Q \in \mathbb{R}^{Nxd}$ represent the d latent factors of M users and N locations respectively.

Since, we want to learn the interaction between user and location factors, the input to FFNN-sigmoid will be the latent factor of any user u_i (i.e, $p_i \in \mathbb{R}^{1xd}$) and any location l_j (i.e., $q_j \in \mathbb{R}^{1xd}$). Figure 5.5 presents a sample FFNN with multiple hidden layers that capture the interaction between the latent factors and provide as output the predicted preference of user u_i for location l_j. The predicted preference ($\hat{r}_{i,j}$) can be defined by the following equation:

$$\hat{r}_{i,j} = \hat{Y}(X) = f(X, \theta_{ul}) = f(p_i, q_j, \theta_{ul}) \tag{5.10}$$

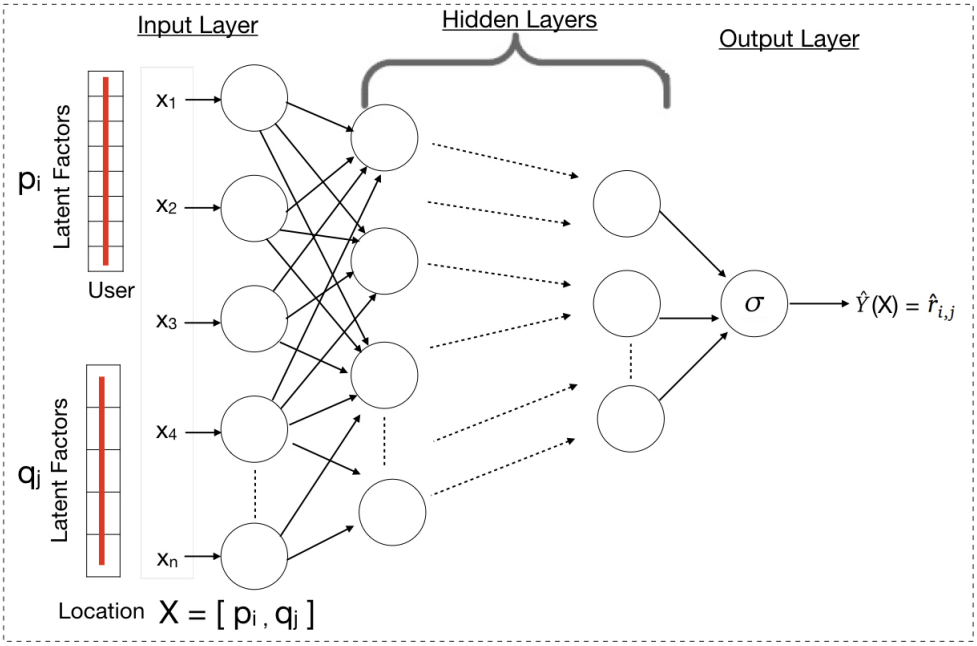


Figure 5.5. A Feed Forward Neural Network that learns the interaction between latent factors of any user u_i and any location l_j (FFNN-sigmoid). The predicted preference of user u_i for location l_j is given by the sigmoid output.

Where θ_{ul} is the collection of weights and biases associated with all neurons of the FFNN that learns the latent factor interaction between user and location. The expected output ($Y(X)$) of the FFNN is 1, if u_i visited l_j or the tuple $< u_i, l_j, > \in C^{'}$. While, the expected output ($Y(X)$) of the FFNN is 0, if u_i has not visited l_j or the tuple $< u_i, l_j, > \notin C^{'}$

In Section 5.5.1 I describe the cost function that will be used to train the above feed forward neural network model.

5.4 A Neural Network model to incorporate Spatial Constraints

Multiple studies on human mobility patterns have ascertained that humans tend to visit places close by Gonzalez et al. [2008b]; Papandrea et al. [2016]. A location recommendation model must capture the spatial constraints in people's mobility patterns by incorporating geographical distance between locations. In the context of my neural network model with latent factors as input, the geo-

graphical distance between locations must be able to captured by the network. The network must ensure that geographically close locations are given similar preference than compared to geographically far locations.

Let $l_j \in N$ and $l_k \in N$ be two locations visited by user $u_i \in M$. Also, let $GPSCoord_j$ and $GPSCoord_k$ be the GPS coordinates of locations l_j and l_k respectively. Let G be the number of groups into which the geographical distance between any pair of locations is normalized. For example, if we normalize geographical distance in two groups i.e., $G = 2$, then one group will contain all location pairs that are within a distance of less than x km and another group will contain all location pairs where distance between then is greater than x km. Figure 5.6 shows a sample FFNN with 2 hidden layers and a softmax output layer that categorizes location pairs in two groups. The softmax layer outputs a normalized probability distribution over 2 different possible outcomes. In Figure 5.6, where $G = 2$ with input X, the softmax output $\hat{S}(X)$ will be a 2-dimensional vector where, $\hat{S}(X)[1]$ represents the predicted probability that the distance between the two locations is less than 1Km and $\hat{S}(X)[2]$ represents the predicted probability that distance is greater than 1Km. Thus, if the geographical distance between two locations is less than 1Km and if the FFNN network is able to correctly group it then the expected output $S(X)$ of the softmax is $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$. More generally, the output of a softmax layer is a G-dimensional vector of probabilities $\hat{S}(X)$ where the elements sum to 1 i.e., $\sum_i^G \hat{S}(X)[i] = 1$.

Figure 5.7 presents a generic FFNN with multiple hidden layers that capture the interaction between the latent factors of any two locations and provide as output a G-dimensional vector of probabilities that represents the probability distribution of geographical distance over G different groups. The predicted output is the group with highest probability and can be written as:

$$\hat{Y}_s(X) = \underset{i}{\text{argmax}}\ \hat{S}(X)[i] \tag{5.11}$$

I represent the entire feed forward neural network model with softmax output as a function (g) of input and model parameters that can represented as follows:

$$\hat{S}(X) = g(X, \theta_{ll}) \tag{5.12}$$

Where θ_{ll} is the collection of weights and biases associated with all neurons of the FFNN including the softmax output layer.

In Section 5.5.1 I describe the objective or cost function that will be used to train the above feed forward neural network model.

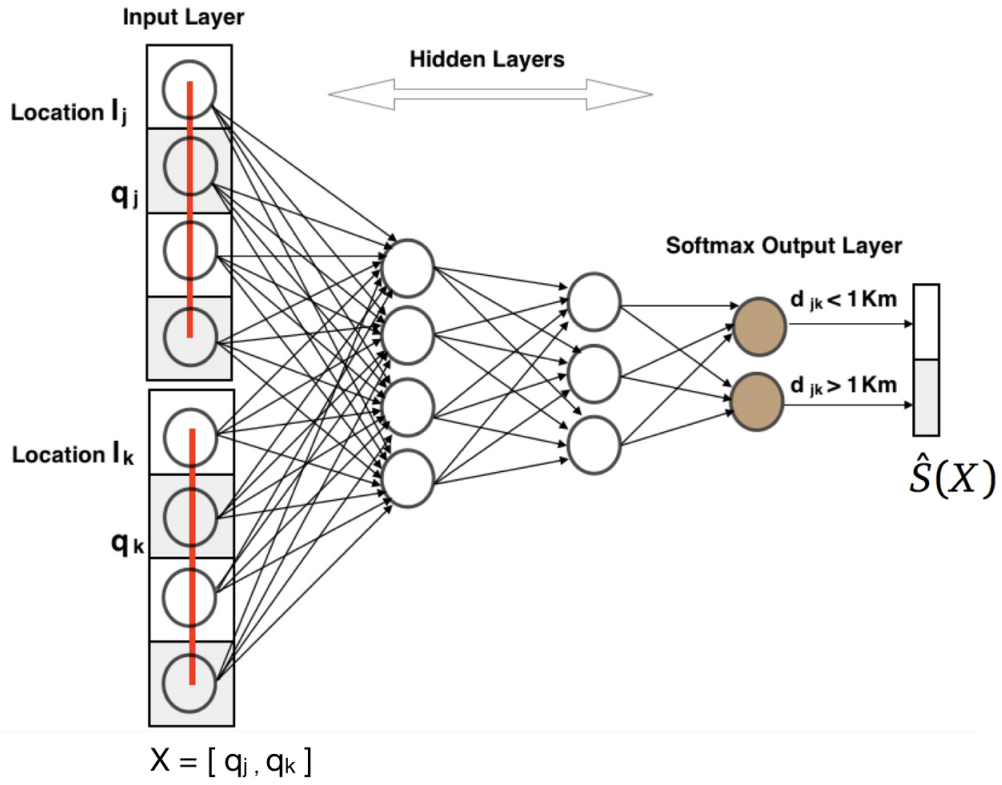


Figure 5.6. A Feed Forward Neural Network model that constraints the factors of two locations l_j and l_k based on their geographical distance $d_{j,k}$ into two groups

5.5 DEEPREC: A joint neural network model

In Section 5.3, I have presented how we can utilize a neural network model to learn the time-independent preferences or latent factors of a user for a location. In Section 5.4, I have presented another neural network model that can enforce constraints on the latent factors of two locations that are geographically close to each other. Figure 5.8, presents a combined neural network model that will be used to jointly learn time-independent preference of a user for a location along with enforcing geographical constraints on latent factors of locations. I refer to this combined neural network model used to recommend locations as `DEEPREC` .

The feed forward neural network model that learns the latent factor interaction between a user and location shown in Figure 5.5 is represented as FFNN-sigmoid in Figure 5.8 because the output layer contains one sigmoid neuron. Similarly, the feed forward neural network model that enforces geographical con-

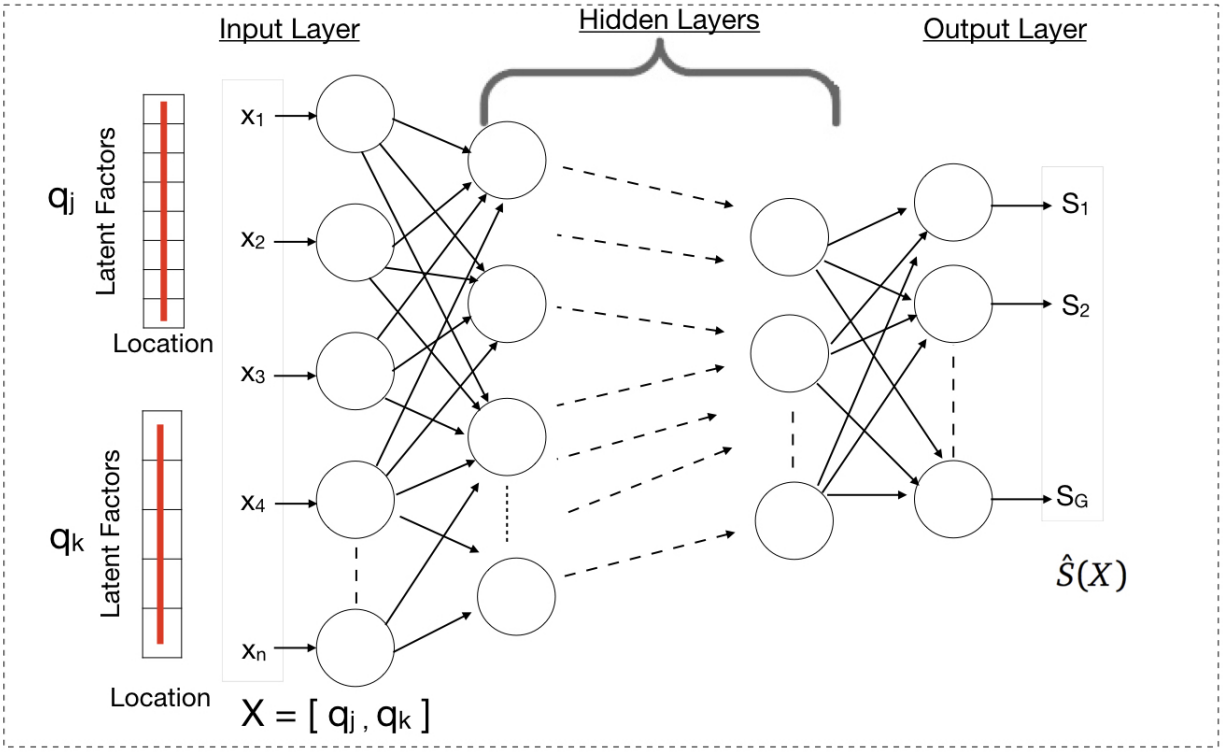


Figure 5.7. A Feed Forward Neural Network model that groups any two locations l_j and location l_k based on their geographical distance into G groups (FFNN-softmax)

straints between latent factors of two locations shown in Figure 5.7 is represented as FFNN-softmax in Figure 5.8 because the output layer is a softmax that outputs a normalized probability distribution.

5.5.1 Training DEEPREC model

Every location visited by a user represents the implicit preference of the user for those locations. Similarly users who visit a particular location represent the characteristic of that location. So we can utilize each check-in to learn the time-independent location preference of a user. Since our goal is also to constrain the latent factors of geographically close locations I perform pair-wise sampling of training data i.e., for each user u_i I consider two check-ins: 1) a positive sample i.e., user has visited the location l_j or $< u_i, l_j, > \in C^{'}$, and 2) a negative sample i.e., user has not visited the location l_k or $< u_i, l_k, > \notin C^{'}$. A negative sample is created by randomly sampling the un-visited locations of a user. I train the

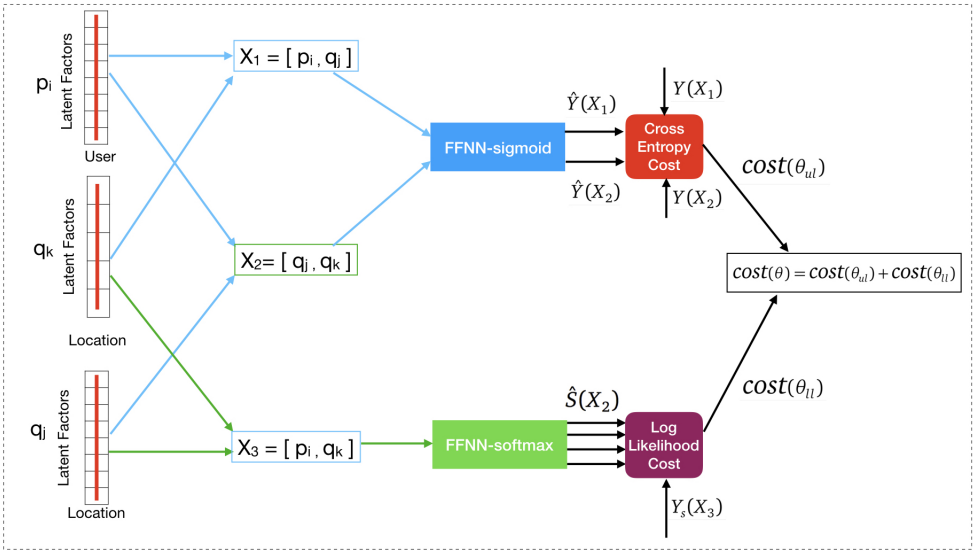


Figure 5.8. DEEPREC model learns the interaction of a user u_i's latent factor with latent factors of two locations l_j and l_k using FFNN-sigmoid network. The FFNN-softmax network incorporates geographical distance between locations l_j and l_k.

neural network model with both positive and negative sample to ensure that the model does not over fit to training data.

Therefore for each triplet (u_i, l_j, l_k) we need to train two neural networks:

1. A feed-forward network that captures the time-independent location preference of the user (Figure 5.5)

2. A feed-forward network that constraints the latent factors of geographically close locations (Figure 5.7).

The overall process to jointly train both the neural network models can be divided into 3 broad steps:

Step 1: For each training triplet (u_i, l_j, l_k). I combine the latent factors of user u_i and locations l_j, l_k to create vectors X_1 and X_2 respectively. I combine the latent factors of two locations l_j, l_k to create vector X_3 (as shown in Figure 5.8).

   (a) I feed the input X_1 to FFNN-sigmoid network to obtain the output $\hat{Y}(X_1)$ that represents the predicted preference of user u_i for location l_j. Since location l_j is a part of the training data of u_i (or positive sample), the expected output $Y(X_1)$ is 1

(b) I feed the input X_2 to FFNN-sigmoid network to obtain the output $\hat{Y}(X_2)$ that represents the predicted preference of user u_i for location l_k. The expected output $Y(X_2)$ is 0 since location l_k is not part of the training data of u_i (or negative sample).

(c) I feed the input X_3 to FFNN-softmax network to obtain a G-dimensional output $\hat{S}(X_3)$ that represents the predicted probability distribution of geographical distance between locations l_j and l_k over G different groups. The expected output $Y_s(X_3) \in [0, G]$ is actual group which belongs to the geographical distance between locations l_j and l_k.

Step 2: Define a cost function that will measure how far is the predicted output from the expected output. We utilize a Cross Entropy Cost function (described below) to train FFNN-sigmoid network and a Log Likelihood Cost function (described below) to train the FFNN-softmax network.

(a) Cross Entropy Cost

$$cost(\theta_{ul}) = \begin{cases} -\log(\hat{Y}(X_1), \text{ for positive sample} \\ \\ -\log(1-\hat{Y}(X_2))), \text{ for negative sample} \end{cases}$$

θ_{ul} is collection of weights and biases of all neurons in the FFNN-sigmoid network

(b) Log Likelihood Cost

$$cost(\theta_{ll}) = -\log(\hat{S}(X_3)[Y_s(X_3)])$$

θ_{ll} is collection of weights and biases of all neurons in the FFNN-softmax network

The cost function of entire joint model is sum of individual cost functions.

$$cost(\theta) = cost(\theta_{ul}) + cost(\theta_{ll})$$

Where θ is model parameters or collection of weights and biases associated with all neurons in the joint model shown in Figure 5.8.

Step 3 Utilize Adagrad Duchi et al. [2011] optimization algorithm to minimize the cost function to obtain the optimal set of weights and biases of all neurons in the network.

$$\min_{\theta} cost(\theta) \tag{5.13}$$

The process of training DEEPREC is detailed in Algorithm 4

Algorithm 4: Training Procedure of DEEPREC

foreach *triplet* (u_i, l_j, l_k) **do**

/* Prepare input for Neural Network models */

Append latent factors of u_i and l_j to create vector X_1

Append latent factors of u_i and l_k to create vector X_2

Append latent factors of l_j and l_k to create vector X_3

/* Feed-Forward Neural Network (FFNN) models */

Input X_1 and X_2 to FFNN-sigmoid and compute cost : $cost(\theta_{ul})$

Input X_3 to FFNN-softmax and compute cost : $cost(\theta_{ll})$

/* Total Cost */

$cost(\theta) = cost(\theta_{ul}) + cost(\theta_{ll})$

/* Train FFNN models */

/* Compute Gradients to update parameters */

Gradient w.r.t to FFNN-sigmoid model parameters : $\frac{\partial cost(\theta)}{\partial \theta_{ul}}$

$\theta_{ul} = \theta_{ul} - \frac{\partial cost(\theta)}{\partial \theta_{ul}}$ // update parameters

Gradient w.r.t to FFN-softmax model parameters : $\frac{\partial cost(\theta)}{\partial \theta_{ll}}$

$\theta_{ll} = \theta_{ll} - \frac{\partial cost(\theta)}{\partial \theta_{ll}}$ // update parameters

end

5.5.2 Recommend locations using DEEPREC

Once the FFNN-sigmoid and FFNN-softmax neural networks part of my DEEPREC model are trained, I utilize the FFNN-sigmoid to recommend new locations to user. The overall process to recommend new locations to users is given by the following four steps:

1. For each user in the training data, obtain the list of un-visited locations i.e., new locations where the user has not checked in in the training period.

2. Obtain the latent factors of all those un-visited locations along with the latent factor of the user that is being given a recommendation.

3. Utilizing the FFNN-sigmoid neural network compute the predicted preference of the user for each one of its un-visited locations.

4. The top-K un-visited locations with highest scores are recommended to the user.

5.6 Evaluation

In this section, I first describe the dataset used to evaluate the effectiveness of my DEEPREC model to recommend new locations to users. I also describe the procedure to evaluate DEEPREC model. I measure the the performance of DEEPREC model using Normalized Discounted Cumulative Gain (NDCG) metric and compare it with baseline recommendation models.

5.6.1 Dataset

The dataset was constructed by crawling SinaWeibo (China's Twitter), the largest social networking website in China, that provides location-based services such as check-ins. Sina Weibo allow their user's to share their check-ins publicly through their tweets. This dataset was constructed by my colleagues at Microsoft Research Asia. Table 6.1 presents the characteristics of dataset constructed using Sina Weibo open APIs from January 2012 to August 2013.

Table 5.1. Sina Weibo Dataset Characteristics

	Sina Weibo
Time period of Check-ins	599 Days
Number of Users	325,447
Number of Locations	79,592
Number of Check-ins	2,348,571

5.6.2 Evaluation procedure

In my evaluation, I first clean the dataset by filtering out all those users and locations who have less than 10 check-ins in the entire dataset. I split the entire

check-in data by time into two parts. All the check-ins between January 2012 to May 2013 were used to train the model, while check-ins from June 2013 to August 2013 were used to test the model. The training data was used to jointly train the FFNN-sigmoid and FFNN-softmax neural networks that constitute my DEEPREC model. For each user, we use the trained FFNN-sigmoid neural network model to compute their predicted preference for all their un-visited locations as discussed in Section 5.5.2. The top-K un-visited locations with highest preference is recommend to the user.

5.6.3 DEEPREC model parameters

I evaluated the performance of DEEPREC that learns the time-independent interaction between latent factors of users and location with different number of hidden layers. Based on multiple experiments I found the best performance for the following neural network parameters:

> FFNN-sigmoid: A neural network with 3 hidden layers. The structure of the neural network to learn the time-independent interaction between user and location factor is: [512, 256, 128, 64, 1]. The first number (512) represents the size of input layer, the next three numbers represent the sizes of three hidden layers. The 5th number represents one sigmoid neuron that takes as input the output of fourth hidden layer (i.e., 64). Since the input is a joint vector of user and location factors, the size of user/location latent factor is a vector of length 256 (512/2).
>
> FFNN-softmax: A neural network with 1 hidden layer. The structure of neural network to learn the geographical distance between latent factors of locations is: [512, 256, 200]. The first number (512) represents the size of input layer, the second number represents the sizes of hidden layer. The 3rd number (200) represents the number of groups into which the euclidean distance between two locations is divided into. A group corresponds to locations that are within a certain distance. Group 0 corresponds to all pairs that are within [0-2km), group 1 corresponding to all pairs that are withing [2-4km), group 3 corresponding to all pairs that are withing [4-6km) and so on. Since the input to FFNN is a joint vector of user and location factors, the size of user/location latent factor is a vector of length 256 (512/2).

Utilizing the insights obtained by Glorot and Bengio [2010b] in the difficulty to train deep neural networks. I use Xavier method (Glorot and Bengio [2010a])

to initialize the model parameters i.e., weights and biases of all neuron in my DEEPREC model. I use $tanh$ activation function (Goodfellow et al. [2016]) for all neurons in the DEEPREC joint model. For all my experiments, I fix the learning rate to 0.01 i.e., all the model parameters of FFNN-sigmoid and FFNN-softmax are updated in steps of 10^{-2}. To improve the learning process, I employ the dropout technique Srivastava et al. [2014] at the output layer with the dropout rate set to 0.5. I train the entire model for one epoch and observe the results to surpass the state of the art location recommendation algorithms. I utilize Adagrad Duchi et al. [2011] optimization method to find the optimal values for model parameters.

5.6.4 Performance metrics

I evaluated the performance of DEEPREC model and all baseline algorithms using a Normalized Discounted Cumulative Gain (NDCG) (Järvelin and Kekäläinen [2002]) metric that jointly measures not only whether a user has visited a recommended location but also takes into account the ranking of recommended location in entire recommendation list.

Discounted cumulative gain (DCG) is a metric that captures the quality of ranking. The metric takes into account the order in which a relevant location appears in the recommend set. For example, let user A be recommended a set of two locations P, Q by two different recommendation models M1 and M2. Let the recommended set by model M1 be: [Q P], while model M2's recommended set be: [P Q]. Also, let us assume that user A has visited location Q in future. Given that both models recommend location Q, the traditional performance metric value i.e, *Precision* (refer Section 3.5.2) would be equal because location Q appears in the set of locations recommended by both models. Therefore, we cannot use *Precision* metric to measure which recommendation model is better. However, we know that the rank of location Q in the list recommended by model M1 is 1 while, in the case of model M2 the rank of Q is 2. Therefore, if we consider the rank of location part of the recommended set then model M1 must be assigned a higher performance metric that compared to model M2. We can capture the ranking of locations in recommend set using DCG metric.

Let $S_u(k)$ be the set of top K locations recommended to a user u and V_u be the set of new locations visited by user in the testing data. The DCG of set of

locations recommended to user u is given by the following equation:

$$\text{DCG}_u@K = \sum_{i=1}^{K} \frac{2^{r_i} - 1}{log_2(i+1)},\ r_i = \begin{cases} 1 \text{ if } S_u(i) \in V_u \\ \\ 0 \text{ Otherwise} \end{cases} \tag{5.14}$$

The DCG of the testing set is called Ideal discounted cumulative gain (IDCG) i.e., what would be the ideal DCG when all the locations in testing set are part of the recommended set in the exact same order. The IDCG of set of un-visited locations part of the testing data of user u is given using the following equation:

$$\text{IDCG}_u = \sum_{i=1}^{|V_u|} \frac{1}{log_2(i+1)} \tag{5.15}$$

The Normalized Discounted Cumulative Gain (NDCG) of the set of recommendations provided to all users is computed as follows:

$$\text{NDCG@K} = \frac{1}{N} \sum_{u=1}^{N} \frac{\text{DCG}_u@K}{\text{IDCG}_u} \tag{5.16}$$

5.6.5 Baseline Algorithms

I compared the performance of `DEEPREC` model with the following standard filtering based recommendation models:

- LibFM Rendle [2012] is a factorization based model that characterizes users and locations by a vector of factors inferred from the user-location visits. The vectors are in the same latent space and, the preference of a user for locations is modeled as inner products in that space. In my evaluation, I perform a grid search of factor size over $Factors \in \{32, 64, 128, 256\}$ to select the best parameters.

- GeoMF Lian et al. [2014] is another factorization based model that augments the user's and location's latent factors to incorporate the spatial constraints. In my evaluation, I perform a grid search of factor size over $Factors \in \{16, 32, 64, 128, 256\}$ to select the best parameters.

- SVD++ Koren [2008] is another factorization based model that also incorporates the similarities between different items or locations. In our evaluation, we evaluate it for a factor of size 16.

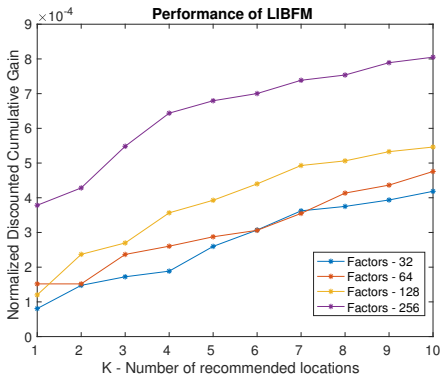


Figure 5.9. Performance of LIBFM model for its different parameters.

5.6.6 Performance Comparison

Figures 5.9, 5.10, 5.11 present the Normalized Discounted Cumulative Gain for different number of recommended locations (i.e., K) for different baseline algorithms. In general the NDCG increases with K because of an increase in the possibility that top-K locations could be part of the set of new locations visited by user. In other words, the Discounted Cumulative Gain increases with K while the Ideal Discounted Cumulative Gain remains constant.

In Figure 5.9, we observe that the best performance for LIBFM recommendation is obtained when it models the preferences of a user/location with a latent factor of size 256. As the size of factor is increased the basic factorization model is able to capture the check-ins of users at different location much better than compared to smaller factor size.

In Figure 5.10, for GeoMF model we observe that the performance is very similar for different latent factors this indicates that GeoMF is able to capture the user and location preferences with small number of latent factors and increasing the number of factors does not have a significant positive impact on the recommendation performance. In Figure 5.11, a similar increase of NDCG with K is observed in the recommendation performance of SVD++ model.

Comparing the performance of different models, we observe that for our Sina Weibo dataset (described in Table 6.1), the performance of LIBFM is better than SVD++ and performance of GeoMF is even better than compared to LIBFM.

5.6.7 Performance of DEEPREC model

In my `DEEPREC` model the size of latent factor that is to learned using a neural network for each user and location is 256 (refer to Section 5.6.3). Fig-

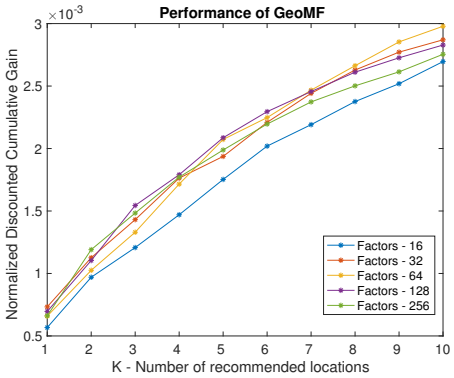


Figure 5.10. Performance of GeoMF model for its different parameters.

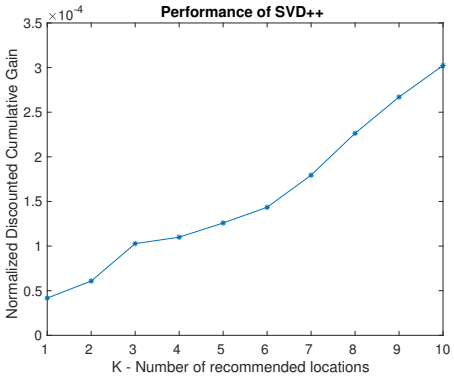


Figure 5.11. Performance of SVD++ model for a factor of size 16.

ure 6.7 presents the comparative performance of my Feed forward neural network based recommendation model i.e., FFNNREC with the baseline algorithms. We can clearly observe that my neural network based model outperforms baseline models with a significant margin. In order to show the relative performance I present the values of NDCG metric in log scale. When 10 locations are recommended to each users DEEPREC model performs 10 times better than compared to GeoMF model which is the best among the baselines considered.

The performance of my DEEPREC model is better that other factorization based model because of the neural network's ability to better learn the interaction between all the latent factors of a user and location. A small neural network with just 3 hidden layers is able to learn the interaction better than compared to other factorization based models. How is it able to learn better has been explained in detail in Section 5.3. In addition, my FFNN-softmax neural network that is a part of DEEPREC model is able to capture the spatial preferences of user by constraining the latent factors of geographically close locations.

These experiments clearly highlight the ability of neural networks to model the time-independent preferences of users and locations.

Comparison with REGULA

The SinaWeibo dataset used to evaluate my DEEPREC model is very large as it contains check-ins gathered over a duration of 599 days. As discussed in Section 3.4, REGULA computes scores to each un-visited location before it provides a recommendation to user. Due to time and space complexity, I could not train REGULA for such a large dataset and therefore evaluate it on a smaller subset of SinaWeibo dataset.

I created a smaller dataset that contained check-ins between January 2012 to April 2012. All check-ins between January and March 2012 were considered as training data and testing data contained all check-ins in the month of April 2012.

The NDCG@10 for REGULA is 0.0152, while for my DEEPREC model it was 0.0243 which is 1.6 times better than compared to the feature based REGULA model. DEEPREC outperforms REGULA because it is able to capture the time-independent preferences of users for different location better by learning the interaction between the latent factors of users and locations. DEEPREC is also able to incorporate the spatial constraints between geographically close locations while REGULA does not incorporate those spatial constraints, it only uses them to filter out candidate set of locations.

Based on my analysis and comparison I conclude that DEEPREC provides better

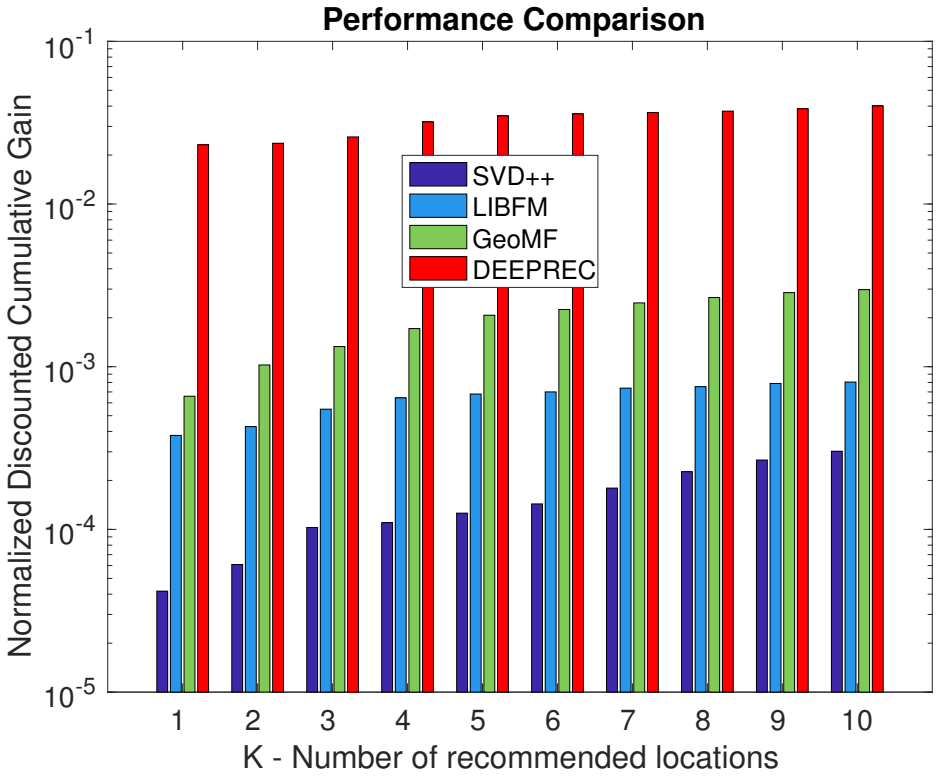


Figure 5.12. Comparison between the performance of DEEPREC, LibFM, GeoMF and SVD++ for their best parameters

recommendations that compared to state-of-the-art recommendation models.

5.7 Conclusion and Limitations of DEEPREC

In this chapter, I first presented a brief overview of how factorization based recommendation algorithms model the time-independent preferences of users and locations. Later, I described the general procedure to obtain these latent factors. I also presented the basic components of a neural network model and how it can be used to obtain better factors compared to matrix factorization based models.

I presented two neural network models: 1) FFNN-sigmoid that was used to learn the latent factors of users and locations based on the check-in information, 2) FFNN-softmax that incorporates spatial constraints into latent factors of locations. I combine the two models to form a `DEEPREC` and jointly train them using two cost functions: Cross Entropy and Log likelihood using ADAGRAD op-

timization method. I presented the comparative performance of DEEPREC with three state-of-the-art factorization based models and show that it outperforms them due to neural network's ability to identify better latent factors to capture the preferences of users and locations. The key findings of this chapter are:

- I present how the time-independent preferences of users and locations represented as latent factors can be learned using a combination of two feed forward neural network models.
- To the best of my knowledge, DEEPREC model is the first to capture the geographical constraints between locations using a feed forward neural network with softmax output model.
- DEEPREC performs 10 times better than state-of-the-art GeoMF recommendation model due to the inherent ability of a neural network to model the function that captures the interaction between latent factors of users and locations.

5.7.1 Limitations

The output of a feed forward neural network (FFNN) depends on fours things: 1) the input information, 2) input layer i.e., weights and biases of input neurons, 3) hidden layers i.e., weights and biases of all hidden neurons, and 4) output layer i.e., weights and biases of output neurons. Therefore, the predicted preference of a user for location depends only on the current latent factors and the current state of the entire neural network. In other words, the output of FFNN at time t does not have any impact on the output of FFNN at time $t+1$.

Further, a FFNN does not store the previous network states (or neuron weights and biases) and therefore cannot capture the historical interaction between users and locations. Therefore a FFNN can only capture time-independent interaction between the latent factors of user and location. In this section, since we are interested to only to learn the time-independent preferences of users and location, I don't utilize the time information in training the FFNN. In next chapter, I will present another deep learning based model called DEEPTREC that models the time-dependent preferences of users.

5.8 Remarks

The work presented in this Chapter was done in collaboration with Dr. Defu Lian (Big Data Research Center, UESTC, Chengdu, China), Dr. Xing Xie (Microsoft

Research Asia, Beijing, China) and Danyang Liu (Ph.D. student of Dr. Xing Xie) and it is under submission to Ubiquitous Intelligence and Computing (UIC 2018) conference.

Chapter 6

Deep Recurrent Neural Network based Recommendation Model

6.1 Introduction

Humans are creatures of habit and exhibit time dependent mobility patterns Noulas et al. [2011]; Cho et al. [2011b] i.e., type of locations visited in the afternoon is different from those visited in the evening. Therefore, a model that recommends new locations to people must learn the time varying mobility behavior of a user and recommend new locations accordingly.

In previous chapters, I have presented REGULA , a feature based model and DEEPREC , a feed forward neural network based model that can provide time-independent location recommendations i.e, the set of recommendations do not change with time. While, in this chapter, I first introduce neural network models that are able to learn and memorize patterns in data. These unique models are called Recurrent Neural Networks (RNN). Specifically, I present the Gated Recurrent Unit (GRU) model that I used learn the temporal mobility patterns of users. One GRU model for each user and location is used to learn the temporal preferences of users and locations. I refer to the combination of GRU models that learn time-dependent preferences both users and locations as DEEPTREC

This chapter is structured as follows: In Section 6.2 I will present the basic structure of a generic recurrent neural network model along with the GRU model. Section 6.3 presents the DEEPTREC model that is a combination of user's and location's GRU models and can be trained to learn the time dependent preferences of users and locations. Section 6.3, presents the procedure to train the DEEPTREC model. Section 6.4, presents how I combine the time-independent model DEEPREC and time dependent model DEEPTREC to form a JOINTDEEPREC

model that learns the combined preferences of users and locations. Section 6.5.2, presents the evaluation of JOINTDEEPREC and its comparative performance with state-of-the-art recommendation models along with REGULA on a large dataset that contains check-ins of users located in the city of Beijing, China. Finally, Section 6.6 concludes the Chapter.

6.2 Recurrent Neural Network Model

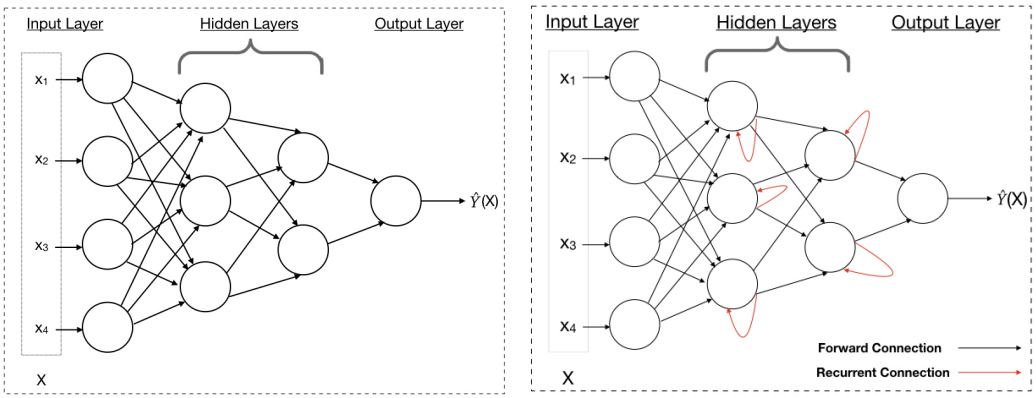
Multiple experiments on human mobility patterns Noulas et al. [2011]; Cho et al. [2011b] and my own analysis on different Location Based Social Networking (LBSN) datasets like Gowalla, Brightkite has shown that the preference of users for different locations can be defined as a combination of: 1) stationary preferences that don't change with time and 2) temporal preference that differ with time. In Section 5.5 of Chapter 5, I described how stationary preferences of users and location can be obtained by using DEEPREC , a feed forward neural network based model. To learn the temporal preferences of users and locations I utilize a recurrent neural network based model.

6.2.1 Artificial neuron network with memory

The Recurrent Neural Network(RNN) is a class of neural network that allows cyclical connections between input and output. Unlike the feed-forward neural network that maps the input to output, a RNN maps the historical input to each output. Figure 6.1(a) presents a sample Feed Forward Neural Network (FFNN) model where the output of neurons in one layer is given as input to neurons in the next layer and Figure 6.1(b) shows a similar Recurrent Neural Network (RNN) model where neurons in a layer also have cyclical connections i.e., the current output of neuron depends and the historical input and output of the neuron. Further, RNNs have memory that enables them to capture historical states of neurons and is referred to as hidden state of the RNN.

Figure 6.2 presents a generic RNN model where, the hidden state (h_t) of the RNN model gets updated based on the current input (x_t) and previous hidden states (h_{t-1}). The output (y_t) of the RNN model depends on the current input (x_t) and updated hidden state (h_t). The followings equations define the parameters of a generic RNN model:

$$h_t = \phi(W_{xh}x_t, U_{hh}h_{t-1})$$
$$y_t = \psi(W_{xy}x_t, W_{hy}h_t)$$



(a) FFNN (b) RNN

Figure 6.1. A sample Feed Forward Neural Network (FFNN) and a similar Recurrent Neural Network (RRN) with cyclical connection between neurons.

The parameter W_{xh} controls how input impacts the hidden state or memory of the network and the parameter U_{hh} controls how the hidden state gets updated. The parameters W_{xy} and W_{hy} control how output of the model is impacted by the input and hidden states respectively. The functions ϕ, ψ and the different parameters of the model depend on the specific type of unit being used to build a RNN.

6.2.2 Gated Recurrent Unit (GRU) based RNN

Long short-term Memory (LSTM) proposed by Hochreiter and Schmidhuber [1997] is one type of basic unit that can be used to build a RNN. LSTM is one of the most popular and widely used RNN units that have been used in a wide range of applicaitons from speech recognition to image processing. Gated Recurrent Unit(GRU) proposed by Chung et al. [2015] is an improved model that can be trained to learn long term sequences better than LSTM and also overcomes the problem of gradient loss better than an LSTM unit. Figure 6.3 presents the architecture of a GRU unit. It consists of a cell, a reset gate and a forget gate. The cell state (h_t) is responsible to store historical values of the unit, it is also referred to as memory of GRU. The reset gate (r_t) is an sigmoid neuron that controls how the cell state gets reset or updated based on the input. The forget gate (z_t) is also a sigmoid neuron that controls how information stored in the cell state is removed or thrown away. The following equations define how GRU unit operates on the input, outputs the hidden state of the cell and records previous

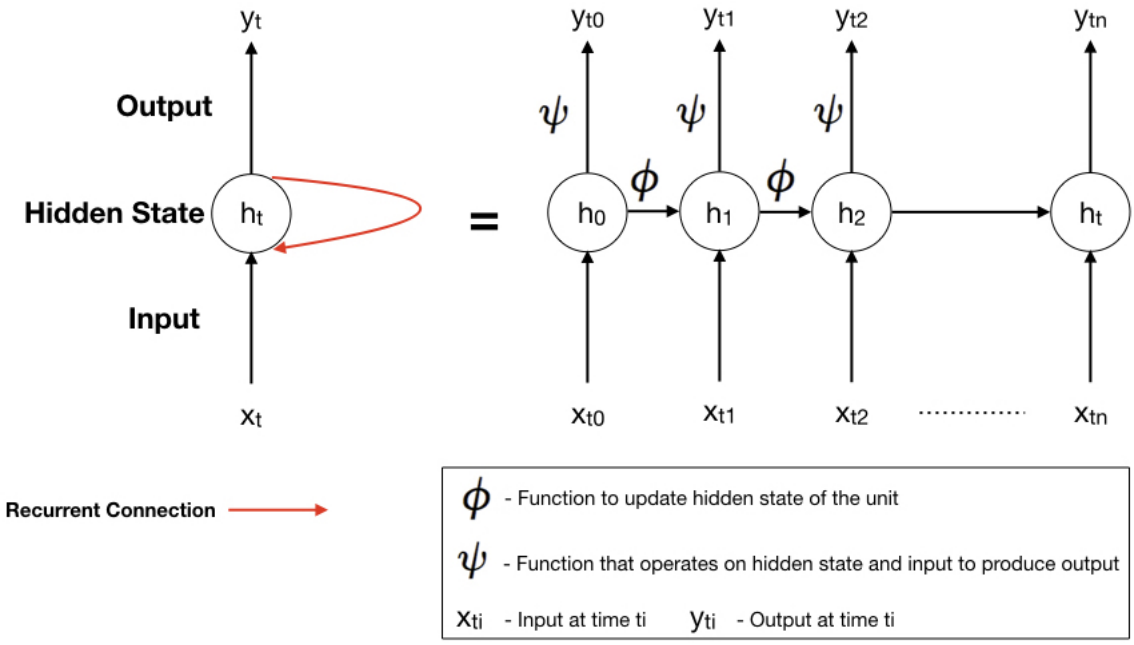


Figure 6.2. A generic Recurrent Neural Network model that updates its hidden states

hidden states.

$$r_t = \sigma(W_{xr}x_t + W_{hr}h_{t-1} + b_r) \tag{6.1}$$
$$z_t = \sigma(W_{xz}x_t + W_{hz}h_{t-1} + b_z) \tag{6.2}$$
$$\tilde{h}_t = \tanh(W_{xh}x_t + W_{hh}(r_t \odot h_{t-1}) + b_h) \tag{6.3}$$
$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \tag{6.4}$$

These functions control how the GRU model is going to remember the previous hidden states and how its going to reset its memory or cell state. I represent the above set of equations with the following equation

$$h_t = \phi^{GRU}(h_{t-1}, x_t, \theta^{GRU}) \tag{6.5}$$

Where θ^{GRU} represents the model parameters of GRU i.e., the weights and biases associated with activation functions described in Equations 6.1 to 6.3. Since, the output of an RNN is a function of its hidden state and its input, I represent the output of a GRU with the following equation:

$$y_t = \psi^{GRU}(h_t, x_t, \theta^{GRU}_{out}) \tag{6.6}$$

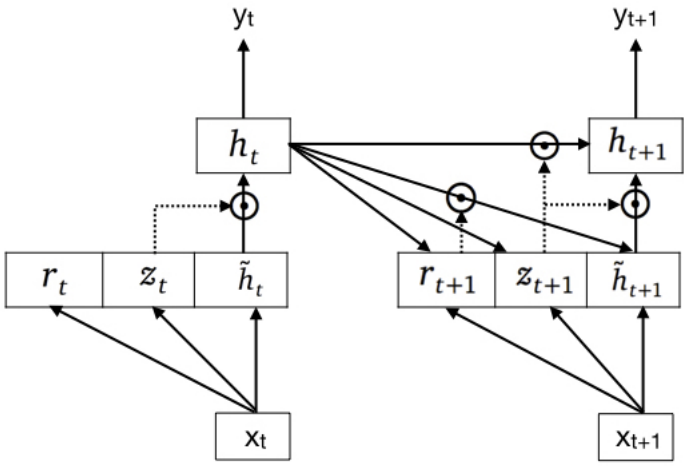


Figure 6.3. Architecture of Gated Recurrent Units (GRU) with different units

Where θ_{out}^{GRU} represents the parameters of a neural network that operate on the hidden state of a GRU (h_t) and input (x_t) to the model.

I utilize one GRU model for representing every user and one GRU model for representing every location. Since the GRU has the ability to remember the previous hidden states we could teach a user GRU model to learn from the sequence of locations visited by the user. Similarly we can teach a location GRU model to learn about the sequence of users that have visited it. The goal is to utilize the hidden states of user and location GRU's to predict which user might visit a location in the next step.

6.3 DEEPTREC: A GRU based RNN to learn time-dependent preferences

Let set $C^{'}$ be set of all check-ins of M users at N locations. Each element of set $C^{'}$ can be represented as $< u_i, l_j, t >$, where $u_i \in M$ and $l_j \in N$ and $t \in \mathbb{N}^{+}_{<8} = 1, 2, 3, 4, 5, 6, 7$ represents the check-in's day of week (i.e., Monday/Tuesday/.../Sunday) . Let $P^{'} \in \mathbb{R}^{Mxd}$ and $Q^{'} \in \mathbb{R}^{Nxd}$ represent the d latent factors of M users and N locations respectively that capture the temporal preferences. Since, I want to learn the time-dependent preferences of all users I represent their daily preferences with a latent factor matrix $T^{U}_{day} \in \mathbb{R}^{7xd}$. Similarly, let $T^{L}_{day} \in \mathbb{R}^{7xd}$ represent the d temporal latent factors of all locations. The 7 different factors represent the preferences for each day of the week. Based on my experiments I found that a much higher granularity i.e., 24 hours is not feasible

because we do not have the hourly visiting patterns of all users.

The latent factor of user u_i can be represented as $p_i^{'} \in \mathbb{R}^{1xd}$ which is the i^{th} row of matrix $P^{'}$. The latent factor of location l_j can be represented as $q_j^{'} \in \mathbb{R}^{1xd}$ which is the j^{th} row of matrix $Q^{'}$. The temporal latent factor of all users for k^{th} day of week ($k \in [1,7]$) can be represented as $t_k^U \in \mathbb{R}^{1xd}$ which is the k^{th} row of matrix T_{day}^U. The temporal latent factor of all locations for k^{th} day of week ($k \in [1,7]$) can be represented as $t_k^L \in \mathbb{R}^{1xd}$ which is the k^{th} row of matrix T_{day}^L. The elements of vector $q_j^{'}$ measure the extent to which location l_j posses the d factors. While the elements of vector $p_i^{'}$ measure the interest of user u_i for such d factors or features.

6.3.1 A GRU model for each user

The location preferences of a user is learned based on the locations visited by the user itself. Since, each user has different visiting patterns I use separate GRU models for each user. When a user u_i visits a location l_j on k^{th} day of the week, the input to the u_i's GRU will be the latent factor of location l_j i.e., $q_j^{'} \in \mathbb{R}^{1xd}$ and the temporal latent factor for k^{th} day of week i.e., $t_k^U \in \mathbb{R}^{1xd}$.

Using Equation 6.5, the change in hidden states and output of u_i's GRU model can be represented by the following equation:

$$h_{i|t+1} = \phi_i^{GRU}(h_{i|t}, q_j^{'}, t_k^U, \theta_i^{GRU}) \tag{6.7}$$

Where θ_i^{GRU} represents the model parameters of user u_i's GRU. $h_{i|t}$ represents the hidden state of u_i's GRU at time t.

6.3.2 A GRU model for each location

The preferences of a location is learned based on the users who visit the place. Since, each location is visited by different set of users I use separate GRU models for each location. When a location l_j is visited by user u_i on k^{th} day of the week, the input to the l_j's GRU will be the latent factor of user u_i i.e., $p_j^{'} \in \mathbb{R}^{1xd}$ and the temporal latent factor for k^{th} day of week i.e., $t_k^L \in \mathbb{R}^{1xd}$.

Using Equation 6.5, the change in hidden states and output of l_j's GRU model can be represented by the following equation:

$$h_{j|t+1} = \phi_j^{GRU}(h_{j|t}, p_j^{'}, t_k^L, \theta_j^{GRU}) \tag{6.8}$$

Where θ_j^{GRU} represents the model parameters of location l_j's GRU. $h_{j|t}$ represents the hidden state of l_j's GRU at time t.

6.3.3 Combining GRU models to learn time-dependent preferences

Each user's GRU model learns the temporal location preferences of the corresponding user and each location's GRU model learns the temporal user preferences for the corresponding location. I define the preference of a user u_i to visit location l_j at time $t+1$ to be a function of the combined hidden state of user and location GRU models at time t. The hidden state outputs of u_i's GRU ($h_{i|t+1}$) and l_j's GRU ($h_{j|t+1}$) are combined to create vector $X_{i,j|t+1}$. The predicted preference ($\hat{r}_{i,j|t+1}$) of user u_i for location l_j at time $t+1$ can be defined by the following equation:

$$\hat{r}_{i,j|t+1} = \hat{Y}(X_{i,j|t+1}) = \sigma(X_{i,j|t+1}, \theta_{ul}^{'}) \; ; X_{i,j|t+1} = [h_{i|t+1}, h_{j|t+1}] \tag{6.9}$$

Where: $h_{i|t}$ and $h_{j|t}$ hidden state outputs of user and location GRUs given by Equation 6.7 and 6.8, respectively. $\theta_{ul}^{'}$ represents the weights and biases of the the sigmoid neuron that takes as input the hidden states of user and location GRUs. Figure 6.4 presents the entire process of obtaining the preference of user u_i for location l_j based on the hidden states of corresponding user and location GRU models.

6.3.4 Training DEEPTREC Model

The locations visited by a user at a specific time represents the implicit temporal preference of the user for those locations. Similarly users who visit a particular location at certain times represent the temporal preferences of that location. So we utilize each user-location check-in along with check-in time to learn the time-dependent location preference of a user and time-dependent user preference of a location.

As described in Section 5.2.2, the overall process to train any neural network model can be divided into 3 steps:

1. For each training sample pair i.e., input X and expected output $Y(X)$, feed the input X to neural network model and obtain the predicted output $\hat{Y}(X)$.

2. Define a cost function that will measure how far is the predicted output from the expected output; for example a quadratic cost function for the neural network model presented in Figure 5.4 is defined as:

$$cost(\theta) = \sum_{\forall X} ||\hat{Y}(X) - Y(X)||^2 \tag{6.10}$$

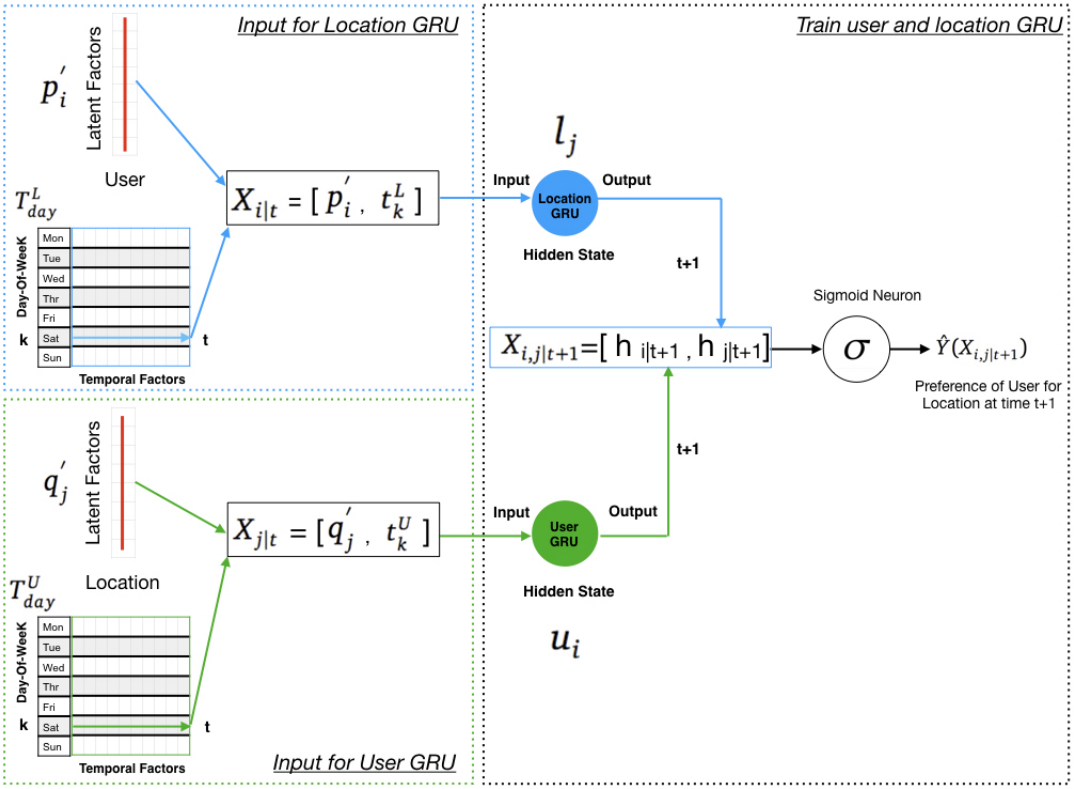


Figure 6.4. Recurrent Neural Network models: a) User GRU Model learns from the latent factor of locations visited by the user; b) Location GRU Model learns from the latent factor of users who visit it. The preference of a user for the location depends on all the previous hidden states of user GRU and location GRU models

Where θ is model parameters or collection of weights and biases associated with all neurons.

3. Utilize an optimization algorithm like gradient descent that minimizes the cost function by finding the optimal set of weights and biases of all neurons in the network.

$$\min_{\theta}\left(cost(\theta)\right) \tag{6.11}$$

Since DEEPTREC model consists of user specific GRUs and location specific GRUs. The goal of training DEEPTREC model is that every user's GRU must learn the temporal locations preferences of the user and every location's GRU must learn the temporal user preferences of the location. Therefore, for user-location

check-in we need to train two GRU models: the user's GRU and the location's GRU. I utilize an alternative subspace descent strategy where I first train all the user GRU models by fixing the parameters of location GRUs and then train all the location GRU models by fixing the parameters of user GRUs. This strategy is similar to the Alternative Least Square (ALS) optimization method Koren et al. [2009].

Given a check-in triplet ($< u_i, l_j, t >$) the procedure to jointly train user and location GRU models can be broadly divided into 6 steps:

1. Initialize user u_i's GRU model and location l_j's GRU model
2. For each check-in triplet ($< u_i, l_j, t >$).
   (a) I combine the latent factors of user u_i i.e, $p_i^{'}$ and location's temporal latent factor t_k^L to create vectors $X_{i|t}$ (as shown in Figure 6.4). I feed the input $X_{i|t}$ to l_j's GRU model to update its hidden state and obtain the hidden state output $h_{j|t+1}$ using Equation 6.7.
   (b) I combine the latent factors of location l_j i.e, $q_j^{'}$ and user's temporal latent factor t_k^U to create vectors $X_{j|t}$. I feed the input X_2 to u_i's GRU model to update its hidden state and obtain the hidden state output $h_{i|t+1}$ using Equation 6.8.
   (c) I combine the hidden state outputs of u_i's GRU ($h_{i|t+1}$) and l_j's GRU ($h_{j|t+1}$) to create vector $X_{i,j|t+1}$. I feed the input $X_{i,j|t+1}$ to a sigmoid neuron to obtain the output $\hat{Y}(X_{i,j|t+1})$ that represents the predicted preference of user u_i for location l_j at time $t+1$. Since we only consider locations visited by user to train user and location GRUs, for all user-location check-ins the expected output $Y(X_{i,j|t+1})$ of the DEEPREC model is 1.
3. Define a cost function that will measure how far is the predicted output from the expected output. I utilize a Cross Entropy Cost function to train the GRU models.
   (a) Cross Entropy Cost

$$cost(\theta) = -\log(\hat{Y}(X_{i,j|t+1}))$$

   Where θ is collection of weights and biases of all neurons in the DEEPTREC model
4. Train u_i's GRU model only

(a) I fix the parameters of l_j's GRU model i.e., θ_j^{GRU} and only update u_i's parameters i.e., θ_i^{GRU}. I utilize Adagrad Duchi et al. [2011] optimization algorithm to minimize the cost function to obtain the optimal set of weights and biases of all neurons part u_i's GRU.

$$\min_{\theta_i^{GRU}} \big(cost(\theta) \big) \tag{6.12}$$

5. Train l_j's GRU model only

(a) I fix the parameters of u_i's GRU model i.e., θ_i^{GRU} and only update l_j's parameters i.e., θ_j^{GRU}. I utilize Adagrad Duchi et al. [2011] optimization algorithm to minimize the cost function to obtain the optimal set of weights and biases of all neurons part l_j's GRU.

$$\min_{\theta_j^{GRU}} \big(cost(\theta) \big) \tag{6.13}$$

6. Train sigmoid neuron in the last layer

(a) Utilize Adagrad Duchi et al. [2011] optimization algorithm to minimize the cost function to obtain the optimal set of weights and biases of sigmoid neuron

$$\min_{\theta'_{ul}} \big(cost(\theta) \big) \tag{6.14}$$

The process of training the joint model is detailed in Algorithm 5

6.4 JOINTDEEPREC Model:

In Chapter 5.3 I described how the time-independent preference of a user u_i for location l_j can be obtained using my `DEEPREC` model that consists of two Feed Forward Neural Networks (FFNN). In Section 6.3, I described `DEEPTREC` model that utilizes GRU models for each user and location to learn about the time-dependent preference of a user for a particular location. I combine `DEEPREC` and `DEEPTREC` models to learns both time-independent and time-dependent preferences of users and location. I refer to the combined model as `JOINTDEEPREC`

We can consider the preference of a user u_i for location l_j to be a combination of the time-independent preferences given by Equation 5.10 and time-dependent

Algorithm 5: Training Procedure of `DEEPTREC`

foreach *triplet* (u_i, l_j, t) **do**

`/* Prepare input for Neural Network models */`

Append latent factor of u_i and temporal factor of l_j to create vector $X_{j|t}$

Append latent factor of l_j and temporal factor of u_i to create vector $X_{i|t}$

`/* Gated Recurrent Unit (GRU) models */`

Input $X_{j|t}$ to u_i's GRU and obtain the hidden state output : $h_{j|t+1}$

Input $X_{i|t}$ to l_j's GRU and obtain the hidden state output : $h_{i|t+1}$

Append hidden state outputs $h_{j|t+1}$ and $h_{i|t+1}$ to create vector $X_{i,j|t+1}$

Input $X_{i,j|t+1}$ to sigmoid neuron and compute cost : $cost(\theta)$

`/* Train user GRU model */`

`/* Compute Gradients to update parameters */`

Gradient w.r.t to user GRU's model parameters : $\frac{\partial cost(\theta)}{\partial \theta_i^{GRU}}$

$\theta_i^{GRU} = \theta_i^{GRU} - \frac{\partial cost(\theta)}{\partial \theta_i^{GRU}}$ `// update user GRU parameters`

`/* Train location GRU model */`

Gradient w.r.t to location GRU's model parameters : $\frac{\partial cost(\theta)}{\partial \theta_j^{GRU}}$

$\theta_j^{GRU} = \theta_j^{GRU} - \frac{\partial cost(\theta)}{\partial \theta_j^{GRU}}$ `// update location GRU parameters`

`/* Train sigmoid neuron */`

Gradient w.r.t to sigmoid neuron's model parameters : $\frac{\partial cost(\theta)}{\partial \theta_{ul}^{'}}$

$\theta_{ul}^{'} = \theta_{ul}^{'} - \frac{\partial cost(\theta)}{\partial \theta_{ul}^{'}}$ `// update sigmoid neuron parameters`

end

preference given by Equation 6.9. We utilize the two equations to obtain the combined predicted preference ($\hat{r}_{i,j|t+1}$) of a user u_i for location l_j at time $t+1$ using the following equation.

$$\hat{r}_{i,j|t+1} = \hat{Y}(X_{i,j|t+1}) + \hat{Y}(X) \tag{6.15}$$

Where $\hat{Y}(X_{i,j|t+1})$ is the predicted preference of user u_i for location l_j at time $t+1$ obtained from `DEEPTREC` model and $\hat{Y}(X)$ is the time-independent preference of user u_i for location l_j obtained using `DEEPREC` model.

6.4.1 Recommend locations using JOINTDEEPREC

Once the Feed Forward Neural Networks (FFNN) part of the `DEEPREC` model are trained and GRU models of all users and locations part of the `DEEPTREC` model are trained, I utilize the FFNN-sigmoid neural network part of the `DEEPREC` and trained user and location GRU models to recommend new locations to user. The overall process to recommend new locations to users is given by the following four steps:

1. For each user in the training data, obtain the list of un-visited locations i.e., new locations where the user has not checked in in the training period.

2. For each user I split the testing data by weeks for example if testing data contain check-ins for 4 weeks then I divide the data into 4 parts, one for each week. I split the entire testing data into weekly data because we want to evaluate the temporal recommendation of `JOINTDEEPREC` model.

3. For each day of the week i.e., Monday to Sunday $k \in \{1,2,3,4,5,6,7\}$

   (a) Calculate the time-aware predicted preference of user for all un-visited locations to visit on k^{th} day of the week

      i. Obtain the time-aware latent factors of all those un-visited locations along with the time-aware latent factor of the user that is being given a recommendation. Also obtain the temporal latent factor of the user for k^{th} day.

      ii. Utilize user's GRU model and each un-visited location's GRU model to compute the predicted preference of user for each one of its un-visited locations to visit on $k+1^{th}$ day. [1]

[1] Please note that if k==7 i.e., Sunday then k+1 refers to k=1 i.e, Monday

(b) Calculate the time-independent predicted preference of user for all un-visited locations

i. Obtain the time-independent latent factors of all those un-visited locations along with the time-independent latent factor of the user that is being given a recommendation.

ii. Utilizing the FFNN-sigmoid neural network of DEEPREC model to compute the predicted preference of the user for each one of its un-visited locations.

(c) Add the time-aware and time-independent predicted preference of user for all un-visited locations for $k+1^{th}$ day.

(d) The top-K un-visited locations with highest predicted scores are recommended to the user.

6.5 Evaluation

In this section, I first describe the dataset used to evaluate the effectiveness of my JOINTDEEPREC model to recommend new locations to users. I also describe the procedure to evaluate JOINTDEEPREC model. I measure the the performance of JOINTDEEPREC model using Normalized Discounted Cumulative Gain (NDCG) metric and compare it with baseline recommendation models.

6.5.1 Dataset

The dataset was constructed by crawling SinaWeibo (China's Twitter), the largest social networking website in China, that provides location-based services such as check-ins. Sina Weibo allow their user's to share their check-ins publicly through their tweets. This dataset was constructed by my colleagues at Microsoft Research Asia. Table 6.1 presents the characteristics of dataset constructed using Sina Weibo open APIs from January 2012 to August 2013. The same dataset was also used to evaluate the DEEPREC model. The properties of dataset are presented in Table 6.1.

6.5.2 Evaluation procedure

In my evaluation, I first clean the dataset by filtering out all those users and locations who have less than 50 check-ins in the entire dataset. I split the entire check-in data by time into two parts. All the check-ins between January 2012

Table 6.1. Sina Weibo Dataset Characteristics

	Sina Weibo
Time period of Check-ins	599 Days
Number of Users	325,447
Number of Locations	79,592
Number of Check-ins	2,348,571

to May 2013 were used to train the model, while check-ins from June 2013 to August 2013 were used to test the model. The training data was used to jointly train the FFNN-sigmoid and FFNN-softmax neural networks that constitute my `DEEPREC` model. Later, the training data was used to train GRU models of all users and locations that are part of my `DEEPTREC` model. For each user, I use the trained FFNN-sigmoid neural network model and GRU models to compute their predicted preference for all their un-visited locations as discussed in Section 6.4.1. The top-K un-visited locations with highest preference is recommend to the user.

6.5.3 Model parameters

In Chapter 5.6.3, I presented the neural network parameters to obtain the best performance for `DEEPREC` model. The neural network parameters of `DEEPREC` model used in evaluating the `JOINTDEEPREC` model are:

FFNN-sigmoid: A neural network with 3 hidden layers. The structure of the neural network to learn the time-independent interaction between user and location factor is: [512, 256, 128, 64, 1]. The first number (512) represents the size of input layer, the next three numbers represent the sizes of three hidden layers. The 5th number represents one sigmoid neuron that takes as input the output of fourth hidden layer (i.e., 64). Since the input is a joint vector of user and location factors, the size of user/location latent factor is a vector of length 256 (512/2).

FFNN-softmax: A neural network with 1 hidden layer. The structure of neural network to learn the geographical distance between latent factors of locations is: [512, 256, 200]. The first number (512) represents the size of input layer, the second number represents the sizes of hidden layer. The 3rd number (200) represents the number of groups into which the euclidean distance between two locations is divided into. A group corresponds to

locations that are within a certain distance. Group 0 corresponds to all pairs that are within [0-2km), group 1 corresponding to all pairs that are withing [2-4km), group 3 corresponding to all pairs that are withing [4-6km) and so on. Since the input to FFNN is a joint vector of user and location factors, the size of user/location latent factor is a vector of length 256 (512/2).

I evaluated the performance of DEEPTREC that learns the time-aware interaction between latent factors of users and location with different size of hidden states. Based on multiple experiments I found the best performance when the size of hidden state vector is 64.

Utilizing the insights obtained by Glorot and Bengio [2010b] in the difficulty to train deep neural networks, I use Xavier method to initialize the model parameters i.e., weights and biases of all neuron in my DEEPREC and DEEPTREC models. I use tanh activation function for all neurons in the DEEPREC and DEEPTREC models. For all my experiments, I fix the learning rate to 0.01 i.e., all the model parameters of FFNN-sigmoid and FFNN-softmax are updated in steps of 10^{-2}. To improve the learning process, I employ the dropout technique Srivastava et al. [2014] at the output layer with the dropout rate set to 0.5. I train the entire model for one epoch and observe the results to surpass the state of the art location recommendation algorithms. I utilize Adagrad Duchi et al. [2011] optimization method to find the optimal values for model parameters.

6.5.4 Performance metrics

I evaluated the performance of JOINTDEEPREC model and all baseline algorithms using a time-aware Normalized Discounted Cumulative Gain (NDCG) metric that takes into account recommendations that vary with time (or day of the week). NDCG jointly measures not only whether a user has visited a recommended location but also takes into account the ranking of recommended location in entire recommendation list.

For the evaluation, I performed experiments with different temporal information like recommending locations for each *hour of the day* and for each *day of the week* and found that the SinaWeibo dataset does not contain enough number of check-ins for each day to model the location visiting patterns for each hour.

Discounted cumulative gain (DCG) is a metric that captures the quality of ranking. The metric takes into account the order in which a relevant location appears in the recommend set. Let $S_u^t(k)$ be the set of top K locations recommended to a user u on t^{th} day of the week and V_u^t be the set of new locations

visited by user in the testing data on t^{th} day of the week. The DCG of set of locations recommended to user u on t^{th} day of the week is given by the following equation:

$$\text{DCG}_u^t@K = \sum_{i=1}^{K} \frac{2^{r_i} - 1}{log_2(i+1)},\ r_i = \begin{cases} 1 \text{ if } S_u^t(i) \in V_u^t \\ \\ 0 \text{ Otherwise} \end{cases} \tag{6.16}$$

The DCG of the testing set is called Ideal discounted cumulative gain (IDCG) i.e., what would be the ideal DCG when all the locations in testing set are part of the recommended set in the exact same order. The IDCG of set of un-visited locations part of the testing data of user u for t^{th} day of the week is given using the following equation:

$$\text{IDCG}_u^t = \sum_{i=1}^{|V_u^t|} \frac{1}{log_2(i+1)} \tag{6.17}$$

The Normalized Discounted Cumulative Gain (NDCG) of the set of recommendations provided to all users is computed as follows:

$$\text{NDCG@K} = \frac{1}{N} \sum_{u=1}^{N} \frac{1}{7} \sum_{t=1}^{7} \frac{\text{DCG}_u^t@K}{\text{IDCG}_u^t} \tag{6.18}$$

6.5.5 Baseline Algorithms

I compared the performance of `JOINTDEEPREC` model with the following standard filtering based recommendation models:

- timeSVD++ Koren [2008] is one of the most popular time-aware factorization based recommendation model that has also shown good performance on Netflix dataset. In my evaluation, I performed a grid search of factor size over *Factors* $\in \{16, 32, 64\}$ to select the best parameters.

- Regularized Content-Aware Tensor Factorization (RCTF) Lian et al. [2016] is another time-aware factorization based model that also augments the user's and location's latent factors to incorporate the spatial constraints. In my evaluation, I perform a grid search of factor size over *Factors* $\in \{16, 32, 64, 128, 256\}$ to select the best parameters.

I do not compare the performance of `JOINTDEEPREC` with baseline algorithms (LFBCA, UCF, LCF, LIBFM, SVD++, GeoMF) compared in previous chapters because they do not provide time-dependent location recommendations.

6.5.6 Performance Comparison

Figures 6.5, 6.6, present the Normalized Discounted Cumulative Gain (NDCG) for different number of recommended locations (i.e., K) and for different day of the week for different baseline algorithms. In general the NDCG increases with K because of an increase in the possibility that top-K locations could be part of the set of new locations visited by user. In other words, the Discounted Cumulative Gain (DCG) increases with K while the Ideal Discounted Cumulative Gain (IDCG) is remains constant.

Figure 6.6 presents the recommendation performance for different days of week for RCTF model when every user and location preference is represented by a latent factor of size 16. We observe that the model is able to model the temporal characteristics of users on Monday's, Friday's and Saturday's better than compared to other days because people are most likely to visit the places near office on Monday's and visit frequently visited recreational places on Friday evening and Saturday's. In addition, the model is able to capture temporal characteristics of users on Wednesday's better than Thursday's and Sunday's because people are very likely to be at work mid week and visit nearby places for lunch or dinner. People in Beijing often have lunch and dinner at work or nearby office as they tend to work till late in the evening.

Figure 6.5 presents the recommendation performance for different days of week measured in terms of NDCG for timeSVD++ model. We observe that it performs significantly bad than compared to the RCTF model due to its inability to capture spatial constraints between locations. As discussed in detail in Chapter 3.3.2, humans exhibit a lot of inertia and are more likely to visit geographically close locations. timeSVD++ does not incorporate this information in its model while RCTF incorporates it. Comparing the performance of RCTF and timeSVD++ highlights the importance of incorporating spatial constraints in a recommendation model along with temporal dynamics.

Comparing the performance of different models, we observe that for our Sina Weibo dataset, the performance of RCTF model is better than SVD++.

6.5.7 Performance of JOINTDEEPREC model

In my `JOINTDEEPREC` model the size of latent factor that is to learned using a neural network for each user and location is 256 (refer to Section 6.5.3). Further, the cell state size or memory size of each user and location GRU is 64. (refer to Section 6.5.3). Figure 6.7 presents the comparative performance of my neural network based recommendation model i.e., `JOINTDEEPREC` with the baseline

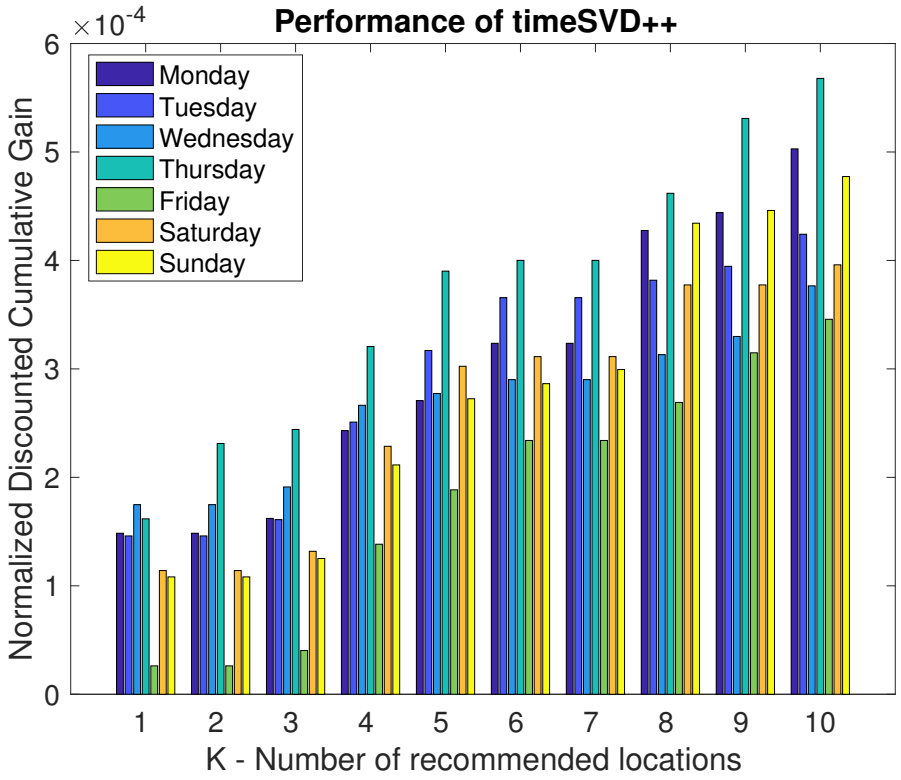


Figure 6.5. Performance of timeSVD++ model for all days of the week starting from Monday to Sunday. The number of user and location factors is 64

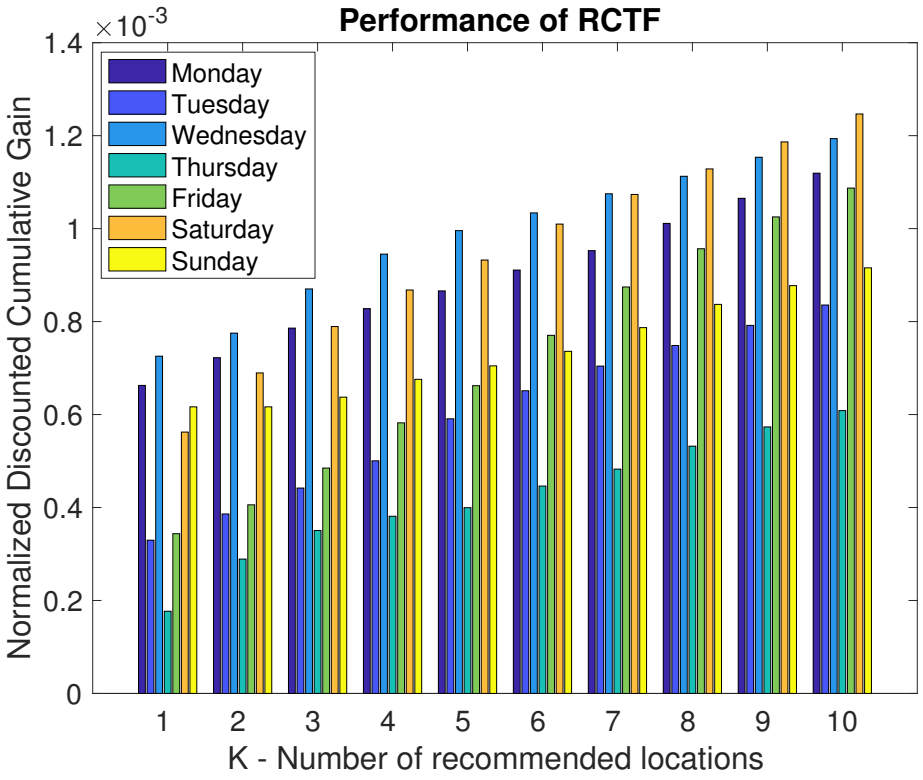


Figure 6.6. Performance of RCTF model for all days of the week starting from Monday to Sunday.The number of user and location factors is 16

algorithms for different days of the week. We can clearly observe that my neural network based model outperforms baseline models with a significant margin. In order to show the relative performance I present the values of NDCG metric in log scale. When 10 locations are recommended to each users JOINTDEEPREC model performs 10 times better than compared to RCTF model which is the best among the baselines considered.

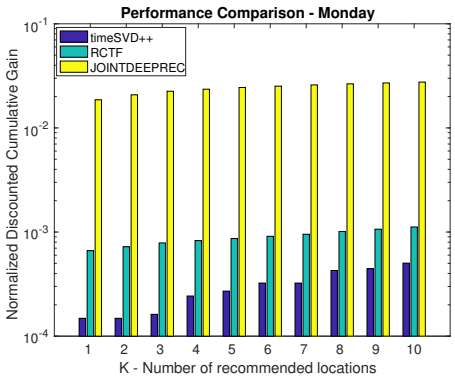
The performance of my JOINTDEEPREC model is better that other factorization based model because of the DEEPREC neural network model's ability to learn time-independent preferences of users and locations by capturing the interaction between the latent factors of all users and locations. In addition, DEEPREC also capture the spatial preferences of user by constraining the latent factors of geographically close locations. Further, DEEPTREC model is able to learn the time-dependent preferences of users and locations with the help of a GRU based Recurrent Neural Networks (RNN). The GRU models enable DEEPTREC to learn daily preferences of users and locations because it models their temporal preferences using a day based latent factor matrix (described in Section 6.3). A small neural network with just 3 hidden layers is able to learn the time-independent preferences of users and 64 dimensional cell state of GRU is able to learn the time-dependent preferences of users and locations. How is it able to learn better has been explained in detail in Section 6.3.4 .

These experiments clearly highlight the ability of neural networks to model the time-independent and time-dependent preferences of users and locations.

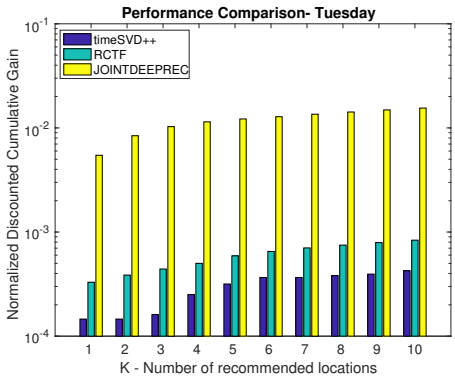
6.6 Conclusion and Limitations of JOINTDEEPREC

In this chapter, I first presented the basics of a Recurrent Neural Network (RRN) and how due to its memory it is able to remember recurrent patterns in a given data. I described how we can utilize a specific RNN called GRU to learn the time-dependent location preferences of users and time-dependent user preferences of locations. Later, I described the DEEPTREC model that learns the temporal latent factors of users and locations by combining GRU models of users and locations. Finally, I described how we can learn the time-independent and time-dependent preference of users and locations by combining the DEEPREC model presented in Chapter 6 with DEEPTREC model to form a JOINTDEEPREC model.

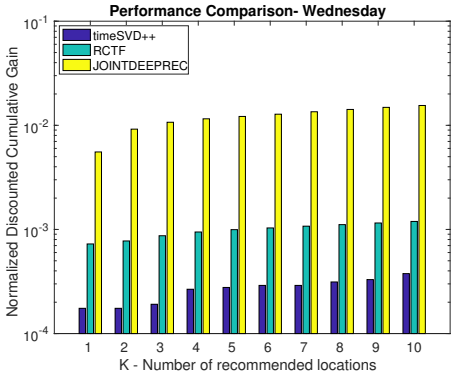
In Section 6.2, I presented basic structure of an RNN and how a GRU-a variant of RNN remembers historical information with the help of a cell state and interacts with current and previous information with the help of reset and forget gates. In Section 6.3, I described the DEEPTREC model where every user's



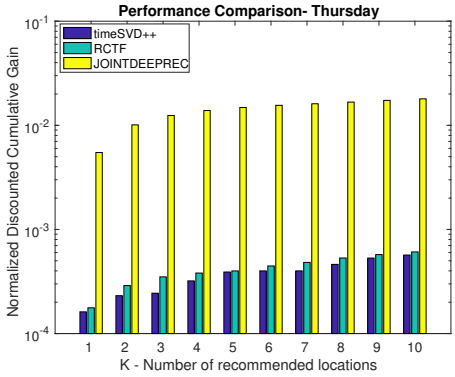
(a) Monday



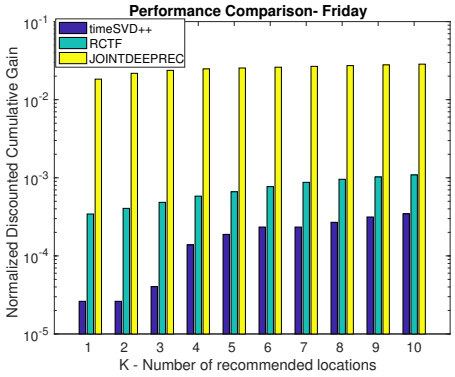
(b) Tuesday



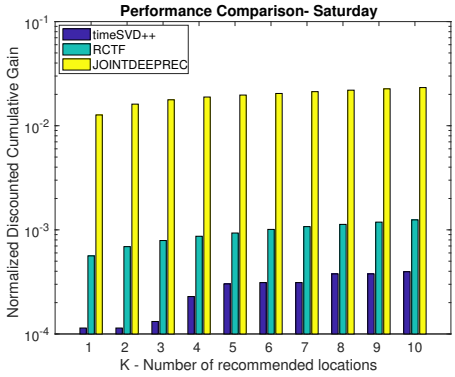
(c) Wednesday



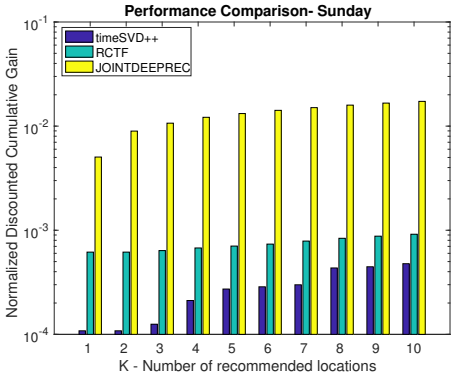
(d) Thursday



(e) Friday



(f) Saturday



(g) Sunday

Figure 6.7. Comparison between the performance of `JOINTDEEPREC` , RCTF and SVD++ for their best parameters for different days of the week.

and location's temporal preferences is captured by their own specific GRUs and I jointly train them using an alternative subspace descent strategy. I utilize a Cross Entropy Cost function to train user and location GRUs. In Section 6.4, I presented how I combine DEEPTREC model with DEEPREC model that learn time-independent preferences of users and locations. I presented the comparative performance of the combined model i.e., JOINTDEEPREC with two state-of-the-art factorization based models and show that it outperforms them due to neural networks that are part of DEEPREC and DEEPTREC to identify better time-independent and time-dependent latent factors to capture the preferences of users and locations. The key findings of this chapter are:

- I present how the time-dependent preferences of users and locations represented as temporal latent factors can be learned using a combination of two Gated Recurrent Unit (GRU) based Recurrent Neural Network (RNN) model.
- To the best of my knowledge, JOINTDEEPREC is the first model to learn the temporal location preferences of users and temporal user preference of locations in the context of a location recommendation.
- JOINTDEEPREC model performs 10 times better than state-of-the-art Regularized Content-Aware Tensor Factorization(RCTF) recommendation model due to the inherent ability of a GRU to remember the regularly mobility patterns of users and regularity of people who visit a location.

Further, with the advent of GPUs, the ability to train and update neural network based models has increased by many orders of magnitude compared to CPUs concerning computation time. This ability to retrain and evaluate models using GPUs will significantly allow JOINTDEEPREC to learn from a new stream of user check-ins continuously. Since each user and location has its own dedicated GRU model, the ability to provide location recommendations using JOINTDEEPREC is scalable to significantly a large number of users and locations.

6.6.1 Limitations

Given that I utilize a GRU model for each user and location, the computational complexity is high as it requires many days to train all GRUs. There is a need to find better procedure to train such complex models that combine different neural networks. Multiple research works are already underway to improve the training procedures of neural networks, as finding the right weights and biases

of all hundreds of neurons is important for a neural network model to learn correctly and predict accurate information.

Since, each GRU model is also provided temporal information i.e., check-in's day of visit, there is a need to have enough data available for each user for each day of the week to be able to learn meaningful information about the user's preferences. Therefore, datasets with much higher granularity of data is need to accurately train and test the effectiveness of RNN based model that aim to learn the temporal preferences of users and locations.

6.6.2 Remarks

The work presented in this Chapter was done in collaboration with Dr. Defu Lian (Big Data Research Center, UESTC, Chengdu, China), Dr. Xing Xie (Microsoft Research Asia, Beijing, China) and Danyang Liu (Ph.D. student of Dr. Xing Xie).

Chapter 7

Conclusions

7.1 Summary and Contributions

A Recommendation system helps users to discover products they may like and also provide a scalable way to offer personalized products and services for a broad set of users in multiple scenarios. Recommendations not only help people to find what they are looking for but also enable them to narrow down their choices. The spread and extensive usage of mobile applications has led to the rise of Location-based social networks (LBSNs) services like Foursquare, Jeipang, Yelp, etc. The novelty-seeking behavior of people has led to the growth of location recommender services that aim to provide new places of interest to people based on their interests and habits. In particular, providing relevant location recommendations to users of an LBSN is essential to drive customer engagement with the mobile application and is also an important research topic. This dissertation focused on the research problem of providing time-aware location recommendations or Point of Interests (POIs) to users of a Location Based Social Network (LBSN). In Chapter 2, I presented the state-of-the-art approaches in the literature to provide time independent and time-aware locations recommendations. I also highlighted research works on deep neural networks based location recommendations. The different research challenges addressed in this dissertation towards achieving the goal of designing a novel time-aware location recommendation are:

R1 How to capture and combine geographical, temporal preferences of users along with influence of their social peers to provide better location recommendations them?

R2 How to build a scalable model than can continuously learn the locations

preferences of users from their check-in history and recommend new locations to them? In addition, How to capture the geographical constraints of neighboring locations in a location recommendation model?

R3 Location preferences of users change with time. How to design a model that can learn the time-varying location preferences of users?

The first research challenge (R1) was addressed in Chapters 3 and 4. In Chapter 3, I presented the experimental analysis done on two LBSN datasets to test 5 hypothesis I have formulated about mobility behavior of users. Based on these tests, I designed three key features: a) Temporal feature that captures the influence of time in user visits; b) Distance feature that captures the geographical influence of user visits; and c) Friendship feature that captures the influence of the social friends on user visits, to capture the spatial and temporal mobility of people along with influence of social peers. These features were used to build my feature based location recommendation model REGULA .

In Chapter 4, I presented my first research work on recommending new cell regions to people based on their Call Detail Records (CDR). I showed how we could utilize CDRs to construct a new CDR-based LBSN and conducted analytical tests to demonstrate that CDR users also exhibit regular mobility behavior. Finally, I presented a variant of REGULA used to recommend new cell regions to users based on their CDR. REGULA can potentially be used by cell phone operators to recommend new places that are relevant to their customers by tying up with multiple businesses. It can also be used to identify cell regions that could potentially face increased network workload and deploy infrastructure resources accordingly so that their customers don't face inconvenience like call drops.

Chapter 5 presents how I addressed the second research challenge (R2). In Chapter 5, I presented DEEPREC , a deep neural network based recommendation model that provides time-independent location recommendations to users. DEEPREC learns the location preferences of users by understanding their complex visiting patterns at different locations. I presented why and how DEEPREC captures the user and location preferences better than existing state-of-the-art factorization based techniques. DEEPREC utilized two Feed Forward Neural Networks (FFNN-sigmoid and FFNN-softmax) to learn the preferences of users and locations along with capturing the geographical constraints between neighborhood locations.

Finally, Chapter 6 presents how I addressed the last research challenge (R3). In Chapter 6, I introduced DEEPTREC a recurrent neural network based model that uses multiple Gated Recurrent Units (GRU) to learn the time-varying locations preferences of users. I also presented how I combine DEEPREC and DEEPTREC to

form a JOINTDEEPREC that learns time-independent and time-varying preferences of users to provide time-aware location recommendations.

All the deep learning based models were evaluated on one of the largest check-in dataset constructed by crawling Sina Weibo (China's Twitter), the largest social networking website in China. My DEEPREC and JOINTDEEPREC models have outperformed the state-of-the-art by a factor of 10. My recommendation models can be used in application domains that consist of users interacting with multiple locations, e.g., OpenTable, Yelp, Foursquare, etc.

7.2 Directions for future research

This dissertation focused on providing time-aware location recommendations to users. I will now present some future research directions that can be addressed to extend the work presented in this dissertation.

7.2.1 Utilize user generated content

Multiple mobile applications that provide information about interesting places to people like Foursquare, OpenTable, etc., also allow their users to rate and comment on the place they visit. These rating and comments represent the explicit feedback of users. Further, the users also provide some meta-data like demographics, preferred places, etc. This explicit feedback of users and metadata information can be utilized to improve the process of learning the user preferences and provide better location recommendations to them. There exist some deep neural networks that can the learn the preferences of users based on the comments and pictures shared by them.

7.2.2 Capture social influence

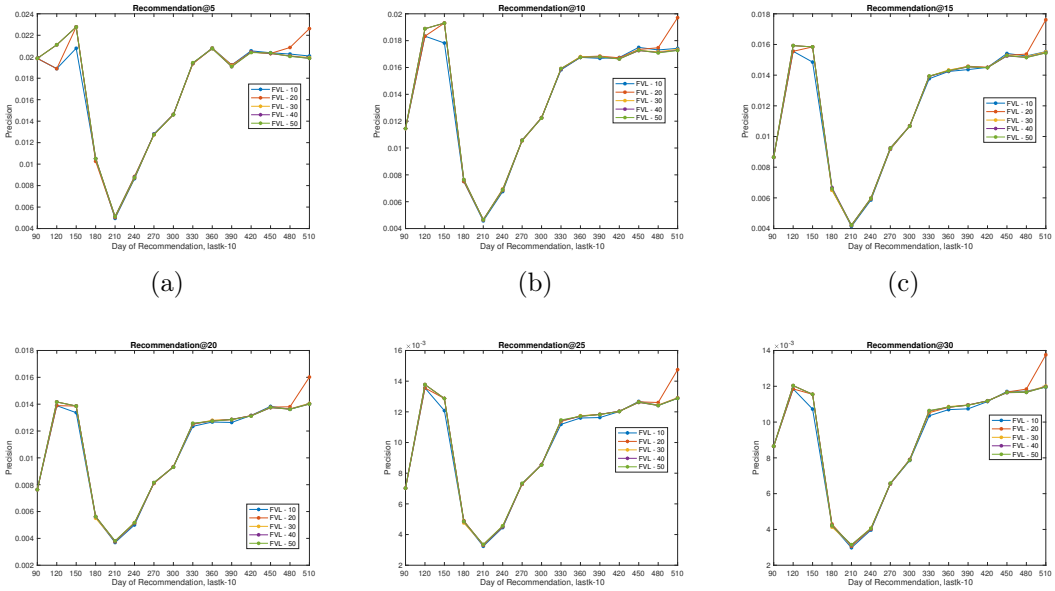
In this dissertation, due to lack of information, the deep learning based recommendation models DEEPREC and JOINTDEEPREC do not incorporate the social ties among users in learning the preferences of users. Humans are social animals, and their social peers do influence the places they visit as highlighted in Chapter 3. The deep learning based models presented in this dissertation can be extended to learn the factors that capture the influence of a user's friends on their locations preferences.

7.2.3 Differential importance to Locations visited both in Time and Space

The new locations visited by a person depends on multiple spatial factors like where are they currently, which are the different places they have been to recently, which are the places they regularly go. It also depends on multiple temporal factors like what is the current time, what are the times during which they usually visit a new place etc. These numerous spatial and temporal factors have different importance and are unique to each person. Recently, a new class of neural networks called attention based neural networks enable them to assign differential importance to temporal factors. These networks are found to memorize things better that simple recurrent neural networks like LSTM or GRU.

Chapter 8

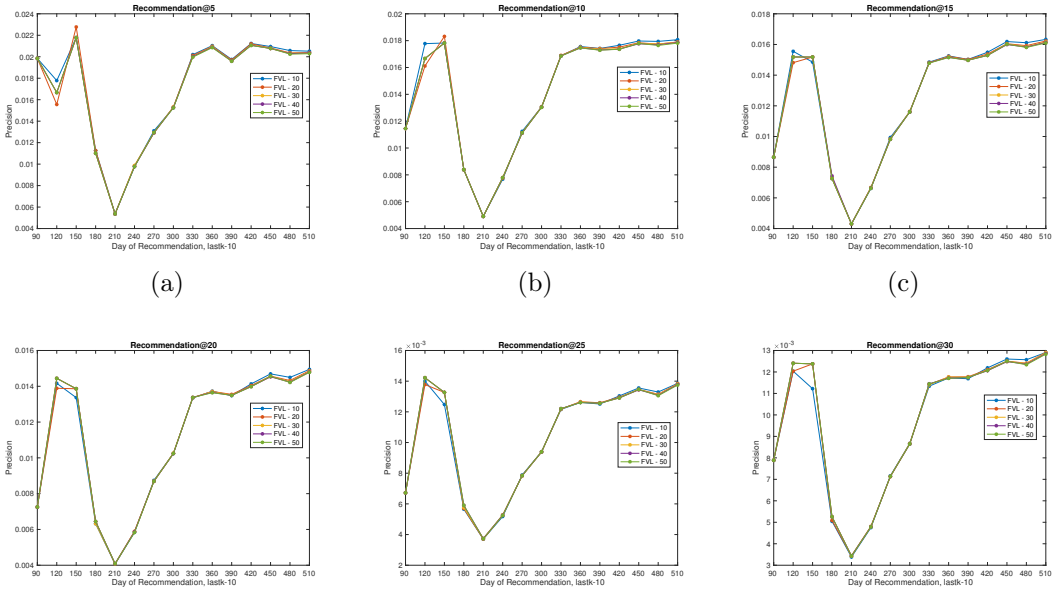
Appendix



(a) (b) (c)

(d) (e) (f)

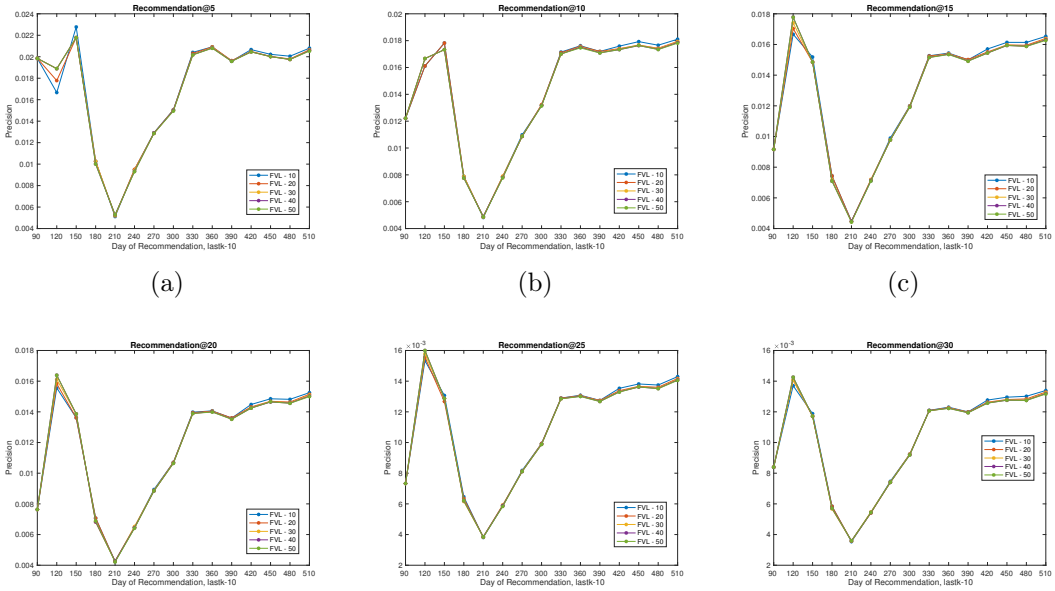
Figure 8.1. Performance ($p@K$) of REGULA on Gowalla for different FVLs, α=100, $lastk$=10, Box=1km.



(a) (b) (c)

(d) (e) (f)

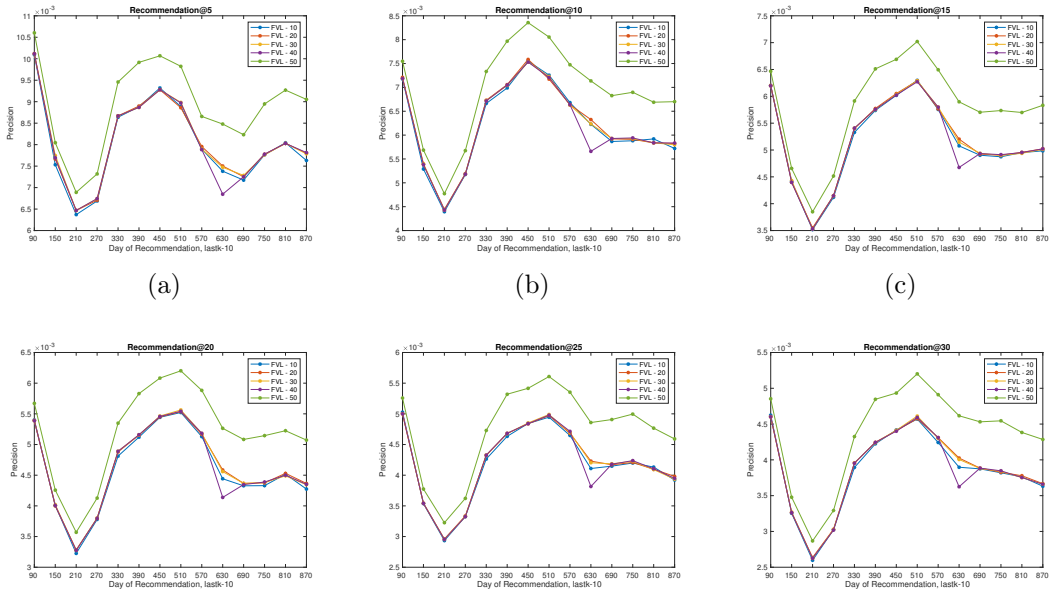
Figure 8.2. Performance ($p@K$) of REGULA on Gowalla for different FVLs, α=100, $lastk$=10, Box=2.5km.



(a) (b) (c)

(d) (e) (f)

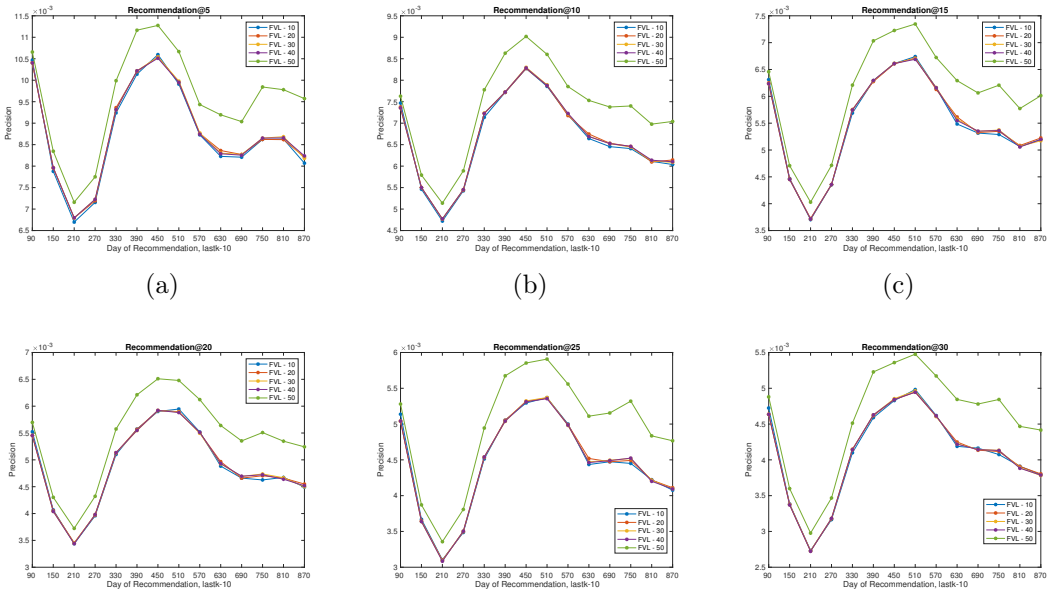
Figure 8.3. Performance ($p@K$) of REGULA on Gowalla for different FVLs, α=100, $lastk$=10, Box=5km.



(a) (b) (c)

(d) (e) (f)

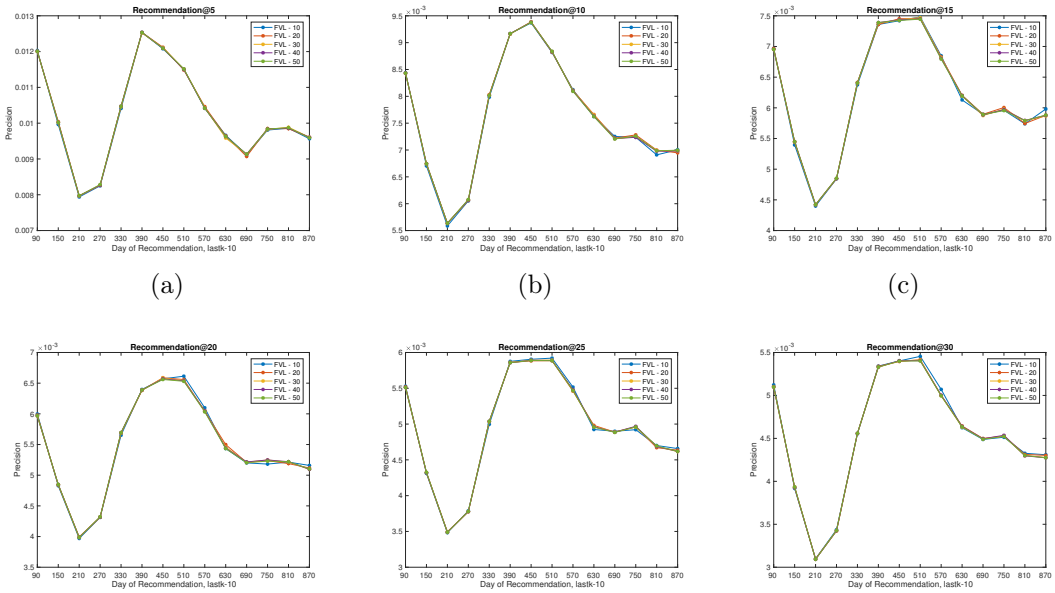
Figure 8.4. Performance ($p@K$) of REGULA on Brightkite for different FVLs, α=100, $lastk$=10, Box=1km.



(a) (b) (c)

(d) (e) (f)

Figure 8.5. Performance ($p@K$) of REGULA on Brightkite for different FVLs, α=100, $lastk$=10, Box=2.5km.



(a) (b) (c)

(d) (e) (f)

Figure 8.6. Performance ($p@K$) of REGULA on Brightkite for different FVLs, α=100, $lastk$=10, Box=5km.

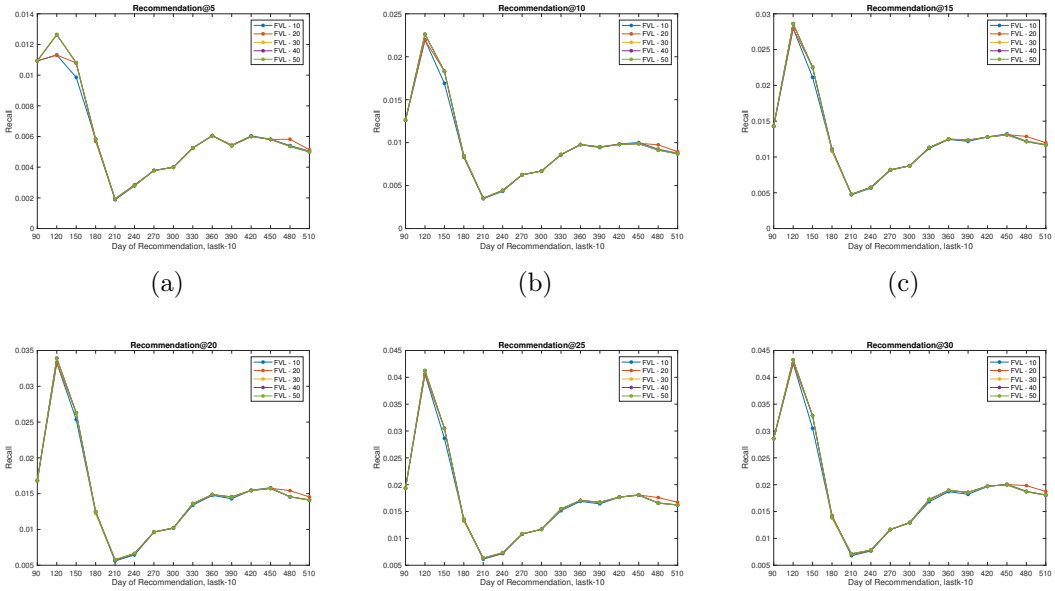


Figure 8.7. Performance ($r@K$) of REGULA on Gowalla for different FVLs, α=100, $lastk$=10, Box=1km.

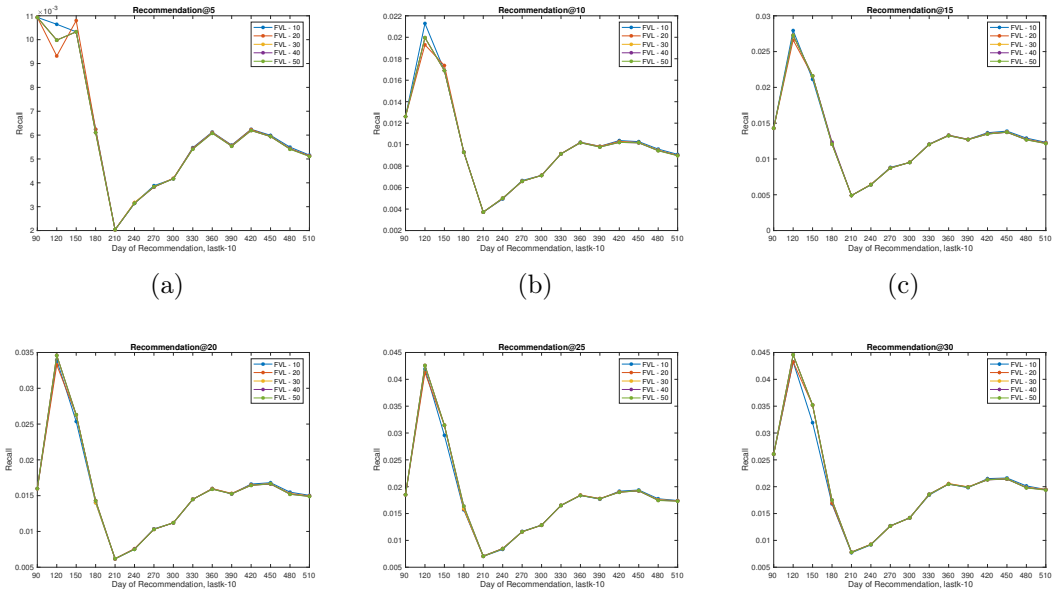
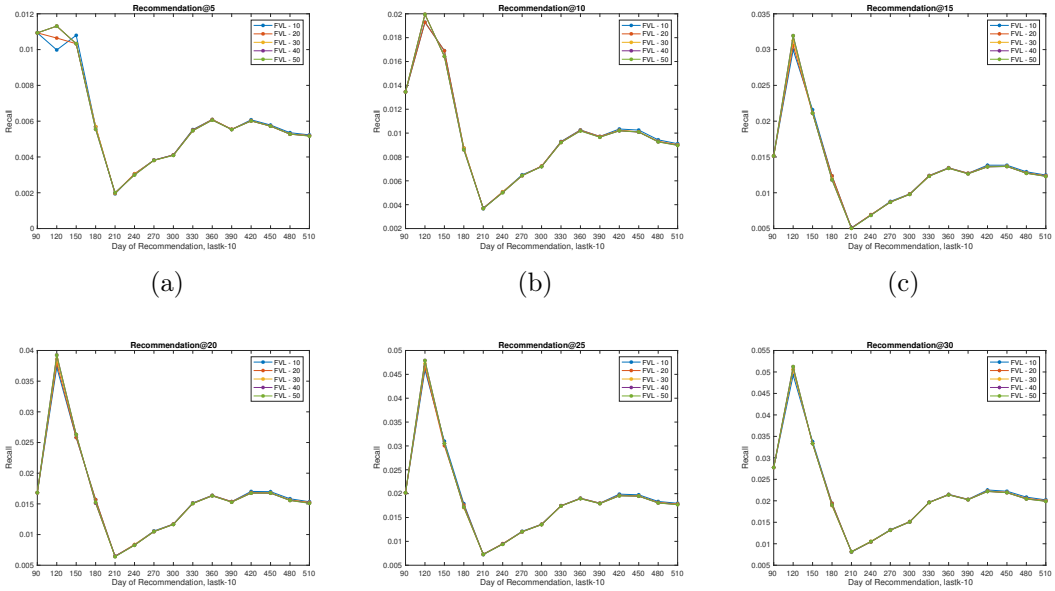


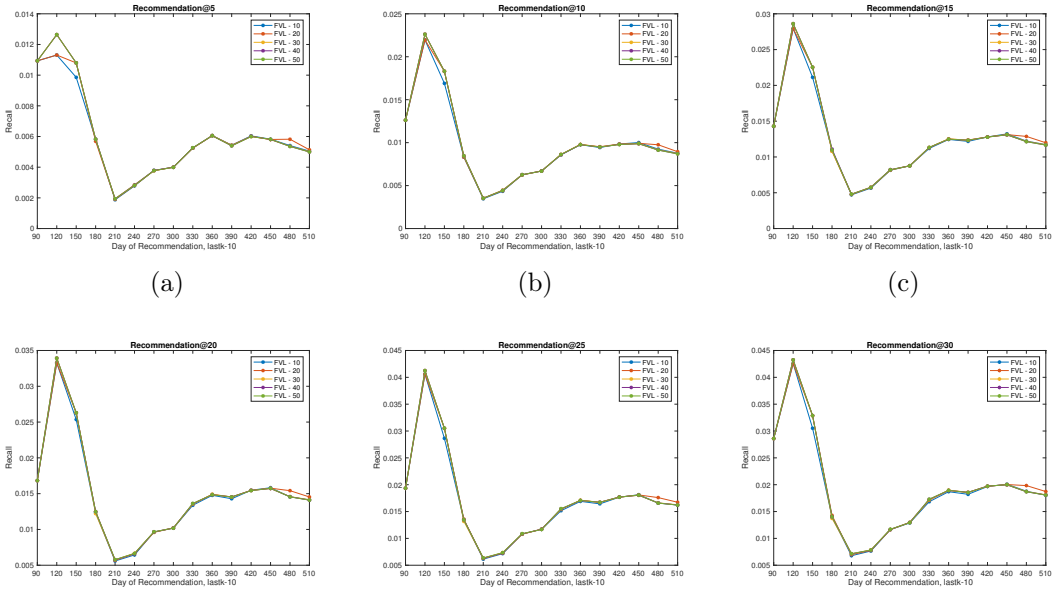
Figure 8.8. Performance ($r@K$) of REGULA on Gowalla for different FVLs, α=100, $lastk$=10, Box=2.5km.



(a) (b) (c)

(d) (e) (f)

Figure 8.9. Performance ($r@K$) of REGULA on Gowalla for different FVLs, α=100, $lastk$=10, Box=5km.



(a) (b) (c)

(d) (e) (f)

Figure 8.10. Performance ($r@K$) of REGULA on Brightkite for different FVLs, α=100, $lastk$=10, Box=1km.

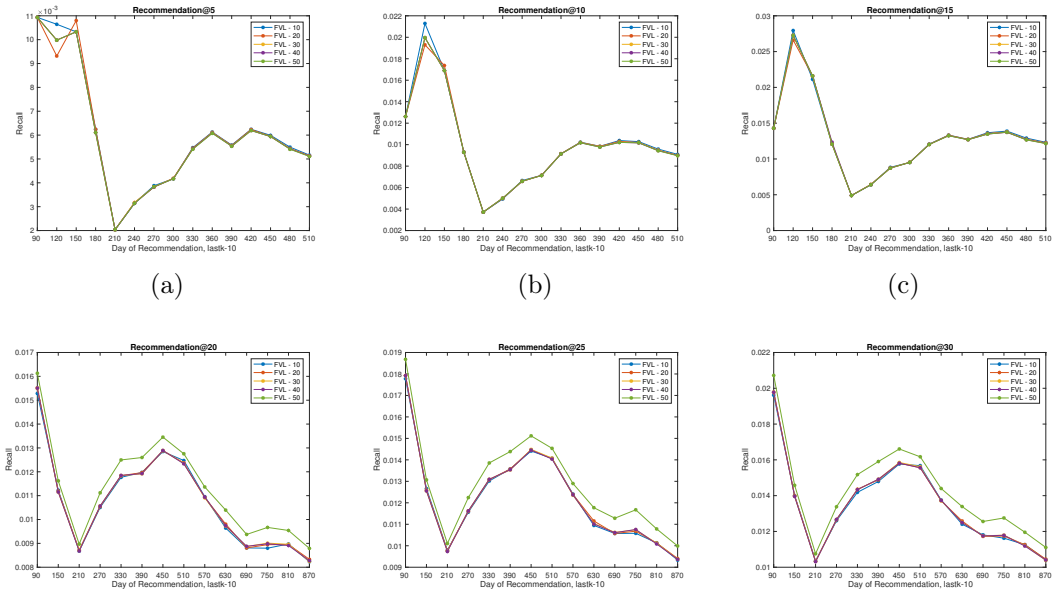
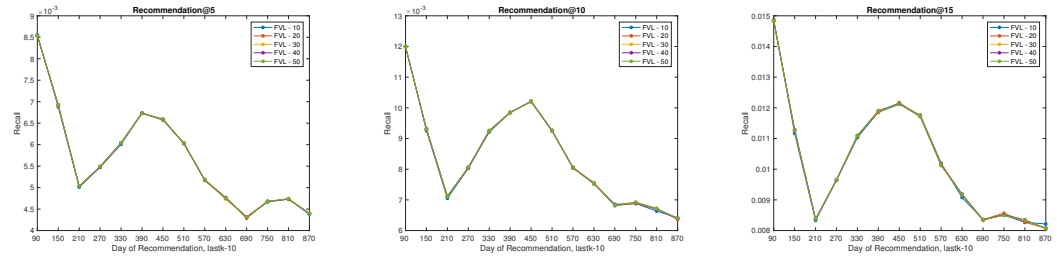
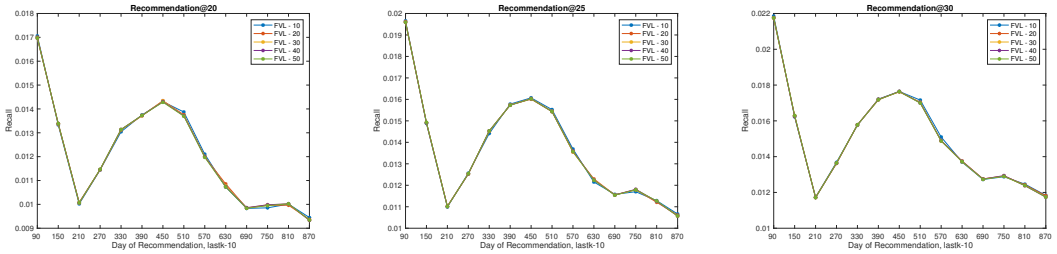


Figure 8.11. Performance ($r@K$) of REGULA on Brightkite for different FVLs, α=100, $lastk$=10, Box=2.5km.

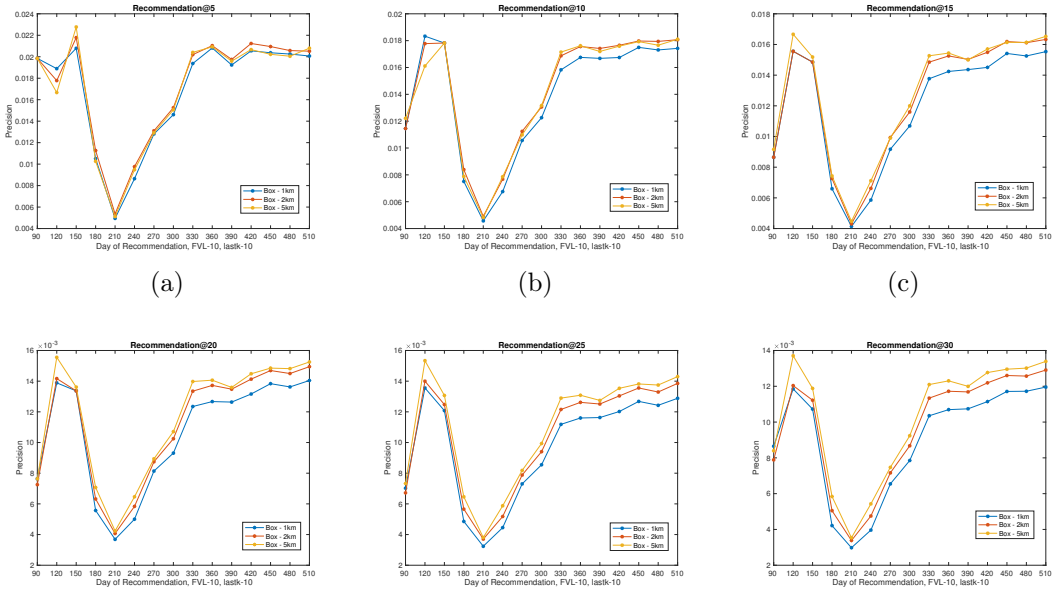


(a) (b) (c)



(d) (e) (f)

Figure 8.12. Performance ($r@K$) of REGULA on Brightkite for different FVLs, α=100, $lastk$=10, Box=5km.



(a) (b) (c)

(d) (e) (f)

Figure 8.13. Performance ($p@K$) of REGULA on Gowalla for different size of Bounding Box, $\alpha = 100$, $FVL = 10$, $lastk$=10.

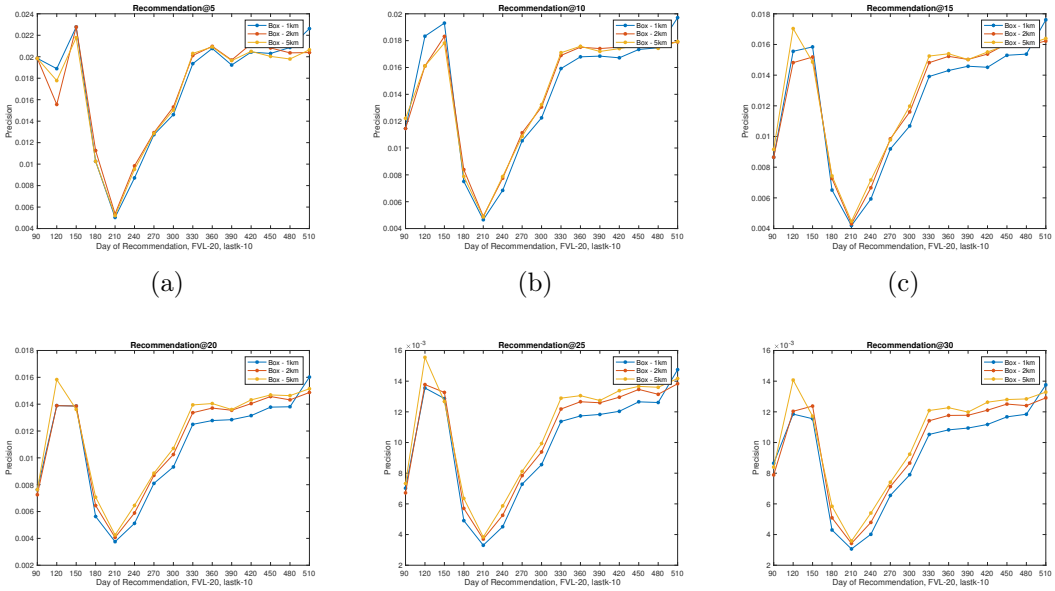


Figure 8.14. Performance ($p@K$) of REGULA on Gowalla for different size of Bounding Box, $\alpha = 100$, $FVL = 20$, $lastk$=10.

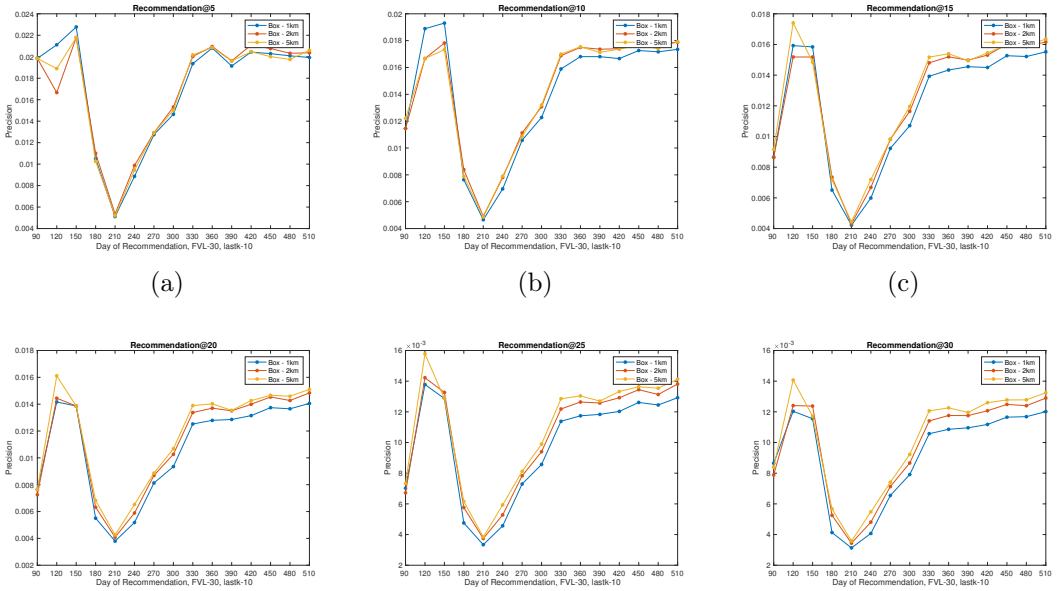


Figure 8.15. Performance ($p@K$) of REGULA on Gowalla for different size of Bounding Box, $\alpha = 100$, $FVL = 30$, $lastk$=10.

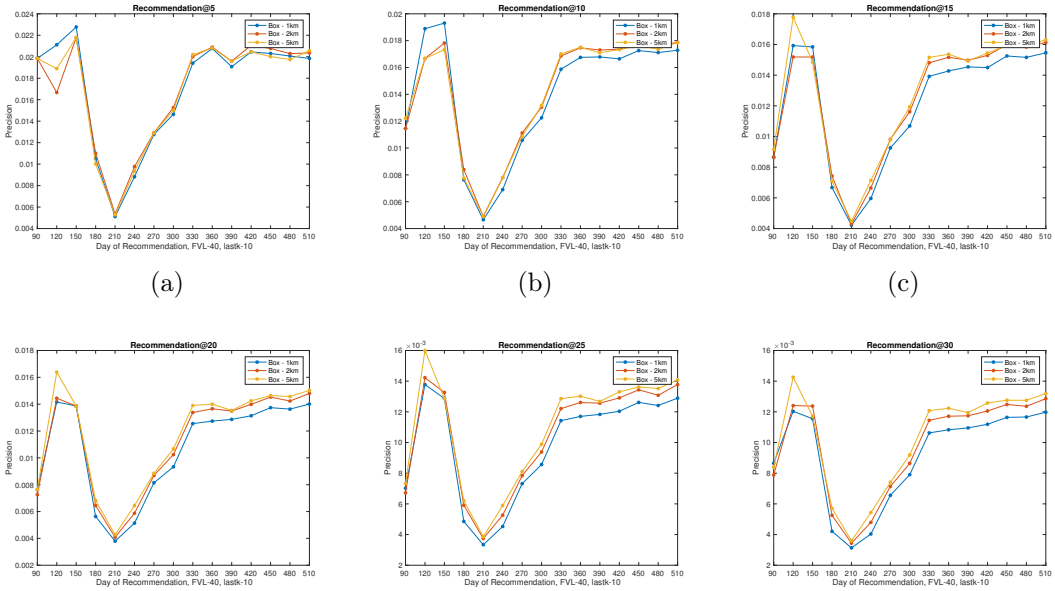


Figure 8.16. Performance ($p@K$) of REGULA on Gowalla for different size of Bounding Box, $\alpha = 100$, $FVL = 40$, $lastk$=10.

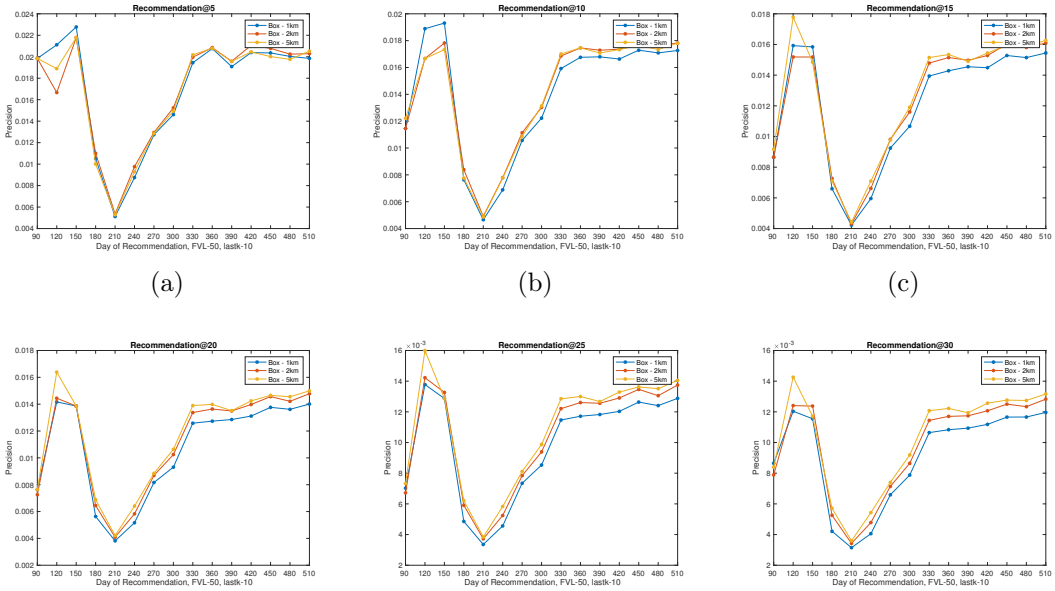


Figure 8.17. Performance ($p@K$) of REGULA on Gowalla for different size of Bounding Box, $\alpha = 100$, $FVL = 50$, $lastk$=10.

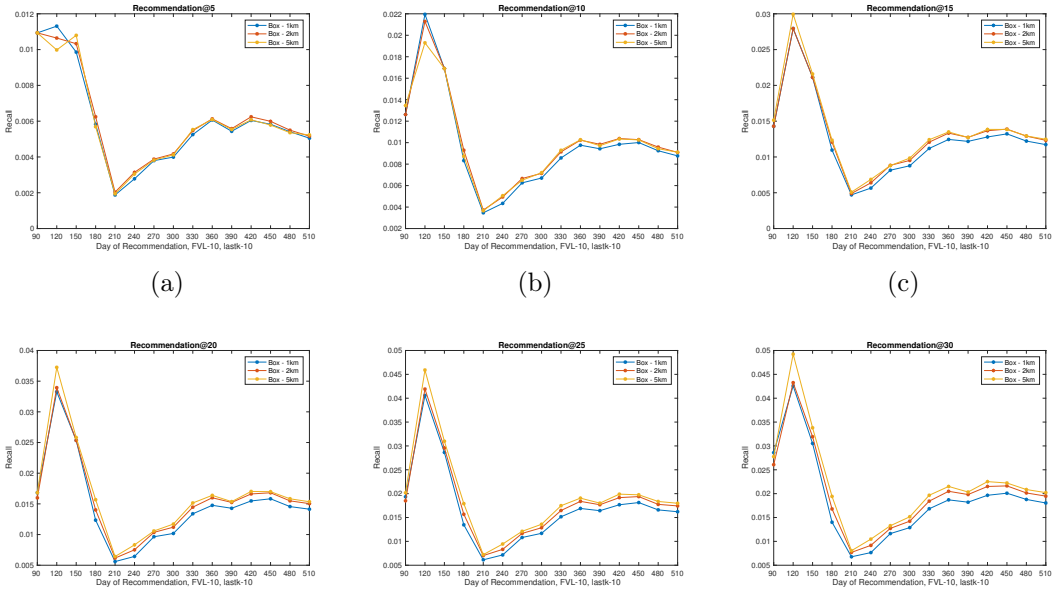
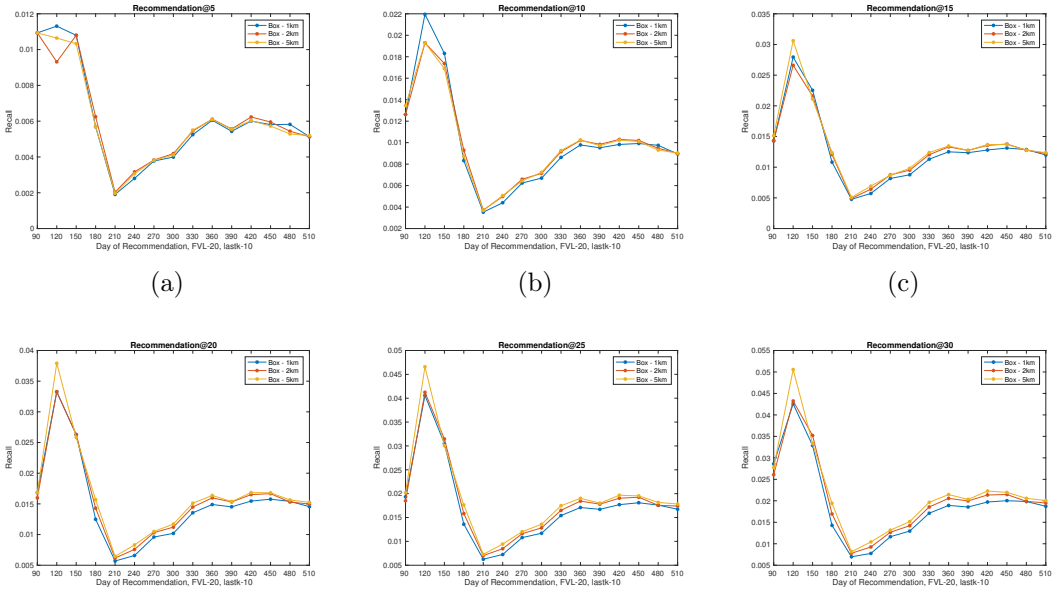


Figure 8.18. Performance ($r@K$) of REGULA on Gowalla for different size of Bounding Box, $\alpha = 100$, $FVL = 10$, $lastk$=10.



(a) (b) (c)

(d) (e) (f)

Figure 8.19. Performance ($r@K$) of REGULA on Gowalla for different size of Bounding Box, $\alpha = 100$, $FVL = 20$, $lastk$=10.

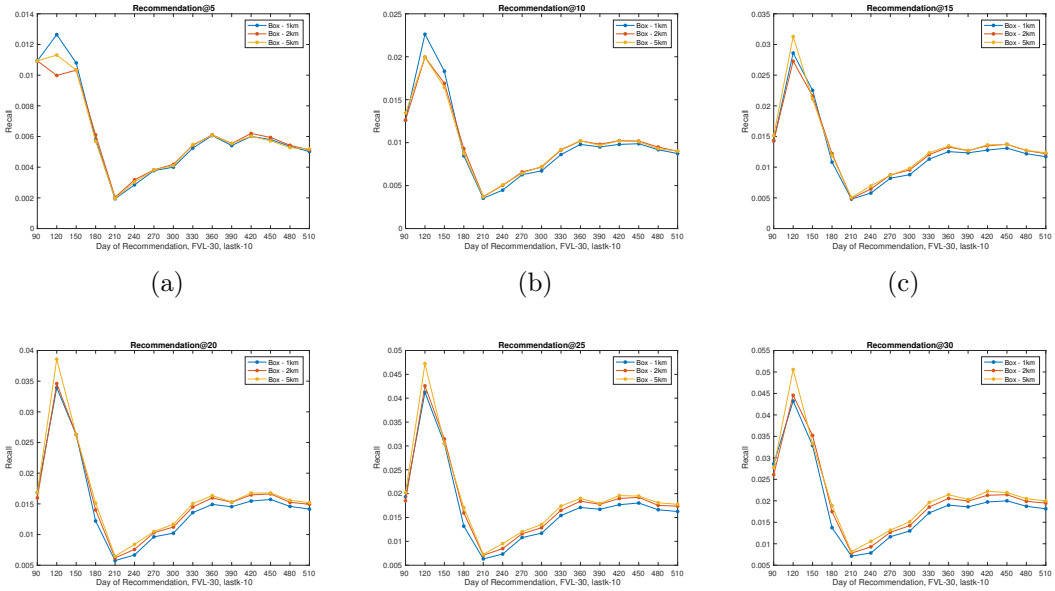
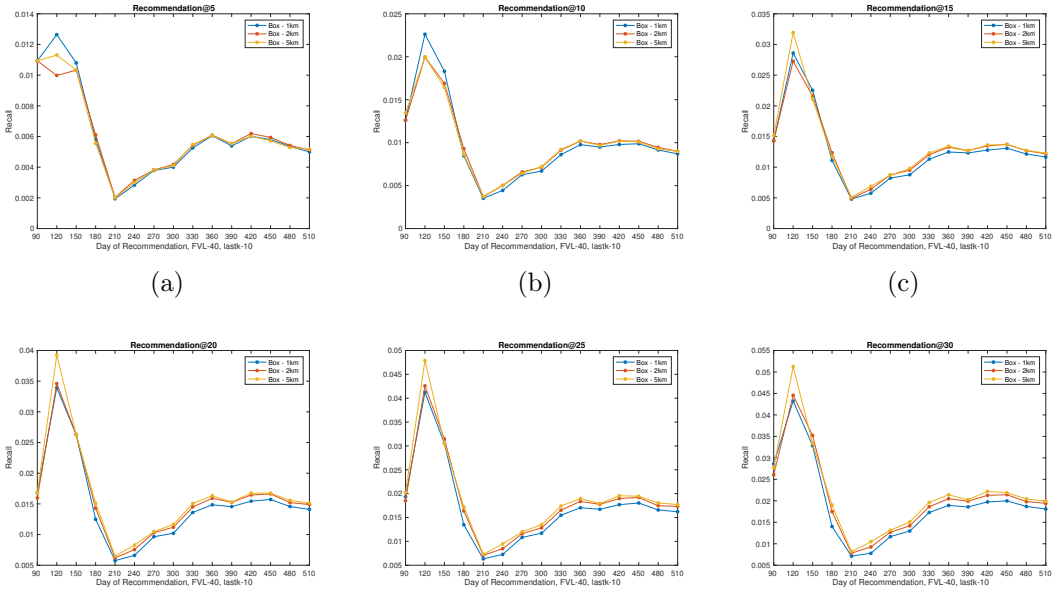


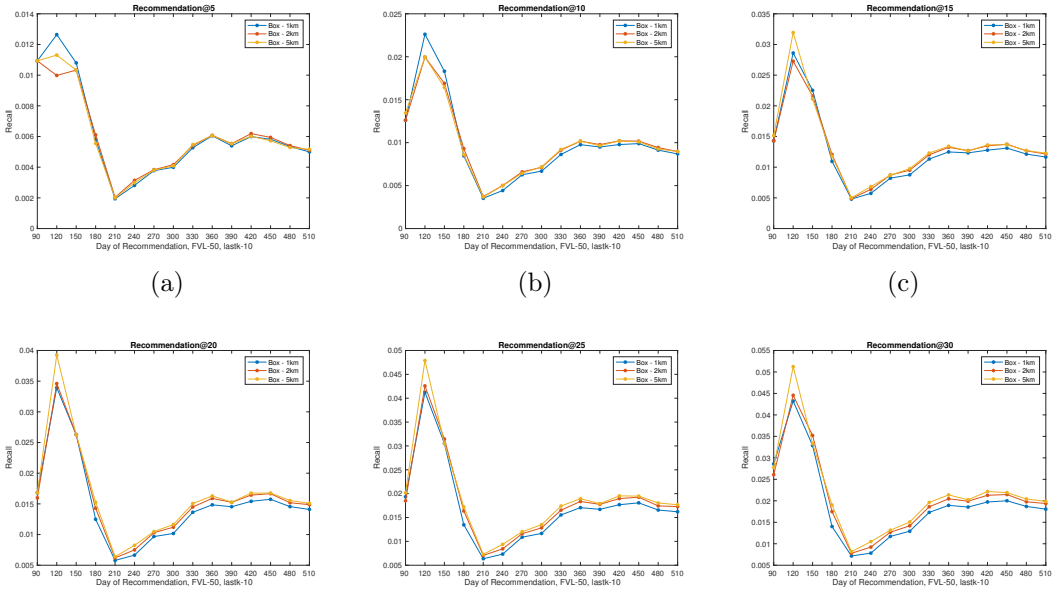
Figure 8.20. Performance ($r@K$) of REGULA on Gowalla for different size of Bounding Box, $\alpha = 100$, $FVL = 30$, $lastk$=10.



(a) (b) (c)

(d) (e) (f)

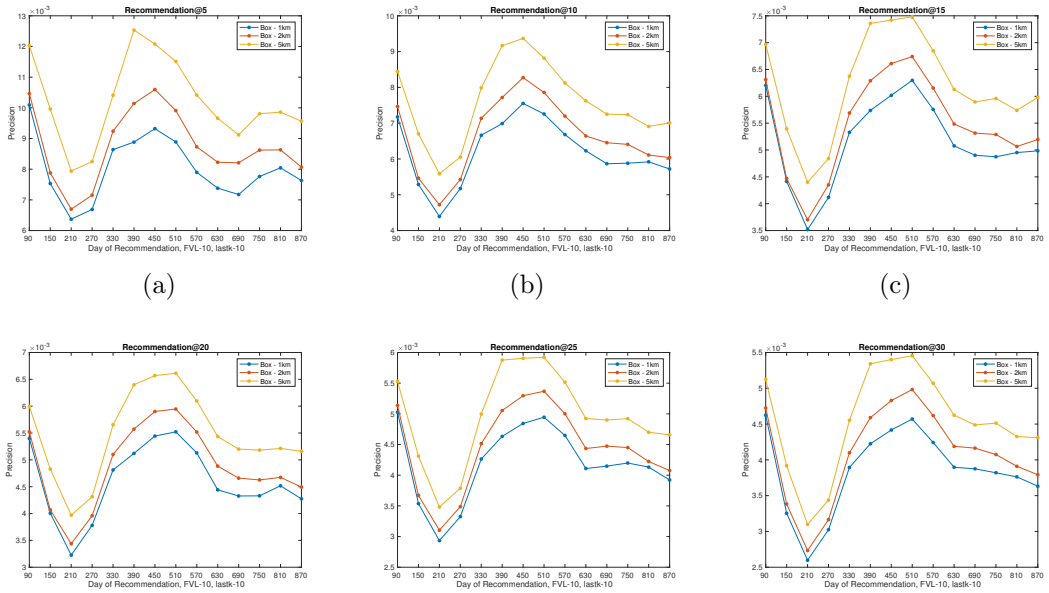
Figure 8.21. Performance ($r@K$) of REGULA on Gowalla for different size of Bounding Box, $\alpha = 100$, $FVL = 40$, $lastk$=10.



(a) (b) (c)

(d) (e) (f)

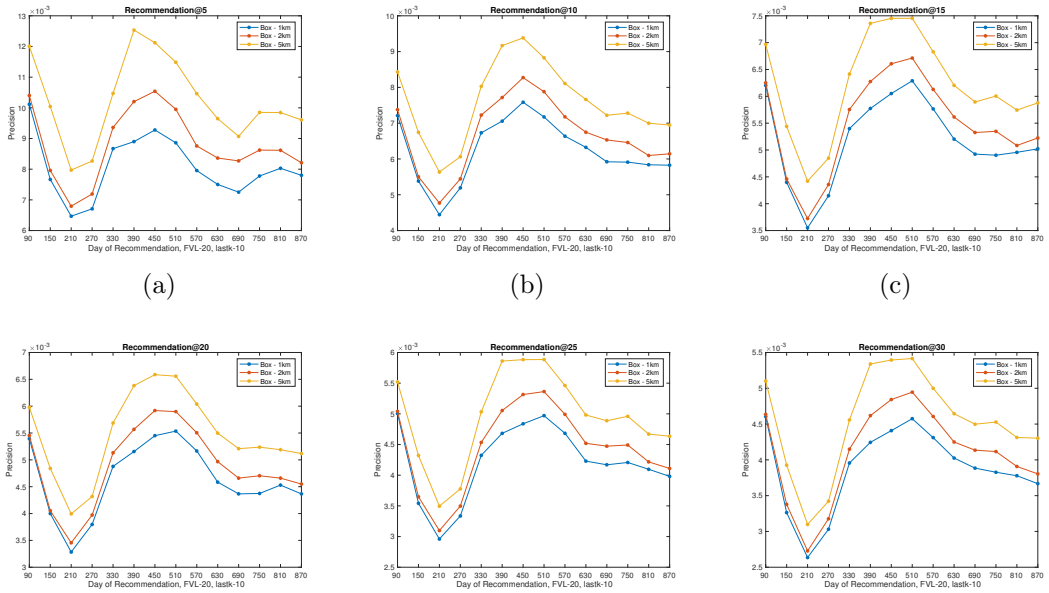
Figure 8.22. Performance ($r@K$) of REGULA on Gowalla for different size of Bounding Box, $\alpha = 100$, $FVL = 40$, $lastk$=10.



(a) (b) (c)

(d) (e) (f)

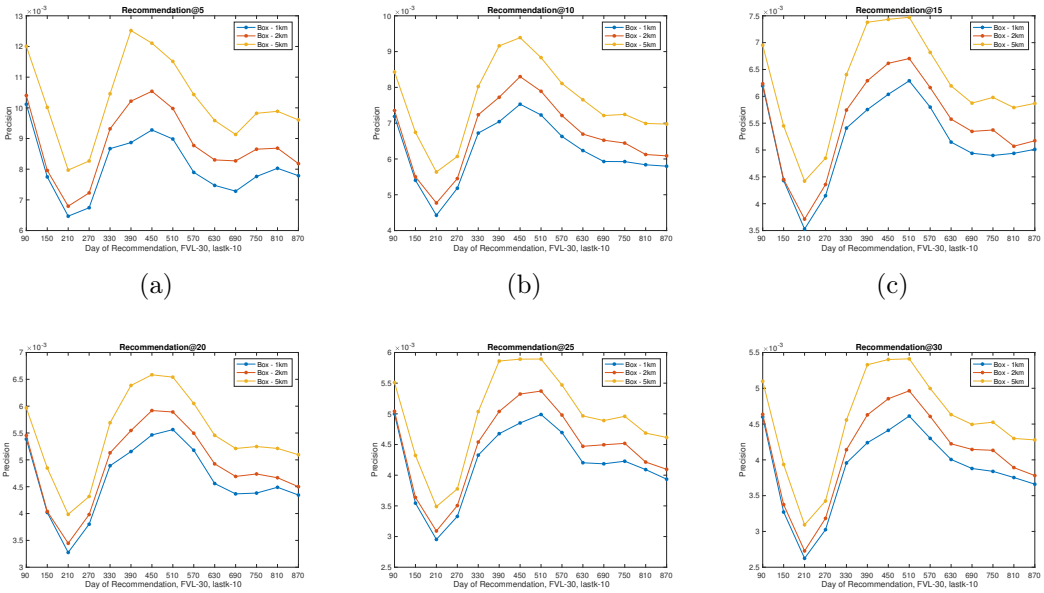
Figure 8.23. Performance ($p@K$) of REGULA on Brightkite for different size of Bounding Box, $\alpha = 100$, $FVL = 10$, $lastk$=10.



(a) (b) (c)

(d) (e) (f)

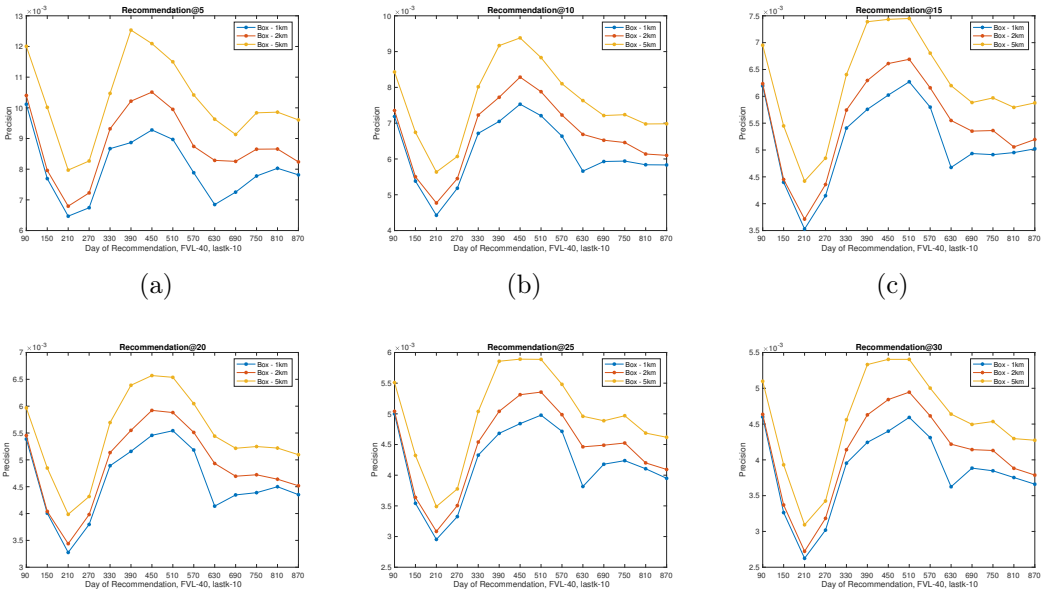
Figure 8.24. Performance ($p@K$) of REGULA on Brightkite for different size of Bounding Box, $\alpha = 100$, $FVL = 20$, $lastk$=10.



(a) (b) (c)

(d) (e) (f)

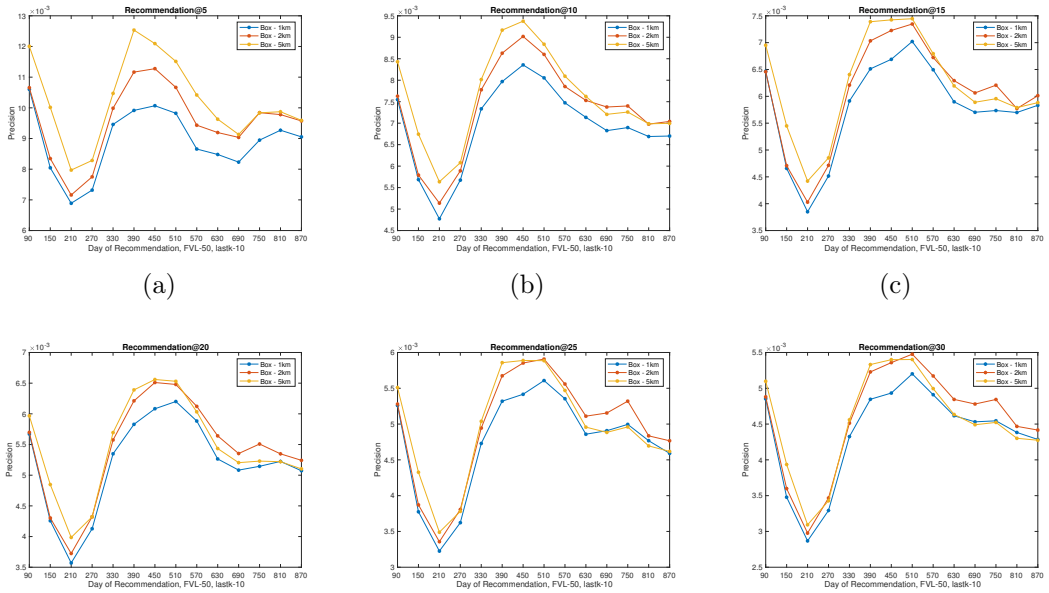
Figure 8.25. Performance ($p@K$) of REGULA on Brightkite for different size of Bounding Box, $\alpha = 100$, $FVL = 30$, $lastk$=10.



(a) (b) (c)

(d) (e) (f)

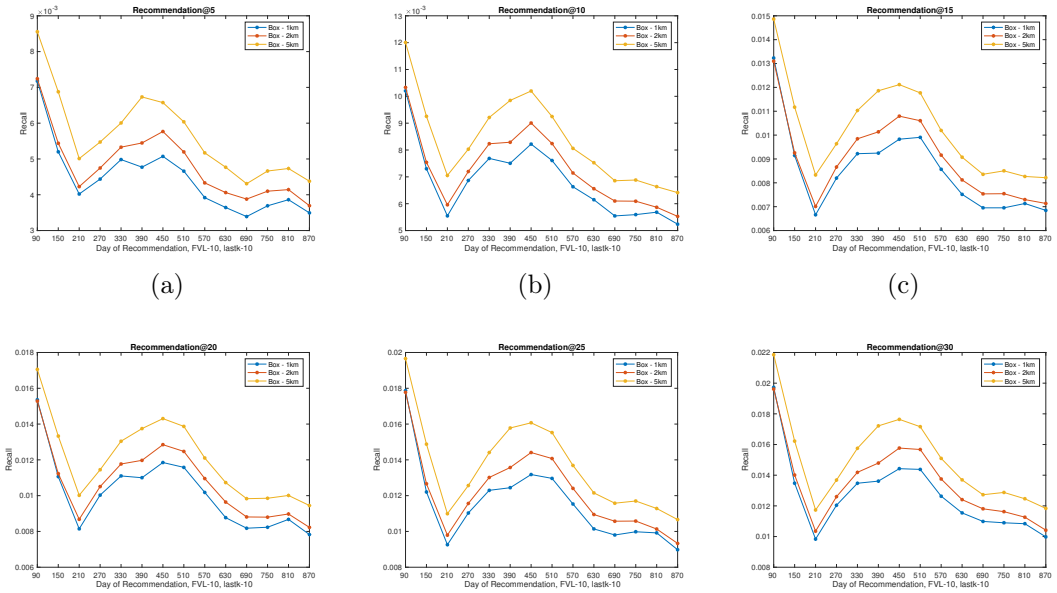
Figure 8.26. Performance ($p@K$) of REGULA on Brightkite for different size of Bounding Box, $\alpha = 100$, $FVL = 40$, $lastk$=10.



(a) (b) (c)

(d) (e) (f)

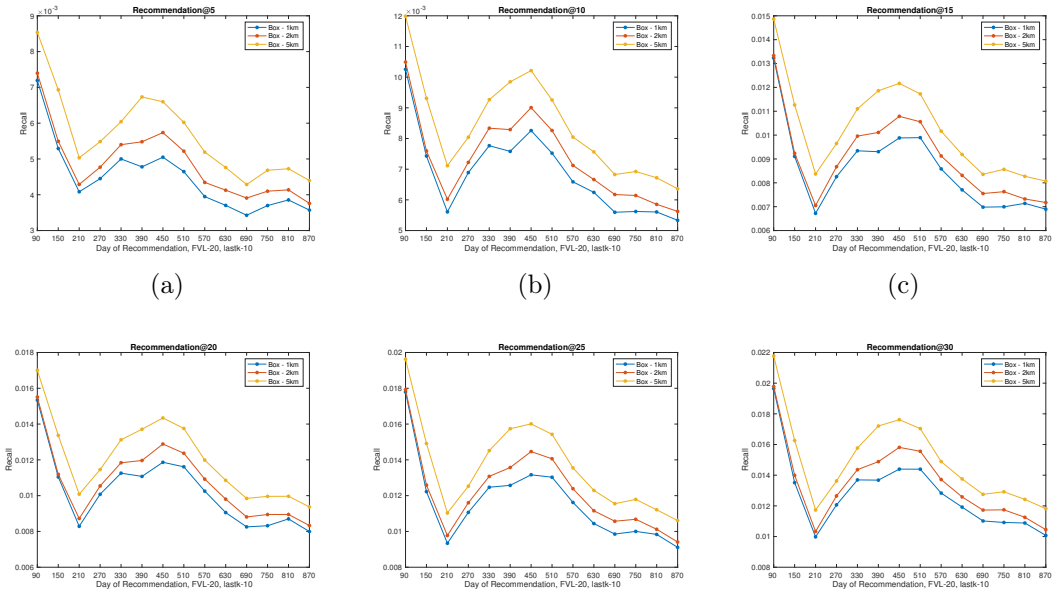
Figure 8.27. Performance ($p@K$) of REGULA on Brightkite for different size of Bounding Box, $\alpha = 100$, $FVL = 50$, $lastk$=10.



(a) (b) (c)

(d) (e) (f)

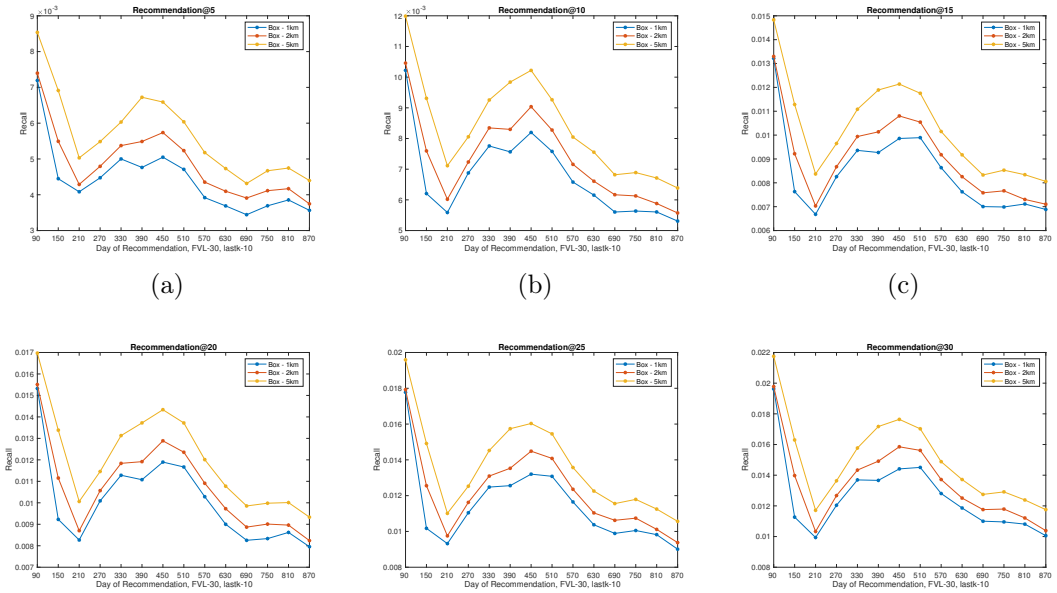
Figure 8.28. Performance ($r@K$) of REGULA on Brightkite for different size of Bounding Box, $\alpha = 100$, $FVL = 10$, $lastk$=10.



(a) (b) (c)

(d) (e) (f)

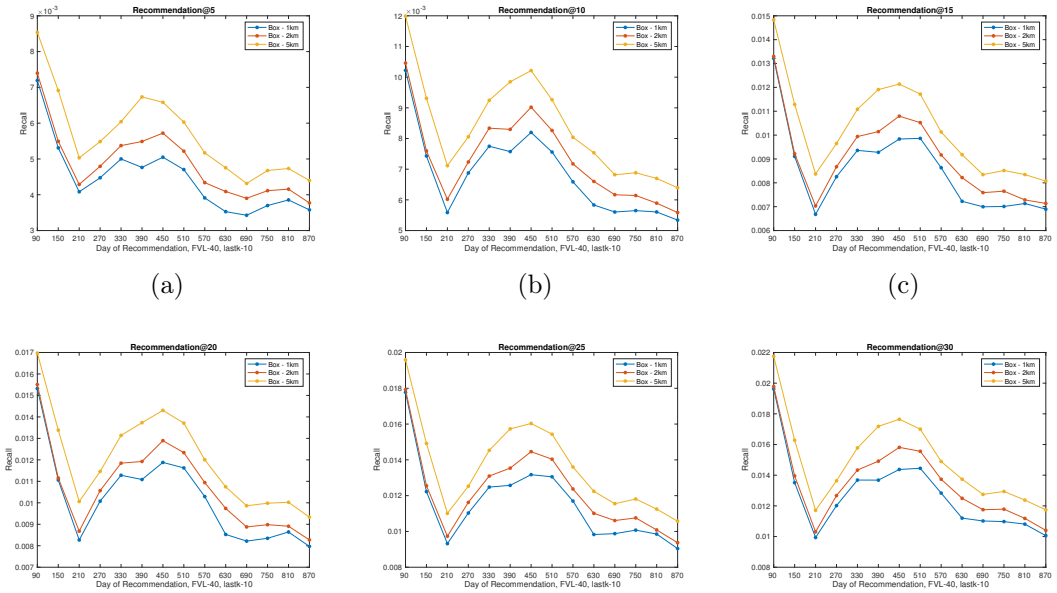
Figure 8.29. Performance ($r@K$) of REGULA on Brightkite for different size of Bounding Box, $\alpha = 100$, $FVL = 20$, $lastk$=10.



(a) (b) (c)

(d) (e) (f)

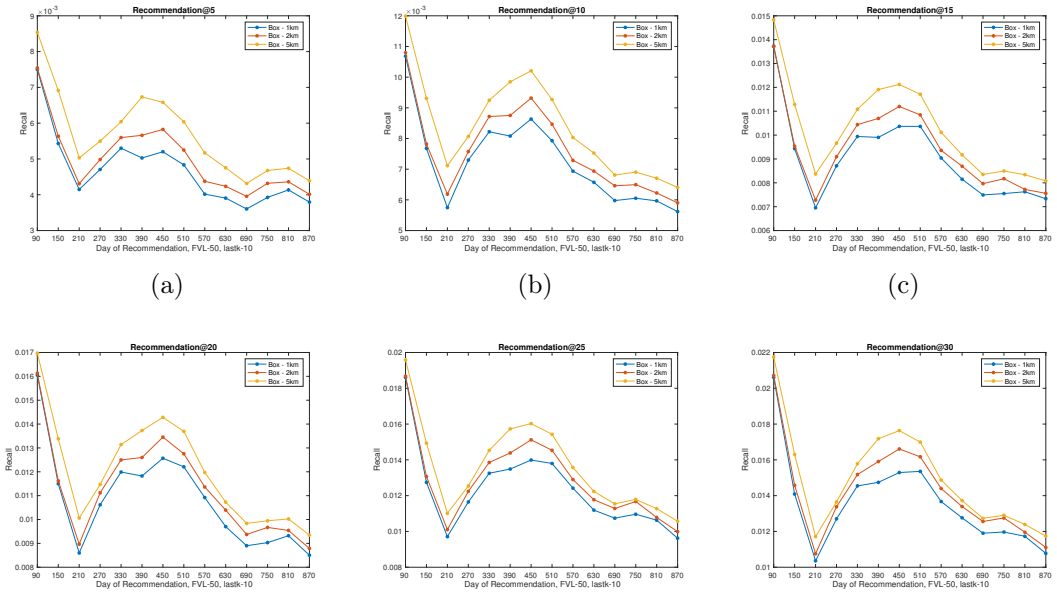
Figure 8.30. Performance ($r@K$) of REGULA on Brightkite for different size of Bounding Box, $\alpha = 100$, $FVL = 30$, $lastk$=10.



(a) (b) (c)

(d) (e) (f)

Figure 8.31. Performance ($r@K$) of REGULA on Brightkite for different size of Bounding Box, $\alpha = 100$, $FVL = 40$, $lastk$=10.



(a) (b) (c)

(d) (e) (f)

Figure 8.32. Performance ($r@K$) of REGULA on Brightkite for different size of Bounding Box, $\alpha = 100$, $FVL = 40$, $lastk$=10.

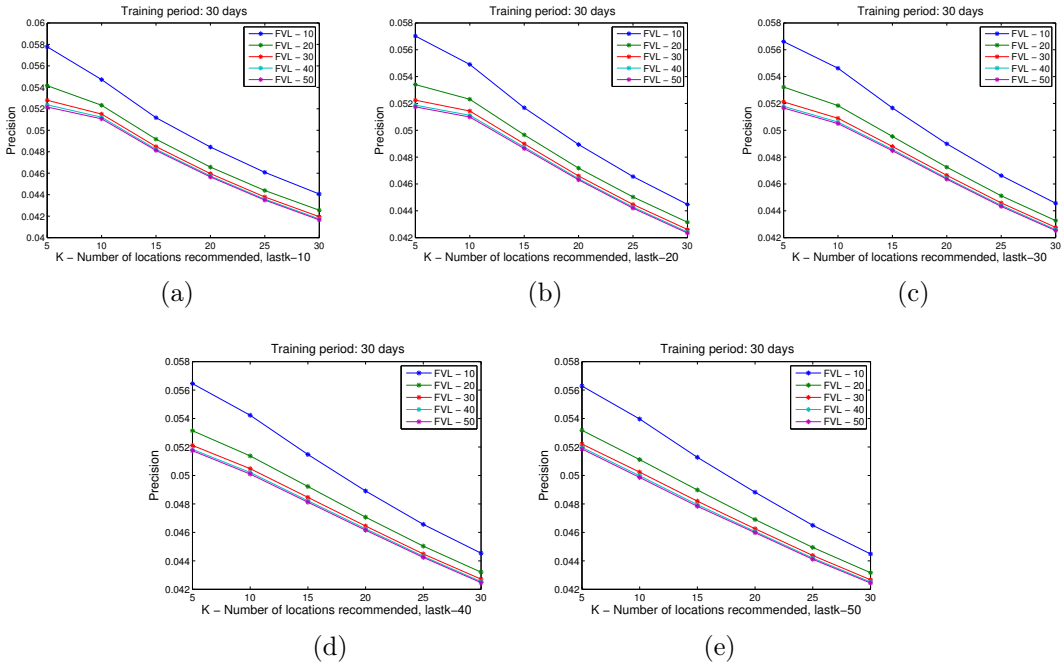


Figure 8.33. Performance ($p@K$) of REGULA on CDR-based LBSN for different FVLs, α=10, Box=1km, and 30 days for training.

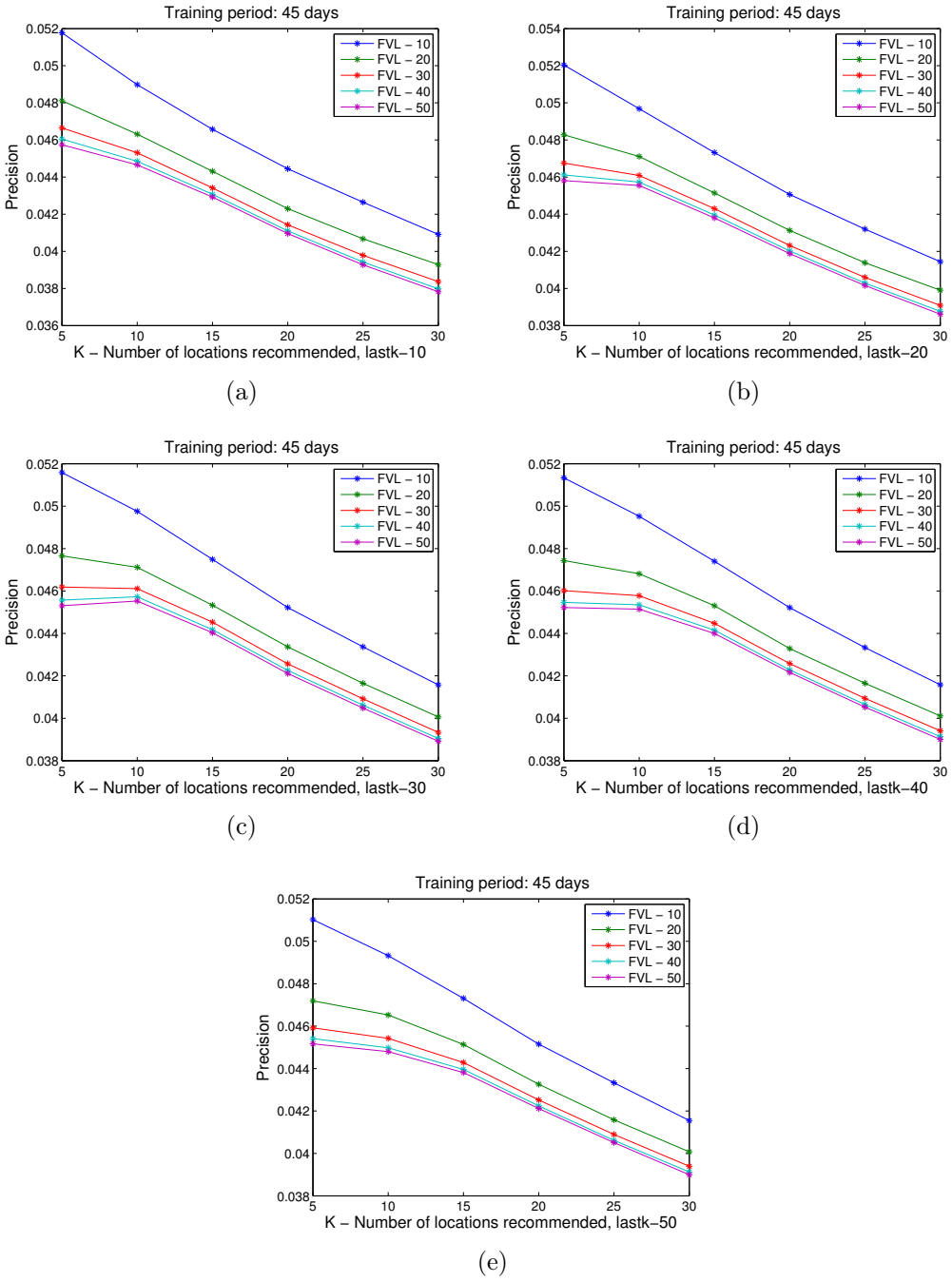


Figure 8.34. Performance ($p@K$) of REGULA on CDR-based LBSN for different FVLs, α=10, Box=1km, and 45 days for training.

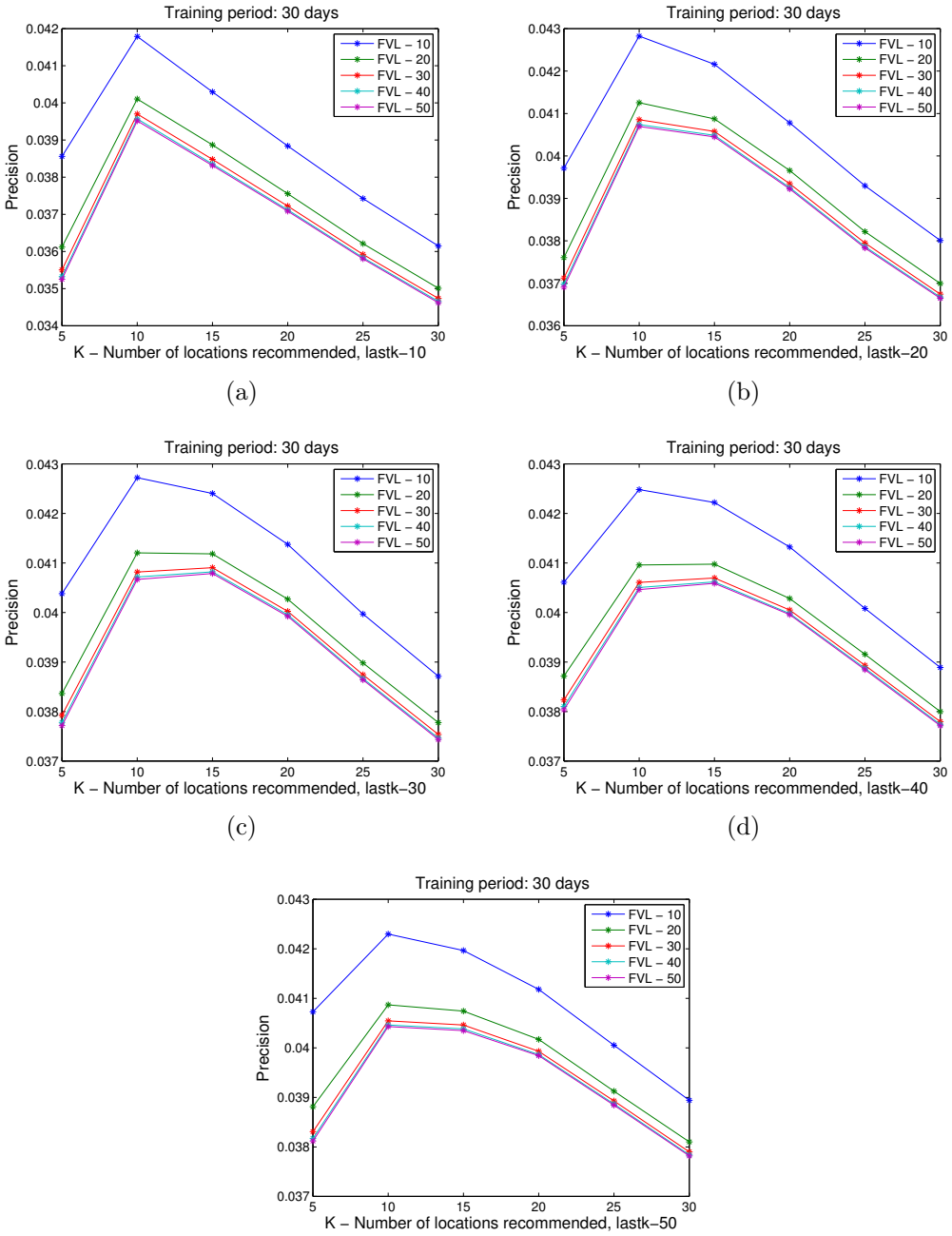


Figure 8.35. Performance ($p@K$) of REGULA on CDR-based LBSN for different FVLs, α=10, Box=2.5km, and 30 days for training.

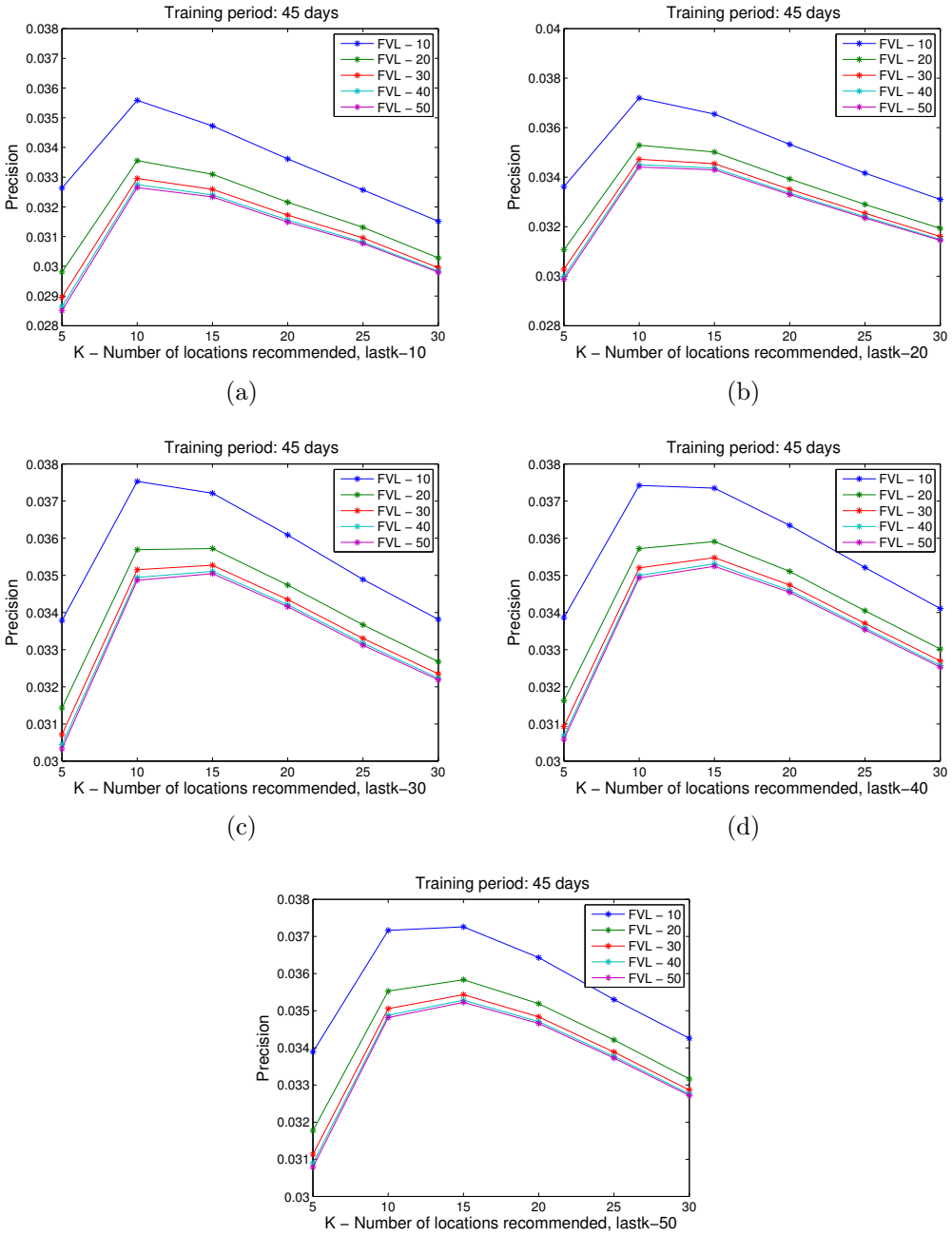


Figure 8.36. Performance ($p@K$) of REGULA on CDR-based LBSN for different FVLs, α=10, Box=2.5km, and 45 days for training.

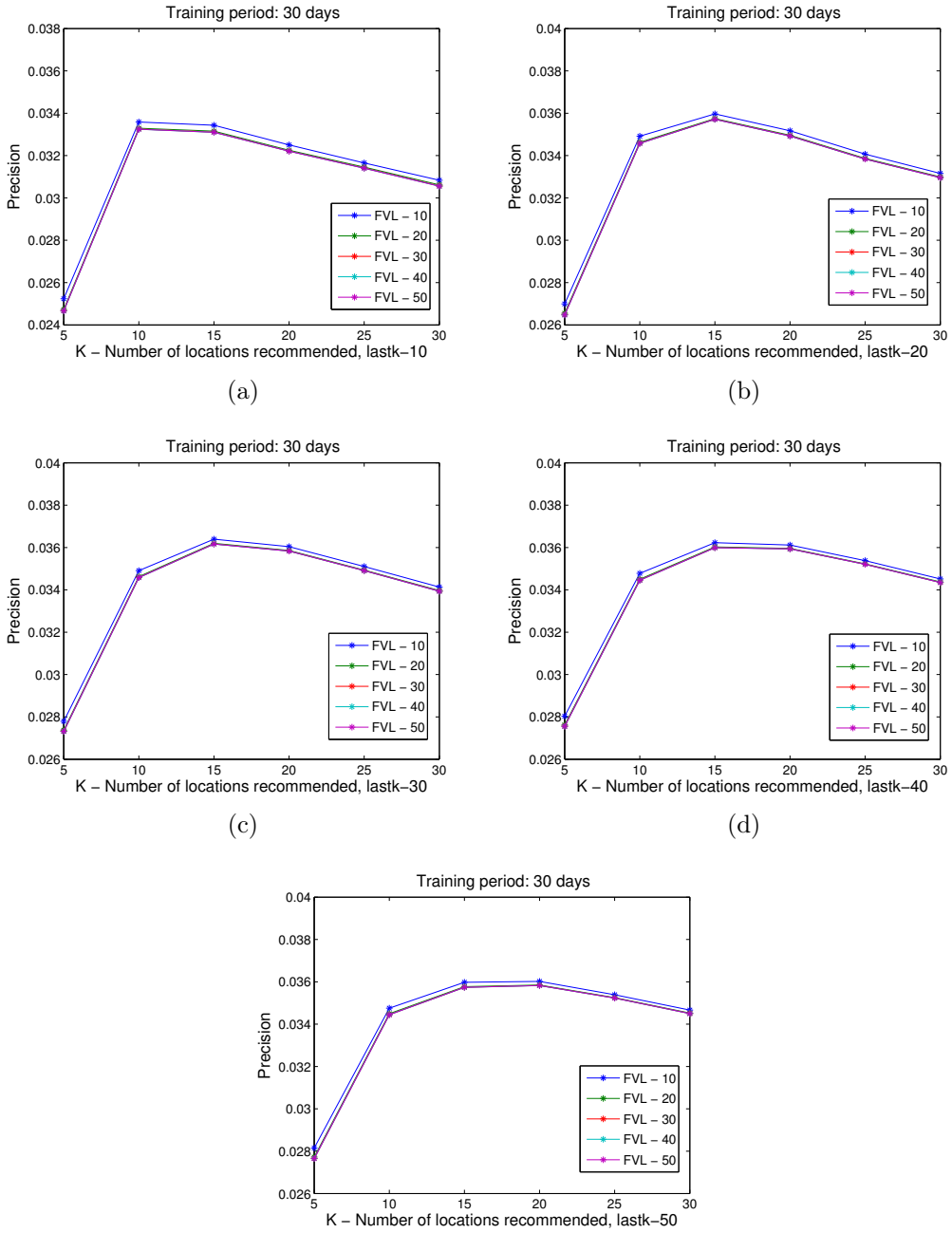


Figure 8.37. Performance ($p@K$) of REGULA on CDR-based LBSN for different FVLs, α=10, Box=5km, and 30 days for training.

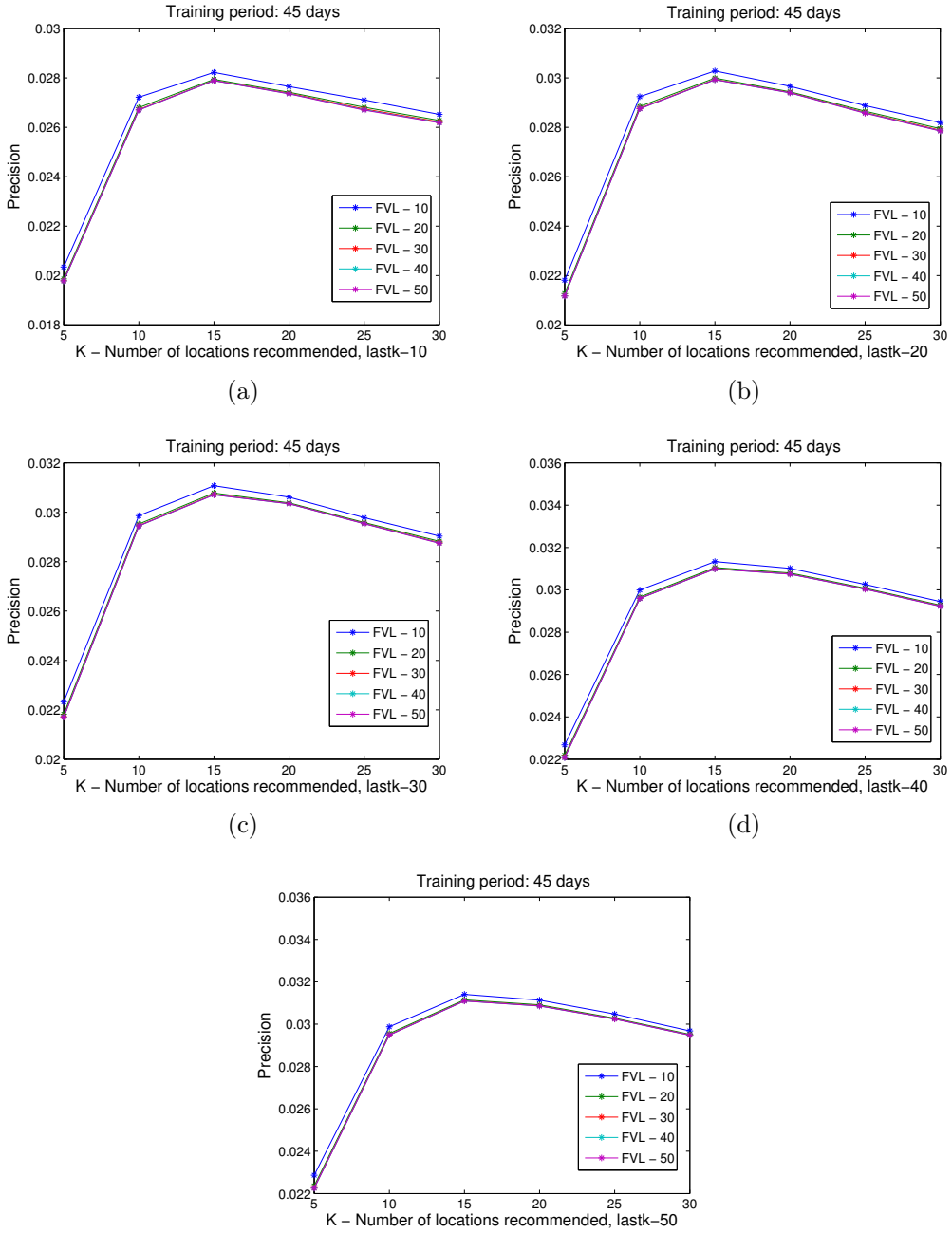


Figure 8.38. Performance ($p@K$) of REGULA on CDR-based LBSN for different FVLs, α=10, Box=5km, and 45 days for training.

Bibliography

Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering*, 17(6):734–749, 2005.

Marcelo G Armentano, Daniela Godoy, and AA Amandi. Recommending information sources to information seekers in twitter. In *International workshop on social web mining*, 2011.

Jie Bao, Yu Zheng, David Wilkie, and Mohamed F Mokbel. A survey on recommendations in location-based social networks. *ACM Transaction on Intelligent Systems and Technology*, 2013.

Jie Bao, Yu Zheng, David Wilkie, and Mohamed Mokbel. Recommendations in location-based social networks: a survey. *GeoInformatica*, 19(3):525–565, 2015.

Yoshua Bengio et al. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

Vincent D Blondel, Adeline Decuyper, and Gautier Krings. A survey of results on mobile phone datasets analysis. *EPJ Data Science*, 4(1):10, 2015.

Jesús Bobadilla, Fernando Ortega, Antonio Hernando, and Abraham Gutiérrez. Recommender systems survey. *Knowledge-based systems*, 46:109–132, 2013.

Léon Bottou. *Stochastic Gradient Descent Tricks*, pages 421–436. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-35289-8. doi: 10.1007/978-3-642-35289-8_25. URL https://doi.org/10.1007/978-3-642-35289-8_25.

Soumen Chakrabarti, Byron Dom, Prabhakar Raghavan, Sridhar Rajagopalan, David Gibson, and Jon Kleinberg. Automatic resource compilation by ana-

lyzing hyperlink structure and associated text. *Computer networks and ISDN systems*, 30(1-7):65–74, 1998.

Kailong Chen, Tianqi Chen, Guoqing Zheng, Ou Jin, Enpeng Yao, and Yong Yu. Collaborative personalized tweet recommendation. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pages 661–670. ACM, 2012a.

Minmin Chen, Zhixiang Xu, Kilian Weinberger, and Fei Sha. Marginalized denoising autoencoders for domain adaptation. *arXiv preprint arXiv:1206.4683*, 2012b.

Wen-Yen Chen, Dong Zhang, and Edward Y Chang. Combinational collaborative filtering for personalized community recommendation. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 115–123. ACM, 2008.

Eunjoon Cho, Seth A. Myers, and Jure Leskovec. Friendship and mobility: User movement in location-based social networks. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pages 1082–1090, New York, NY, USA, 2011a. ACM. ISBN 978-1-4503-0813-7. doi: 10.1145/2020408.2020579. URL `http://doi.acm.org/10.1145/2020408.2020579`.

Eunjoon Cho, Seth A Myers, and Jure Leskovec. Friendship and mobility: user movement in location-based social networks. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1082–1090. ACM, 2011b.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Gated feedback recurrent neural networks. In *International Conference on Machine Learning*, pages 2067–2075, 2015.

Thomas G Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural computation*, 10(7):1895–1923, 1998.

Ye Ding, Siyuan Liu, Jiansu Pu, and Lionel M Ni. Hunts: A trajectory recommendation system for effective and efficient hunting of taxi passengers. In *Mobile Data Management (MDM), 2013 IEEE 14th International Conference on*, volume 1, pages 107–116. IEEE, 2013.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

Paul W Farris, Neil Bendle, Phillip Pfeifer, and David Reibstein. *Marketing metrics: The definitive guide to measuring marketing performance*. Pearson Education, 2010.

Gregory Ference, Mao Ye, and Wang-Chien Lee. Location recommendation for out-of-town users in location-based social networks. In *Proceedings of the 22Nd ACM International Conference on Conference on Information & Knowledge Management*, CIKM '13, pages 721–726, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2263-8. doi: 10.1145/2505515.2505637. URL http://doi.acm.org/10.1145/2505515.2505637.

Foursquare. http://foursquare.com.

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*, pages 93–100. ACM, 2013.

Huiji Gao, Jiliang Tang, Xia Hu, and Huan Liu. Content-aware point of interest recommendation on location-based social networks. In *AAAI*, pages 1721–1727, 2015.

Global-Smartphone-Penetration. https://www.statista.com/statistics/203734/global-smartphone-penetration-per-capita-since-2005/.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010a.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010b. PMLR. URL http://proceedings.mlr.press/v9/glorot10a.html.

Carlos A Gomez-Uribe and Neil Hunt. The netflix recommender system: Algorithms, business value, and innovation. *ACM Transactions on Management Information Systems (TMIS)*, 6(4):13, 2016.

Marta C Gonzalez, Cesar A Hidalgo, and Albert-Laszlo Barabasi. Understanding individual human mobility patterns. *Nature*, 453(7196):779–782, 2008a.

Marta C Gonzalez, Cesar A Hidalgo, and Albert-Laszlo Barabasi. Understanding individual human mobility patterns. *Nature*, 453(7196):779–782, 2008b.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

Gowalla. https://en.wikipedia.org/wiki/gowalla.

John Hannon, Mike Bennett, and Barry Smyth. Recommending twitter users to follow using content and collaborative filtering approaches. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 199–206. ACM, 2010.

Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. Fast matrix factorization for online recommendation with implicit feedback. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '16, pages 549–558, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4069-4. doi: 10.1145/2911451.2911489. URL `http://doi.acm.org/10.1145/2911451.2911489`.

Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*, pages 173–182. International World Wide Web Conferences Steering Committee, 2017.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 263–272. Ieee, 2008.

Instagram. https://www.instagram.com.

Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446, 2002.

Zhaoyan Jin, Dianxi Shi, Quanyuan Wu, Huining Yan, and Hua Fan. Lbsnrank: personalized pagerank on location-based social networks. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, pages 980–987. ACM, 2012.

Marton Karsai, Kimmo Kaski, Albert-László Barabási, and János Kertész. Universal features of correlated bursty behaviour. *Scientific Reports*, 2, 2012.

Pavlos Kefalas, Panagiotis Symeonidis, and Yannis Manolopoulos. A graph-based taxonomy of recommendation algorithms and systems in lbsns. *IEEE Transactions on Knowledge and Data Engineering*, 28(3):604–622, 2016.

Younghoon Kim and Kyuseok Shim. Twilite: A recommendation system for twitter using a probabilistic model based on latent dirichlet allocation. *Information Systems*, 42:59–77, 2014.

Xiangjie Kong, Feng Xia, Jinzhong Wang, Azizur Rahim, and Sajal K Das. Time-location-relationship combined service recommendation based on taxi trajectory data. *IEEE Transactions on Industrial Informatics*, 13(3):1202–1212, 2017.

Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, Aug 2009. ISSN 0018-9162. doi: 10.1109/MC.2009.263.

Yehuda Koren. Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pages 426–434, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-193-4. doi: 10.1145/1401890.1401944. URL `http://doi.acm.org/10.1145/1401890.1401944`.

Yehuda Koren. Collaborative filtering with temporal dynamics. *Communications of the ACM*, 53(4):89–97, 2010.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

Renaud Lambiotte, Vincent D Blondel, Cristobald De Kerchove, Etienne Huens, Christophe Prieur, Zbigniew Smoreda, and Paul Van Dooren. Geographical dispersal of mobile communication networks. *Physica A: Statistical Mechanics and its Applications*, 387(21):5317–5325, 2008.

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 Retrieval*, pages 305–314. ACM, 2011.

Ming-Xia Li, Wen-Jie Xie, Zhi-Qiang Jiang, and Wei-Xing Zhou. Communication cliques in mobile phone calling networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2015(11):P11007, 2015a.

Sheng Li, Jaya Kawale, and Yun Fu. Deep collaborative filtering via marginalized denoising auto-encoder. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, pages 811–820, New York, NY, USA, 2015b. ACM. ISBN 978-1-4503-3794-6. doi: 10.1145/2806416.2806527. URL http://doi.acm.org/10.1145/2806416.2806527.

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, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2956-9. doi: 10.1145/2623330.2623638. URL http://doi.acm.org/10.1145/2623330.2623638.

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 *Data Mining (ICDM), 2015 IEEE International Conference on*, pages 261–270. IEEE, 2015.

Defu Lian, Zhenyu Zhang, Yong Ge, Fuzheng Zhang, Nicholas Jing Yuan, and Xing Xie. Regularized content-aware tensor factorization meets temporal-aware location recommendation. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on*, pages 1029–1034. IEEE, 2016.

Defu Lian, Rui Liu, Yong Ge, Kai Zheng, Xing Xie, and Longbing Cao. Discrete content-aware matrix factorization. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, pages 325–334, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4887-4. doi: 10.1145/3097983.3098008. URL http://doi.acm.org/10.1145/3097983.3098008.

Defu Lian, Yong Ge, Fuzheng Zhang, Nicholas Jing Yuan, Xing Xie, Tao Zhou, and Yong Rui. Scalable content-aware collaborative filtering for location recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 2018.

Greg Linden, Brent Smith, and Jeremy York. Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing*, 7(1):76–80, 2003.

Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. Predicting the next location: A recurrent model with spatial and temporal contexts. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 194–200. AAAI Press, 2016. URL http://dl.acm.org/citation.cfm?id=3015812.3015841.

Yong Liu, Wei Wei, Aixin Sun, and Chunyan Miao. Exploiting geographical neighborhood characteristics for location recommendation. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, CIKM '14, pages 739–748, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2598-1. doi: 10.1145/2661829.2662002. URL http://doi.acm.org/10.1145/2661829.2662002.

Zhongqi Lu, Zhicheng Dou, Jianxun Lian, Xing Xie, and Qiang Yang. Content-based collaborative filtering for news topic recommendation. In *AAAI*, pages 217–223, 2015.

Brian McFee, Luke Barrington, and Gert Lanckriet. Learning content similarity for music recommendation. *IEEE transactions on audio, speech, and language processing*, 20(8):2207–2218, 2012.

Raymond J Mooney and Loriene Roy. Content-based book recommending using learning for text categorization. In *Proceedings of the fifth ACM conference on Digital libraries*, pages 195–204. ACM, 2000.

Diala Naboulsi, Marco Fiore, Stephane Ribot, and Razvan Stanica. Large-scale mobile traffic analysis: a survey. *IEEE Communications Surveys & Tutorials*, 18 (1):124–161, 2015.

Anastasios Noulas, Salvatore Scellato, Cecilia Mascolo, and Massimiliano Pontil. An empirical study of geographic user activity patterns in foursquare. *ICwSM*, 11(70-573):2, 2011.

OpenTable. https://www.opentable.com.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

M. Papandrea, M. Zignani, S. Gaito, S. Giordano, and G.P. Rossi. How many places do you visit a day? In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2013 IEEE International Conference on*, pages 218–223, March 2013. doi: 10.1109/PerComW.2013.6529485.

Michela Papandrea, Karim Keramat Jahromi, Matteo Zignani, Sabrina Gaito, Silvia Giordano, and Gian Paolo Rossi. On the properties of human mobility. *Computer Communications*, 87:19–36, 2016.

Christian Quadri, Matteo Zignani, Lorenzo Capra, Sabrina Gaito, and Gian Paolo Rossi. Multidimensional human dynamics in mobile phone communications. *PloS one*, 9(7):e103183, 2014.

Steffen Rendle. Factorization machines with libFM. *ACM Trans. Intell. Syst. Technol.*, 3(3):57:1–57:22, May 2012. ISSN 2157-6904.

Frank Rosenblatt. *Perceptions and the theory of brain mechanisms*. 1962.

Maayan Roth, Assaf Ben-David, David Deutscher, Guy Flysher, Ilan Horn, Ari Leichtberg, Naty Leiser, Yossi Matias, and Ron Merom. Suggesting friends using the implicit social graph. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 233–242. ACM, 2010.

Sebastian Ruder. An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747, 2016. URL `http://arxiv.org/abs/1609.04747`.

Tara N Sainath, Oriol Vinyals, Andrew Senior, and Haşim Sak. Convolutional, long short-term memory, fully connected deep neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 4580–4584. IEEE, 2015.

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*, pages 285–295. ACM, 2001.

Amit Sharma and Baoshi Yan. Pairwise learning in recommendation: Experiments with community recommendation on linkedin. In *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys '13, pages 193–200, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2409-0. doi: 10.1145/2507157.2507175. URL `http://doi.acm.org/10.1145/2507157.2507175`.

Nitai B Silva, Ren Tsang, George DC Cavalcanti, and Jyh Tsang. A graph-based friend recommendation system using genetic algorithm. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1–7. IEEE, 2010.

Brent Smith and Greg Linden. Two decades of recommender systems at amazon.com. *IEEE Internet Computing*, 21(3):12–18, May 2017. ISSN 1089-7801. doi: 10.1109/MIC.2017.72. URL https://doi.org/10.1109/MIC.2017.72.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Xiaoyuan Su and Taghi M Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in artificial intelligence*, 2009:4, 2009.

Twitter. https://twitter.com.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM, 2008.

Hao Wang, Manolis Terrovitis, and Nikos Mamoulis. Location recommendation in location-based social networks using user check-in data. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL'13, pages 374–383, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2521-9. doi: 10.1145/2525314.2525357. URL http://doi.acm.org/10.1145/2525314.2525357.

Hao Wang, Naiyan Wang, and Dit-Yan Yeung. Collaborative deep learning for recommender systems. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pages 1235–1244, New York, NY, USA, 2015a. ACM. ISBN 978-1-4503-3664-2. doi: 10.1145/2783258.2783273. URL http://doi.acm.org/10.1145/2783258.2783273.

Hao Wang, Naiyan Wang, and Dit-Yan Yeung. Collaborative deep learning for recommender systems. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1235–1244. ACM, 2015b.

Suhang Wang, Yilin Wang, Jiliang Tang, Kai Shu, Suhas Ranganath, and Huan Liu. What your images reveal: Exploiting visual contents for point-of-interest

recommendation. In *Proceedings of the 26th International Conference on World Wide Web*, pages 391–400. International World Wide Web Conferences Steering Committee, 2017.

Xinxi Wang and Ye Wang. Improving content-based and hybrid music recommendation using deep learning. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 627–636. ACM, 2014.

Yingzi Wang, Nicholas Jing Yuan, Defu Lian, Linli Xu, Xing Xie, Enhong Chen, and Yong Rui. Regularity and conformity: Location prediction using heterogeneous mobility data. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1275–1284. ACM, 2015c.

Yao Wu, Christopher DuBois, Alice X. Zheng, and Martin Ester. Collaborative denoising auto-encoders for top-n recommender systems. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, WSDM '16, pages 153–162, New York, NY, USA, 2016a. ACM. ISBN 978-1-4503-3716-8. doi: 10.1145/2835776.2835837. URL `http://doi.acm.org/10.1145/2835776.2835837`.

Yao Wu, Christopher DuBois, Alice X Zheng, and Martin Ester. Collaborative denoising auto-encoders for top-n recommender systems. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, pages 153–162. ACM, 2016b.

Bin Xia, Yun Li, Qianmu Li, and Tao Li. Attention-based recurrent neural network for location recommendation. In *Intelligent Systems and Knowledge Engineering (ISKE), 2017 12th International Conference on*, pages 1–6. IEEE, 2017.

Xing Xie. Potential friend recommendation in online social network. In *Proceedings of the 2010 IEEE/ACM Int'L Conference on Green Computing and Communications & Int'L Conference on Cyber, Physical and Social Computing*, GREENCOM-CPSCOM '10, pages 831–835, Washington, DC, USA, 2010. IEEE Computer Society. ISBN 978-0-7695-4331-4. doi: 10.1109/GreenCom-CPSCom.2010.28. URL `http://dx.doi.org/10.1109/GreenCom-CPSCom.2010.28`.

Rui Yan, Mirella Lapata, and Xiaoming Li. Tweet recommendation with graph co-ranking. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 516–525. Association for Computational Linguistics, 2012.

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*, pages 458–461. ACM, 2010.

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 Retrieval*, pages 325–334. ACM, 2011.

Peifeng Yin, Mao Ye, Wang-Chien Lee, and Zhenhui Li. Mining gps data for trajectory recommendation. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 50–61. Springer, 2014.

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, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2034-4. doi: 10.1145/2484028.2484030. URL `http://doi.acm.org/10.1145/2484028.2484030`.

Quan Yuan, Gao Cong, and Aixin Sun. Graph-based point-of-interest recommendation with geographical and temporal influences. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, CIKM '14, pages 659–668, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2598-1. doi: 10.1145/2661829.2661983. URL `http://doi.acm.org/10.1145/2661829.2661983`.

Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. Collaborative knowledge base embedding for recommender systems. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 353–362. ACM, 2016a.

Fuzheng Zhang, Nicholas Jing Yuan, Kai Zheng, Defu Lian, Xing Xie, and Yong Rui. Exploiting dining preference for restaurant recommendation. In *Proceedings of the 25th International Conference on World Wide Web*, pages 725–735. International World Wide Web Conferences Steering Committee, 2016b.

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, 2016.

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*, pages 103–112. ACM, 2014.

Shuai Zhang, Lina Yao, and Aixin Sun. Deep learning based recommender system: A survey and new perspectives. *arXiv preprint arXiv:1707.07435*, 2017.

Guanjie Zheng, Fuzheng Zhang, Zihan Zheng, Yang Xiang, Nicholas Jing Yuan, Xing Xie, and Zhenhui Li. Drn: A deep reinforcement learning framework for news recommendation. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 167–176. International World Wide Web Conferences Steering Committee, 2018.

Vincent Wenchen Zheng, Yu Zheng, and Qiang Yang. Joint learning user's activities and profiles from gps data. In *Proceedings of the 2009 International Workshop on Location Based Social Networks*, pages 17–20. ACM, 2009.

Vincent Wenchen Zheng, Bin Cao, Yu Zheng, Xing Xie, and Qiang Yang. Collaborative filtering meets mobile recommendation: A user-centered approach. In *AAAI*, volume 10, pages 236–241, 2010a.

Yu Zheng, Xing Xie, and Wei-Ying Ma. Geolife: A collaborative social networking service among user, location and trajectory. *IEEE Data Eng. Bull.*, 33(2):32–39, 2010b.