# DEVELOPING RECOMMENDATION TECHNIQUES FOR LOCATION BASED SOCIAL NETWORKS USING RANDOM WALK

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

HAKAN BAĞCI

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
COMPUTER ENGINEERING

DECEMBER 2015

Approval of the thesis:

**DEVELOPING RECOMMENDATION TECHNIQUES FOR LOCATION BASED SOCIAL NETWORKS USING RANDOM WALK**

submitted by **HAKAN BAĞCI** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Gülbin Dural Ünver _____
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Adnan Yazıcı _____
Head of Department, **Computer Engineering**

Assoc. Prof. Dr. Pınar Karagöz _____
Supervisor, **Computer Engineering Department, METU**

**Examining Committee Members:**

Prof. Dr. İsmail Hakkı Toroslu
Computer Engineering Department, METU _____

Assoc. Prof. Dr. Pınar Karagöz
Computer Engineering Department, METU _____

Prof. Dr. Adnan Yazıcı
Computer Engineering Department, METU _____

Assoc. Prof. Dr. İbrahim Körpeoğlu
Computer Engineering Department, Bilkent University _____

Assoc. Prof. Dr. Pınar Duygulu Şahin
Computer Engineering Department, Hacettepe University _____

**Date:** _____

**I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.**

Name, Last Name: HAKAN BAĞCI

Signature :

# ABSTRACT

DEVELOPING RECOMMENDATION TECHNIQUES FOR LOCATION BASED SOCIAL NETWORKS USING RANDOM WALK

Bağcı, Hakan

Ph.D., Department of Computer Engineering

Supervisor : Assoc. Prof. Dr. Pınar Karagöz

December 2015, 104 pages

The location-based social networks (LBSN) enable users to check-in their current location and share it with other users. The accumulated check-in data can be employed for the benefit of users by providing personalized recommendations. In this thesis, we propose three recommendation algorithms for location-based social networks. These are random walk based context-aware location (CLoRW), activity (RWCAR) and friend (RWCFR) recommendation algorithms. All the algorithms consider the current context (i.e. current social relations, personal preferences and current location) of the user to provide personalized recommendations. We propose an undirected unweighted graph model for representing LBSN data that contains users, locations and activities. We build a graph according to the current context of the user for each algorithm depending on this LBSN model. A random walk with restart approach is employed on this graph to predict the recommendation scores. Lists of users, locations and activities are recommended to users after ordering the nodes according to estimated scores. We compare CLoRW with popularity-based, friend-based and expert-based baselines, collaborative filtering approach and a similar work in the literature. According to results, our location recommendation algorithm outperforms these approaches in all of the test cases. Moreover, we also compare RWCAR and RWCFR algorithms with respective popularity-based, friend-based and expert-based baselines. In all of the experiments, RWCAR and RWCFR perform better than the

baselines. The results clearly indicate that random walk based context-aware recommendation approach is a good candidate for recommending locations, activities and friends for LBSNs.

Keywords: Location-Based Social Networks, Random Walk, Location Recommendation, Activity Recommendation, Friend Recommendation

# ÖZ

KONUM TABANLI SOSYAL AĞLAR İÇİN RASTGELE YÜRÜYÜŞ YÖNTEMİ KULLANARAK ÖNERME TEKNİKLERİ GELİŞTİRME

Bağcı, Hakan
Doktora, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi : Doç. Dr. Pınar Karagöz

Aralık 2015 , 104 sayfa

Konum tabanlı sosyal ağlar kullanıcıların şu anda bulunduğu konumu kaydetmesine ve diğer kullanıcılarla paylaşmasına olanak sağlar. Birikmiş olan konum kayıt bilgileri kullanıcıların yararına kullanılarak kişiselleştirilmiş öneriler yapılabilir. Bu tezde konum tabanlı sosyal ağlar için üç farklı önerme algoritması sunuyoruz. Bu algoritmalar rastgele yürüyüş yaklaşımı tabanlı bağlam farkında konum (CLoRW), aktivite (RWCAR) ve arkadaş (RWCFR) önerme algoritmalarıdır. Bütün bu algoritmalar kişiselleştirilmiş öneriler yapabilmek için kullanıcının mevcut şartlarını (mevcut sosyal bağlantıları, kişisel tercihleri ve mevcut konumu) dikkate almaktadır. Kullanıcı, konum ve aktiviteleri içeren konum tabanlı sosyal ağ verilerini ifade edebilmek için yönsüz ağırlıksız bir çizge modeli öneriyoruz. Bu konum tabanlı sosyal ağ modeline dayanarak kullanıcının şu anki bağlamına göre her bir algoritma için bir çizge oluşturuyoruz. Öneri skorlarını tahmin etmek için bu çizge üzerinde tekrar başlamalı rastgele yürüyüş yaklaşımı kullanılmaktadır. Öneri skorları sıralandıktan sonra kullanıcı, konum ve aktivite listeleri kullanıcılara önerilir. CLoRW algoritmasını popülerlik tabanlı, arkadaşlık tabanlı ve uzmanlık tabanlı, işbirlikçi filtreleme yaklaşımı ve literatürdeki benzer bir algoritma ile kıyasladık. Sonuçlara göre, konum önerme algoritmamız tüm test senaryolarında bu yaklaşımlardan daha iyi performans göstermiştir. Ayrıca RWCAR ve RWCFR algoritmalarını da popülerlik tabanlı, arkadaşlık tabanlı ve uzmanlık tabanlı referans yaklaşımlarla kıyasladık. Tüm testlerde RWCAR

ve RWCFR referans yaklaşımlardan daha iyi performans göstermiştir. Bu sonuçlar açıkça göstermektedir ki rastgele yürüyüş tabanlı bağlam farkında önerme yaklaşımı, konum tabanlı sosyal ağlarda konum, aktivite ve arkadaş önermek için iyi bir adaydır.

Anahtar Kelimeler: Konum Tabanlı Sosyal Ağlar, Rastgele Yürüyüş, Konum Önerme, Aktivite Önerme, Arkadaş Önerme

*To my wife and son*

# ACKNOWLEDGMENTS

First and foremost I am heartily thankful to my supervisor Assoc. Prof. Dr. Pınar Karagöz, for her professional guidance and suggestions throughout my thesis studies.

I acknowledge with thanks and appreciation to the thesis monitoring committee members Prof. Dr. Adnan Yazıcı and Prof. Dr. İbrahim Körpeoğlu for their support and feedback on my thesis work.

I would like to show my gratitude to defense jury members Prof. Dr. Hakkı Toroslu and Assoc. Prof. Dr. Pınar Duygulu Şahin for reviewing and evaluating my thesis.

I would also like to thank TUBİTAK BİLGEM İLTAREN for deeply supporting my graduate studies.

I thank to my wife and my son for their patience throughout my thesis studies.

Finally, I would like to thank my parents and my brother for their love, and endless support and guidance during my educational life.

# TABLE OF CONTENTS

# LIST OF TABLES

TABLES

# LIST OF FIGURES

FIGURES

# LIST OF ABBREVIATIONS

| | |
|---|---|
| ACF | Activity-based CF |
| ALAPR | Also-Like based Activity Recommendation |
| APR | Average Percentile Ranking |
| AUC | Area Under the ROC Curve |
| BN | Bayesian Networks |
| CADC | Community-based Agglomerative-Divisive Clustering |
| CFAPR | Collaborative Filtering based Activity-Partner Recommendation |
| CGAR | Collaborative Group Activity Recommender |
| CLAF | Collaborative Location and Activity Filtering |
| CLM | Community Location Model |
| CLR | Collaborative Location Recommendation |
| CLoRW | Context-aware Location Recommendation with Random Walk |
| CLoRW_A | Context-Aware Location Recommendation with Random Walk using Activities |
| CLoRW_A_F | Context-Aware Location Recommendation with Random Walk using Activities (Filtered Version) |
| CLoWRW | Context-Aware Location Recommendation with Weighted Random Walk |
| CF | Collaborative Filtering |
| DBSCAN | Density-based Spatial Clustering of Applications with Noise |
| EBAR | Expert-Based Activity Recommendation |
| EBFR | Expert-Based Friend Recommendation |
| EWI | Equal Width Intervals |
| EM | Expectation Maximization |
| FBAR | Friend-Based Activity Recommendation |
| FBFR | Friend-Based Friend Recommendation |
| GEFR | Geo-Friends Recommendation |
| GPS | Global Positioning System |

| | |
|---|---|
| HGSM | Hierarchical Graph Based Similarity Measurement |
| HITS | Hypertext Induced Topic Search |
| HMM | Hidden Markov Model |
| HOSVD | Higher-Order Singular Value Decomposition |
| ITR | Incremental Tensor Reduction |
| İLTAREN | İleri Teknolojiler Araştırma Enstitüsü (Advanced Technologies Research Institute) |
| kNN | K Nearest Neighbor |
| LA | Los Angeles |
| LBSN | Location-based Social Network |
| LCF | Location-based CF |
| MAE | Mean Absolute Error |
| MemCF | Memory-Based Collaborative Filtering |
| MF | Matrix Factorization |
| ModelCF | Model-Based Collaborative Filtering |
| MPC | Most-Preferred-Category-based Recommendation |
| METU | Middle East Technical University |
| NDCG | Normalized Discounted Cumulative Gain |
| NYC | New York City |
| PBAR | Popularity-Based Activity Recommendation |
| PBFR | Popularity-Based Friend Recommendation |
| PCF | Preference-based CF |
| PCLAF | Personalized Collaborative Location and Activity Filtering |
| PDA | Personal Digital Assistant |
| POI | Point of Interest |
| POIC | Point of Interest Count based Ranking |
| RMF | Regularized Matrix Factorization |
| RMSE | Root Mean Square Error |
| RPCLAF | Ranking-based Personalized Collaborative Location and Activity Filtering |
| RW | Random Walk |
| RWCAR | Random Walk based Context-aware Activity Recommendation |
| RWCFR | Random Walk based Context-aware Friend Recommendation |
| RWR | Random Walk with Restart |

| | |
|---|---|
| SCAPR | Social-Closeness based Activity Recommendation |
| SCF | Single CF |
| SIAPR | Similar-Interest based Activity Recommendation |
| SVD | Singular Value Decomposition |
| TF-IDF | Term Frequency–Inverse Document Frequency |
| UCF | User-based CF |
| ULACF | Unifying User-Location-Activity CF |
| WCH | Weighted Category Hierarchy |

# CHAPTER 1

# INTRODUCTION

## 1.1 Overview

The advances in mobile communication and pervasiveness of location-acquisition technologies encouraged mobile users to employ their location data with existing social networks in several ways [58]. For example users can share their current location with other users using Foursquare, and share their travel experiences with their friends in GeoLife [60] [59]. The location information helps connecting physical world with social networks. A Location-Based Social Network (LBSN) enables users to add their location information to the social network, hence they can share location-embedded information with other users [57].

The datasets that are resulted from the users' check-in activities through LBSNs can be used for building recommender systems. These datasets contain information about users, locations, activities and relationships between them. The recommender systems may suggest friends, locations and activities to users according to their context. For example, a particular user may want a personalized list of the restaurants in the vicinity. By employing some available data such as user's current location and previous check-in behaviors, his/her friends' check-in data etc., a list of restaurants can be recommended to that user.

In this thesis, we develop algorithms and techniques to build a recommendation system for location-based social networks using a random walk approach. This recommendation system is able to suggest locations, activities and friends to users based on the check-in data. In order to achieve this, firstly LBSN elements (users, loca-

tions and activities) and their relationships are represented by using an undirected unweighted graph model. After building an instance of this graph model, a random walk is performed on that graph to calculate the recommendation probabilities of the nodes. Lists of locations, activities and users are recommended to users after ordering the nodes according to estimated probabilities.

## 1.2 Motivation

Given a LBSN check-in history as a set of tuples where each tuple includes user, check-in time, and check-in location information, and users' social network connections, location recommendation problem for LBSN aims to recommend a set of location for a target user. The input data can be further enriched by additional information such as the type of the activity performed or demographic information of users. Location recommendation problem for LBSNs is an interesting research problem since LBSNs capture both social connections between users and physical interactions between users and locations [50]. Most of the research studies related to this problem are not able to employ these connections and interactions together. Zheng *et al.* have used only GPS trajectories of users to recommend locations [55] [54]. They did not consider social connections between users. There are other approaches that employ multiple criteria for location recommendation. For example, Ye *et al.* introduces an algorithm that considers user preference, social and geographical influence [50]. However, it is not a context-aware algorithm since it does not consider the current location or current social context of users. Moreover, it does not employ other users such as local experts for location recommendation.

In [30], the authors also used a random walk approach to recommend locations. However, their proposed technique only recommend new locations, which are the locations that are not visited in the last month. Moreover, it does not provide recommendations for current context of the user. It suggests general recommendations such as the restaurants in a city for each user. On the other hand, our proposed approach recommends friends, locations and activities considering the current context of the user. Our recommendation technique will traverse a subgraph of the LBSN for personalized recommendation. However, in [30], they traverse the entire LBSN graph to

provide recommendations to each user.

In [27], the authors introduce a Community Location Model (CLM) graph that contains users, locations and activities. Their approach Collaborative Location Recommendation (CLR) employs collaborative filtering techniques and clustering methods to recommend locations to the users. Our proposed approach also use an underlying LBSN graph similar to CLM graph. However, CLM graph does not define the friendship relationships between users. Our recommendation method is based on a random walk approach, and does not employ collaborative filtering techniques and clustering methods like CLR.

In this thesis, we introduce a random walk based context-aware location recommendation algorithm, CLoRW (**C**ontext-aware **Lo**cation Recommendation with **R**andom **W**alk), for LBSNs. CLoRW is able to suggest locations to users based on previous check-in data. Our proposed location recommendation algorithm considers current location, social and personal contexts to suggest locations. Friendship relations of a user defines social context. Similarly, user's previous check-ins present personal context. In addition to this, local experts and popular locations are employed in location recommendation.

There are several location recommendation algorithms in the literature. On the other hand, there are only few number of friend and activity recommendation algorithms. Activity recommendation problem is an interesting research problem, because LBSNs acquire social relations between users, and physical interactions between users, locations and activities [50]. Most of the proposed activity recommendation methods use raw GPS data and they do not consider social connections between users [55] [40]. In [49], Ye *et al.* propose a method to predict the category of next activity, but they cannot recommend a set of activities to user. The approaches proposed in [36] and [46] are related to activity group/partner recommendation, and they are not able to recommend activities for a single user. Moreover, to the best of our knowledge, none of the proposed activity recommendation methods are context-aware and none of them employ local activity experts.

In this thesis, we also introduce a context-aware activity recommendation algorithm, RWCAR (**R**andom **W**alk based **C**ontext-aware **A**ctivity **R**ecommendation), for LB-

SNs using a random walk approach. This algorithm considers social, personal and spatial context of the user. Social context includes the friendship relations of a user. Personal context is built according to user's previous location and activity history. Moreover, local activity experts and popular activities are employed in activity recommendation. Spatial context defines the geographical boundaries of the recommendation.

LBSN data can also be utilized for friend recommendation. LBSNs provide a new way of friend recommendation based on user location histories [5]. User location histories provide valuable contextual information. Moreover, location histories and social behaviors are notably correlated [13]. Friend recommendation is extensively studied for traditional social networks. However, there are a few number of friend recommendation algorithms that employ LBSN data in recommendation. In [59], the authors propose a method that measures the similarity between users based on the location histories. In [11], Chu *et al.* propose a friend recommendation method based on the similar interests of users and location histories. Moreover, in [52], the authors propose a random walk based statistical framework for geo-friends recommendation (GEFR) . All these three methods employ raw GPS data to find the similarity between users. When compared to raw GPS data, check-in data provides more contextual information. Moreover, most of the LBSNs collect check-in data rather than GPS trajectories. The majority of the proposed friend recommendation algorithms are not context-aware algorithms, hence they do not consider the current location of the user. A contextual friend recommendation algorithm can be more useful for finding new friends at visited locations.

In order to suggest new friends to users, we propose a friend recommendation algorithm, RWCFR (**R**andom **W**alk based **C**ontext-aware **F**riend **R**ecommendation). Our friend recommendation algorithm also considers social, personal and spatial context. It employes second degree friends (friends of friends). User's visited locations in recommendation region are also considered to identify place friends. Place friends are potential friends that have check-ins at the locations where the current user visited before. In addition to this, local experts and popular locations are employed in populating the context of the user.

We employ a random walk based approach in all of our algorithms. Collaborative filtering (CF) approaches depend on like-mindedness and similarity of locations. Approaches that employ social filtering make use of friend's data. Spatial filtering only considers physical distance [30]. In order to achieve a higher recommendation accuracy, it is better to employ each of these features together. We define an LBSN graph that models the relationships between users, locations, activities, experts, popular locations and popular activities. Then we employ random walk with restart on this graph to automatically combine personal, social and spatial features. In order to achieve this, firstly LBSN elements (users, locations and activities) and their relationships are represented using a graph model. In this study, we employ an undirected unweighted graph model to represent a particular LBSN. After building the graph, random walk is performed on it to calculate the recommendation probabilities of the nodes. A list of locations, activities or users are recommended to users after ordering the nodes according to estimated probabilities.

## 1.3 Contributions

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

- We propose a model that represents the relationships between LBSN items (i.e. users, locations and activities).
- Three novel recommendation algorithms are proposed based on this LBSN model.
- User preference, social connections and current location are employed together for location, activity and friend recommendation.
- Location experts and popular locations are extracted locally and used in location and friend recommendation.
- Activity experts and popular activities are extracted locally and used in activity recommendation.
- We employ an undirected unweighted graph based on the LBSN model to represent the current context of users. This graph is constructed dynamically ac-

cording to current user's personal, social and spatial context. A random walk is performed on this graph to recommend locations, activities and friends.

- When a new user, location or an activity is inserted into the database, we do not need to update any model or structure, such as tensor. Since our location, activity and friend recommendation algorithms operate locally, we only construct the subgraph for users according to their current context dynamically by employing graph database queries. On the other hand, CF-based and model based recommendation methods need to update the model or tensor periodically.

- We compare the performance of CLoRW with popularity-based, friend-based, expert-based baselines and CF-based approach using Brightkite, Gowalla and Foursquare datasets. Moreover, we compare our algorithm with a well-known location recommendation algorithm, USG [50]. The results of the experiments show that our algorithm outperforms all the baselines, CF-based approach and USG in all of the test cases. This indicates that consideration of location, social and personal context together increases the location recommendation accuracy.

- We compare the performance of RWCAR with popularity-based, friend-based, expert-based baselines using the Foursquare dataset. In all of the experiments RWCAR outperforms all the baselines.

- We compare the performance of RWCFR with popularity-based, friend-based, expert-based baselines using Brightkite, Gowalla and Foursquare datasets. In all of the tests RWCFR performs better than the baselines.

Location recommendation approach[1] described in this thesis has been published in [1]. The paper content is split and reused in corresponding sections of the thesis.

Activity recommendation approach[2] described in this thesis has been published in [2]. The paper content is split and reused in corresponding sections of the thesis.

[1] © 2015 Springer London. Reprinted by permission with license number 3753661370200. Hakan Bagci, Pinar Karagoz, Context-aware location recommendation by using a random walk-based approach, Knowledge and Information Systems, First Online: 15 July 2015, Online ISSN 0219-3116. http://dx.doi.org/10.1007/s10115-015-0857-0.

[2] © 2015 IEEE. Reprinted, with permission, from Hakan Bagci, Pinar Karagoz, Random walk based context-aware activity recommendation for location based social networks, in Data Science and Advanced Analytics (DSAA), 2015 IEEE International Conference on, Oct 2015. http://dx.doi.org/10.1109/DSAA.2015.7344852.

## 1.4 Thesis Organization

The remainder of this thesis is organized as follows. In Chapter 2, an overview of recommendation systems for LBSNs is given. In Chapter 3, we describe our proposed work in detail. The evaluation methodology and experiment results are presented in Chapter 4. Finally, in Chapter 5 we conclude the thesis with comments.

# CHAPTER 2

# RELATED WORK

In this chapter we give a brief overview of the recent work addressing location, activity and friend recommendation in location-based social networks.

## 2.1 Location Recommendation Methods

Most of the location recommendation systems usually use Collaborative Filtering (CF) based methods. On the other hand, decision trees, ontologies, Bayesian networks and random walk methods are also employed in location recommendation. First, we present the CF-based location recommendation methods, then we continue with presenting the other related studies.

### 2.1.1 Collaborative Filtering Based Location Recommendation Methods

Leung *et al.* [27] proposed a Collaborative Location Recommendation (CLR) framework based on co-clustering. In this study, the Community Location Model (CLM) is introduced to organize user, location and activity data into a meaningful data structure. An instance of this model is a User-Acitivity-Location tri-partite graph, called CLM graph. This graph consists of three disjoint node sets respresenting the users, locations and activities. An instance of a CLM graph can be clustered with a co-clustring algorithm. The clustering algorithm outputs clusters of similar users, similar activities and similar locations. The authors claim that CLM exploits all of the spatial properties, temporal-spatial properties, and long-term spatial properties in the

co-clustering process.

The traditional co-clustering algorithms require the CLM graph to be clustered repeatedly when new data are inserted. Therefore, a Community-based Agglomerative-Divisive Clustering (CADC) algorithm is adopted to iteratively cluster different types of entities concurrently based on CLM. When new users, activities and locations arrive, they are added to current CLM graph with their links. The new graph is given as an input to the CADC algorithm. This algorithm groups the nodes into correct clusters by the agglomerative phase. When a particular cluster becomes too large because of the new members, the cluster is partitioned into smaller ones in the divisive phase.

After clustering process, they get the refined clusters of similar locations that are visited by similar users and have similar activities. The clusters are further refined according to user types which are pattern users, normal users and travelers. The users are classified into these categories according to their location entropy values. If a user visits more different places than it has a higher location entropy value. Therefore, the location entropy values get higher values as we move from pattern users to travelers. After refining the clusters by employing user types, the top $k$ locations are recommended to a particular user according to the highest *LF (Location Frequency) X IUF (Inverse User Frequency)* scores in the refined subclusters.

The proposed algorithm is compared against a few state of the art methods. These methods are Distance, SimUser, Memory-Based Collaborative Filtering (MemCF), Model-Based Collaborative Filtering (ModelCF) and HITS (Hypertext Induced Topic Search). MemCF method is based on the approach proposed in [47] and the HITS method is based on the method presented in [61]. Experimental results show that the proposed CADC approach can provide more accurate and refined recommendations according to the existing clustering and collaborative filtering methods. We pay more attention to this study since we plan to build a user-location-activity graph based on the user check-in behaviors in a location-based social network.

Zheng *et al*. proposed a collaborative activity and location recommendation system in [56]. They introduce three collaborative filtering algorithms to address the data sparsity problem in activity and location recommendation. These algorithms are CLAF (Collaborative Location and Activity Filtering) [55], PCLAF (Personalized Collabo-

rative Location and Activity Filtering) [54] and RPCLAF (Ranking-based Personalized Collaborative Location and Activity Filtering).

CLAF algorithm merges all the users' data together and obtains a location-activity matrix. It uses a collective matrix factorization model to provide general recommendations. This algorithm is focused on general recommendations. Therefore, the algorithm outputs the same recommendations to different users. In order to overcome this limitation, they propose a PCLAF algorithm that can provide personalized recommendations to each user. In this algorithm, they directly model the user-activity-location tensor under the factorization framework. The aim of this algorithm is to fill the missing entries in the tensor. In addition to location features and activity correlations that are used in CLAF, they employ user-user similarities and user-location visiting preferences. User-user similarity information is used for obtaining the like-minded users. User-location visiting preferences are useful to model the user preferences on each location.

CLAF and PCLAF algorithms aim to find some model that can minimize the prediction errors with respect to the existing ground truth ratings. In order to rank the recommendations, predictions on the missing values are used. The strategy used in CLAF and PCLAF is an indirect way of solving a ranking problem. However, RPCLAF uses ranking loss as the objective function to solve the recommendation problem directly.

A real-world dataset is employed for evaluating CLAF, PCLAF and RPCLAF algorithms. The dataset contains data from 119 users who carried GPS devices to record their outdoor trajectories. RMSE (Root Mean Square Error) and AUC (Area under the ROC curve) are used as evaluation metrics. RMSE measures the tensor/matrix reconstruction loss on a hold-out test data while AUC measures the ranking results based on the reconstructed tensor from training data. The algorithms are compared with six competing baselines which are UCF (User-based CF), LCF (Location-based CF), ACF (Activity-based CF), ULACF (Unifying User-Location-Activity CF), SCF (Single CF) and POIC (POI Count based ranking). UCF, LCF and ACF memory-based methods are adopted from [21] while SCF is defined in [43]. ULACF is also a memory-based approach and adopted from [47]. The results show that their algo-

rithms generally outperform these baselines. PCLAF has the lowest RMSE values among all the algorithms. Since RPCLAF is a ranking-oriented algorithm its AUC performance is the highest among the others. PCLAF and RPCLAF show better performance than CLAF implying that personalization is an important factor in location and activity recommendation.

In [6] the authors question if the available check-in data of an LBSN has a structure that allows building a recommendation system for the benefit of the users. The check-in dataset is accumulated by crawling Gowalla on a daily basis using its web-based API. As a result of data collection process, 212,000 profiles and their last 20 check-in events are recorded. The dataset includes 1.5 million locations with their coordinates and all related check-in events.

First, in order to define preference definitions, they devise an interpretation of the data. In order to generate a model, they present the data as user-location matrix. The entry of a matrix represents the interest of user $i$ at location $j$. The proposed CF algorithm aims to predict the preference of users for selected locations which they have not visited yet. A list of the top N recommended locations for a particular user can be generated by using the prediction values in the matrix.

In order to estimate the user's interest at a particular location, they mainly focus on two potential preference definitions. The first one is the *binary* preference definition. This definition suggests that if a user's average visits for a location is more often than other users, then the interest value is equal to 1, otherwise 0. The other preference definition employs *equal width intervals* (EWI) to achieve a discrete rating scale with N values. Therefore, they divide the open scale of check-ins for each user to N intervals of equal width. The interest of a user at a particular location hence is the index of the interval which includes the selected number of check-ins.

In recommendation phase, they employ the Regularized Matrix Factorization (RMF) technique for CF. This technique is based on latent factor models. They mapped the users and locations to a joint latent factor space of dimensionality $d$. For a given user $u_i$ and a location $l_k$, the inner product of the corresponding vectors results the preference of the user for that location.

One of the advantages of the approach proposed in [6] is that it does not need to load the entire data in memory at the same time as opposed to memory-based CF techniques. Moreover, RMF recommender has a fast recommendation time. After model learning phase it takes only $O(1)$ to make prediction. On the other hand, it is difficult to configure the parameters of the RMF.

In order to evaluate the proposed recommender, they compare it to an Item Average Recommender. This non-personalized recommender is commonly used as a baseline. The prediction rating for a particular user $u_i$ at location $l_k$ is simply estimated as the average rating of the location over all users. In evaluation phase, they select two specific sub-datasets, namely Austin and New York City, that cover large amounts of users with higher check-in activity. They divide each dataset into a training and a test set. RMF recommender tries to predict ratings for user-location pairs which were not selected for training set. The test set is employed for evaluating the predictive accuracy of the recommender. In order to measure this accuracy, Mean Absolute Error (MAE) and the Root Mean Squared Error (RMSE) metrics are utilized. In the experiments, the RMF recommender performs better than Item Average Recommender for both datasets. Hence, the authors claim that it is possible to make personalized recommendations for locations using CF techniques with the proposed preference definitions.

Papadimitriou *et al.* developed a prototype online recommender system, namely GeoSocial [33]. It aims to provide friend, location and activity recommendations relying on user check-ins. In this study, friend recommendation is based on the FriendLink algorithm presented in [32]. This algorithm constructs a friends similarity matrix where the weigths are the geographical distances between user check-ins. The GeoSocial recommendation engine also constructs a dynamically analyzed 3-order tensor. This tensor is firstly built by the HOSVD (Higher-Order Singular Value Decomposition) algorithm. This tensor is updated incrementally using the methods in [7] [39].

An element of tensor $\hat{\mathrm{A}}$ represents a quadruplet $\{u, l, a, p\}$ where $p$ is the predicted probability of the user $u$ will visit location $l$ and perform activity $a$. $\{u, l\}$ pair gives the activities at a particular location while $\{u, a\}$ pair returns the locations in which

the particular activity can be performed.

When new users, activities or locations are inserted to the database, the $\hat{A}$ tensor needs to be updated. However, the reconstruction of this tensor is a costly operation. Therefore, they employ incremental update solutions. Depending on the size of the update, the selected method varies. The $folding-in$ technique proposed in [39] is suitable for small-scaled updates, while Incremental SVD techniques [7] are appropriate for larger updates.

In experimentations, they use their own dataset that contains 102 users, 46 locations and 18 activities. The number of total check-ins is 1173. In order to evaluate the accuracy of the proposed approach, the precision and recall metrics are employed. *Precision* is the ratio of the number of relevant friends, locations or activities in the top-N list relative to N. *Recall* metric is similar but it is relative to the total number of relevant friends, locations and activities. The results of the experiments indicate that the activity recommendations are more accurate than location recommendations. The friend recommendation accuracy of the FriendLink algorithm is not so high. According to the authors, the reason of this result is data sparsity. We take this study into consideration since it tries to suggest users, locations and activities together. Our approach is also be able to recommend users, locations and activities to users.

In [4] the authors present a location-based and preference-aware recommender system. This system recommends a particular user a set of locations (e.g. restaurants) within a geospatial range considering both user preferences and social opinions.

A user's preferences are learned based on his/her location history which consists of his/her previous check-ins. These preferences are modeled as a weighted category hierarchy (WCH). WCH is employed for estimating the similarity between two users. This similarity measure is used for recommendations based on the fact that similar users tend to visit similar locations. This method handles the data spareness problem for location recommendations by extracting similar users. Social opinions are extracted from the location histories of the local experts. The local expert for each location category is determined in advance by applying an iterative model for social knowledge learning that is based on HITS (Hypertext Induced Topic Search) inference model [9], [25]. This model is based on the fact that people who have visited

many high quality locations in a particular region are more likely to have high knowledge about that region. Moreover, if a particular location is visited by many people that have high knowledge, that location is more likely to be a quality location.

WCH is represented as a tree in which each node is labeled with a value indicating the number of visits of the user to a particular category. This tree is regarded as a document and its nodes are considered as terms of that document for estimating the TF-IDF (Term Frequency–Inverse Document Frequency) values of each node. The intution behind using TF-IDF values is that a user tends to visit more locations belonging to a category that user likes. Moreover, this method reveals the categories of the locations that is often visited by a particular user but rarely visited by other users.

The online recommendation phase of this approach consists of two parts which are preference-aware candidate selection and location rating calculation. In the first part, a set of candidate local experts and locations are selected. The candidate selection algorithm starts from the bottom level of the WCH and moves up to the upper level if the number of candidate locations is not sufficient. In every level, the locations that are visited by local experts are added to the candidate locations set. The algorithm stops when sufficient number of locations are obtained or all the candidate users are evaluated.

In the second part of the recommendation, the similarity score between a particular user and each local expert are calculated depending on their WCHs. The similarity between two WCHs are calculated using the weighted sum of the similarities between each corresponding levels in each hierarchy. The deeper levels of the WCH have greater weights. After calculating the similarity scores, the local experts and candidate locations are placed in a user-location matrix. Then this matrix is used as an input to a user-based CF model that infers users' ratings to candidate locations. The recommender system returns the top-N locations that have the highest scores among candidate locations.

In order to evaluate their proposed approach, the authors used two datasets obtained from Foursquare for New York City (NYC) and Los Angeles (LA). The datasets consist of 221,128 tips generated by 49,062 users in NYC and 104,478 tips generated by

31,544 users in LA. Since Foursquare does not allow to obtain check-in information of users, they employ the tips generated by the users. Each tip contains a location ID, comments and a timestamp.

They compare their approach with three baseline approaches which are Most Preferred Category based (MPC) recommendation, Location based CF (LCF) and Preference based CF (PCF). MCF recommends the top-N locations based on an iterative model similar to [61]. This method does not consider the local experts' opinions. LCF directly applies the traditional user-based CF method to offer locations using a user-location matrix. PCF gathers all the user and locations in a region and constructs a user-location matrix online. It also applies a user-based CF model to infer the locations that will be recommended.

The authors evaluated both the effectiveness of the recommendation results and the efficiency for generating online recommendations. In order to evaluate the effectiveness of recommendations results, they calculate the precision and recall values depending on the ground truths and recommendations. According to the evaluation results the proposed approach outperforms the baseline methods. LCF has the lowest performance among other methods, indicating the advantage of using location categories to model user's location history. MPC has a higher performance than LCF while having a lower performance than PCF. The reason for that could be the lack of consideration of social opinions. PCF outperforms LCF and MPC but the proposed approach has a higher performance because WCH is more capable of modeling user preferences.

In order to evaluate the efficiency for generating online recommendations, they test 200 users in LA and NYC by randomly choosing a location in the city for each user. The fastest method is LCF because it only performs an online selection. Since MPC does not consider the location history of other users, it is faster than the proposed approach. On the other hand, the proposed approach is significantly faster than PCF.

In [50], authors propose a multi-criteria location recommendation algorithm, namely USG. They explore user preference, social influence and geographical influence for location recommendations. User preference and social influence are derived based on user-based collaborative filtering and friend-based collaborative filtering, respectively. In this study, authors put a special emphasis on geographical influence. USG

uses a linear model to fuse user preference to a location with social influence and geographical influence. USG is not a context-aware algorithm since it does not consider the current location or current social context of users. Moreover, it does not employ other users (non-friend users) such as local experts for location recommendation.

### 2.1.2 Model-Based Location Recommendation Methods

In [26], Lee *et al*. proposed a restaurant recommendation method that employs context information and decision treel model. The context information consists of three types of data which are location, personal and environment context. Location context represents the location of the current user in GPS coordinates. Personal context has attributes such as age, gender, marital status, occupation and salary. Environment context information consists of season, weather, temperature, time and user feeling (mood) data.

In this study, three different decision trees are constructed for location, personal and environment context using C4.5 algorithm. Firstly, the recommender system generates three different top-N restaurant recommendation lists for each context. The user preferences are employed for assigning proper weights to each recommendation list. After this weight refinement, the optimal top-N recommendation list is obtained.

The performance of the recommender is evaluated using k-fold cross-validation and Mean Absolute Error (MAE). In order to measure the accuracy of the recommender, three other recommender system models are considered which are LS, LPS and LES. LS considers location context while LPS considers both personal and location contexts. On the other hand, LES employs location and environment contexts. The last system is the proposed recommendation system, abbreviated as LPES, and it considers location, personal and environment contexts together.

In the evaluation phase, the k is set to 10 and the experiments are repeated 10 times. The results of the experiments show that LPES outperforms the remaining recommendation systems. The authors claim that it is a small, but statistically significant improvement in accuracy. They conclude that personal context and environmental context help recommendation system to make more accurate decisions. Moreover,

consideration of user preference leads to different recommendation results for distinct users.

Park *et al*. proposed a map-based personalized location recommendation system that models user's preference by Bayesian Networks (BN) [34]. If BNs are built by an expert, they are not sensitive to changes in the environment. The proposed BN in this study is also built by an expert. However, they employ a parameter learning technique, Expectation Maximization (EM) algorithm, to overcome this problem. Using the collected data, conditional probability tables of the nodes are learned.

They apply their approach on food domain and recommend restaurants to mobile users. In order to model user preferences, they consider name, gender, age, birthday, blood type, car ownership, monthly income, and user's food preferences. In order to suggest restaurants to users, they calculate the preference on restaurants by using the inference results of constructed BN. A restaurant has three attributes, which are class, price and mood. The Equation 2.1 is used for calculating the preference on a restaurant. In order to recommend a list of restaurants, $X_{ijk}$ values are ordered and the top-N list is returned.

$$X_{ijk} = (c_i \cdot w_{class}) + (p_j \cdot w_{price}) + (m_k \cdot w_{mood}) \tag{2.1}$$

In order to evaluate their approach, they employ a dataset collected around Shinchon area. The data are collected by 4 users during a week. The preprocessed dataset consists of context information such as time, user request, user profile, location and weather. Different sets of restaurants are recommended to the users for breakfast, lunch and dinner. The results of the experiments are not compared to another work. Moreover, the employed dataset is not sufficient to validate the stability of the proposed approach. However, this study employs BNs to model the user's preference, which makes it different from other approaches.

In [51], the authors proposed an approach, namely $Urban\ POI-Mine$ ($UPOI\text{-}Mine$), that recommends locations to users according to user preferences and location properties simultaneously. Users' preferences are exploited from their visited locations. The authors noticed that location recommendation systems that are based on the users'

check-in behaviors mostly consider only the social properties of users. Living-sphere of a user consists of social links and check-ins. Therefore, CF-based recommendation methods are restricted by the user's living-sphere. The authors argue that social-links and check-in behaviors itself is not sufficient for recommending locations. In order to recommend locations outside the living-sphere, previously unvisited locations should also be taken into consideration.

In this study the location recommendation problem is modeled using the formula 2.2. $U$ represents the set of user, and $P$ represents the set of locations. The problem of location recommendation is formulated as predicting the relevance score of a given location for each user.

$$f(u, p) \rightarrow v, where\ u \in U, r \in P, and\ v \in [0, 1] \tag{2.2}$$

In order to support location recommendation based on users' preferences and social factors, they train a regression-tree based predictor. It is important to select descriptive features of user-location pairs. In this study the features are extracted in three different aspects, which are Social Factor (SF), Individual Preference (IP) and POI (Point of Interest) Popularity (PP). Features derived from social factor are *CheckSim* and *DisSim*. *CheckSim* feature represents the similarity between users based on their common check-ins. *DisSim* is employed for evaluating the similarity between users according to their relative base-points. Features from individual preference are *Pref* and *HPref*. *CPref* measures the user's personal preference of a location category such as coffee, pizza. The locations are annotated with highlight tags such as coffee, ice-cream, cheese. *HPref* represents the user's preference on a highlight tag. Social factor and individual preference features do not work well for new users. In order to solve this cold start problem, POI popularity feature is employed.

After extracting the features, these features are used as inputs for the location recommendation phase. Since M5Prime has shown excellent performance in similar prediction tasks [15], they choose M5Prime as the relevance score predictor.

In order to evaluate the performance of *UPOI-Mine*, they conduct a series of experiments. Gowalla dataset is employed in these experiments. The Normalized Dis-

counted Cumulative Gain (NDCG) [29] is used to measure the accuracy of the recommended locations. A higher NDCG means that the highly relevant locations have appeared earlier in the recommendation list. NDCG measures the ranking performance but it is also important to identify how close the predictions are to the eventual outcomes. Hence, MAE is also employed as an evaluation metric.

The experiments are conducted in two phases, which are internal and external experiments. Internal experiments compare the performance of their social, location information and user preference factors. On the other hand, external experiments compare the performance of the proposed approach with TrustWalker [22] and CF-based model proposed in [50] according to NDCG and MAE. In internal experiments, they observe that the effect of user preference is higher than other two factors. The results also indicate that user's self check-ins are still more important compared to that of friends. MAE value gets the smallest value when all the factors are combined. CPref feature outperforms other feature if they are individually experimented considering NDCG@10 metric. The results of external experiments show that *UPOI-Mine* outperforms both TrustWalker and CF-based approach. The authors claims that the reason of this result is the consideration of user's self check-in behaviors.

In [41] the authors proposed a location-based context-aware recommendation system namely, I'm feeling LoCo. The aim of this study is to design a complete ubiquitous location-based recommendation system by considering user's location context and similarity measurements. The proposed approach is similar to the study presented in [12], but it does not require users to fill an extensive questionnaire or update their contextual information. Their system mines user's social network profile and extracts the user preferences automatically.

The proposed recommendation algorithm considers the location history of the user (Foursquare check-ins), transportation mode, current location and user's mood. The transportation mode of the user is detected automatically by a method that is similar to the approach proposed in [38]. In order to decide the transportation mode, mobile phone's accelerometer variance and the GPS speed data are employed as inputs to the decision tree. The decision tree selects one of the transportation modes, which are biking, walking and driving, according to these inputs. A series of transportation

modes that are encountered by the decision tree are given as an input to the Hidden Markov Model. Finally, this model selects a single transportation mode based on the presented pattern.

A user's preference model is like a document that consists of series of words. When a user visits a location, its name, category and tags are appended to this document. The recommendation algorithm employs a content based filtering approach to offer locations. The items for content based filtering are the locations visited by the user and their associated information. The algorithm firstly filters the locations according to user current location using a predefined radius. The second filtering is performed using the feeling of the user. The user can choose one of the feelings from artsy, nerdy, hungry, workaholic, party animal, outdoorsy and shopaholic. Each category maps to a location category in Foursqaure. After the locations are filtered by the mood, the associated tags are obtained for each location. For each location a set of words that contains the intersection of the tags of the user and the tags of the particular location are constructed. The log frequency weight of each term in this set of words is calculated using the equation 2.3.

$$f(t) = 1 + log(tf_{t,d}),\ tf_{t,d} > 0 \tag{2.3}$$

$tf_{t,d}$ represents the number of times term $t$ occurs in document $d$ that holds all of the words associated with the user's visited locations. After all of the log frequency weights are estimated for a particular location, the summation of these weights are calculated. This value represents the prediction score of that location. The $k$ places with the highest scores are recommended to the user.

If a user is not active in Foursquare, then this algorithm fails to recommend locations. For this reason, if the user location history is not sufficient for recommendation, they employ a wikitravel page (wikitravel.org) to suggest that city's iconic places and landmarks to the user.

The authors test the usability of the interface of the system. However, the effectiveness of the recommendation system is not evaluated in this study.

### 2.1.3 Ontology-Based Location Recommendation Methods

In [53], Yu *et al.* proposes a location-based recommendation system that supports user participation as collective intelligence. The system consists of three modules, which are user interface module, knowledge base manager module, and inference engine module. A user can view inference results and create/modify the knowledge base and location information through user interface module. Knowledge base manager stores refined information on locations created/modified by users into a knowledge database. Knowledge database keeps information and association rules on locations. The inference engine consists of static data such as ontology, rules, individuals, and facts. The rule based inference engine employs both static data and dynamic data such as current location of the user and user preferences.

The inference engine is implemented by using Bossam [23]. Bossam is a RETE algorithm-based forward chaining inference engine. In order to construct an ontology and the relationship between classes, Protégé ontology editing language is employed. The ontology models the relationships between the profile of the user, location of the user, shops (POIs), and time information. In this study, the presented ontology is constructed for only recommending movie/theater for easy understanding. There are several implemented association rules that depend on the constructed ontology. Association rule for location is defined as 'The shop whose service range overlaps with the search range of the user, will be recommended to the user'. The rule for time indicates that the user will be recommended a movie/theater if he/she can view the movie within his/her available time. The individuals store information about movies, time information of movies, the discount cards of movies, users etc.

In experiments, they make a user to move through Sinchon and check the recommendations on a PDA. The shops are classified into seven types and corresponding ontologies are constructed. They implemented 173 association rules and store them in the knowledge base with 735 individuals and facts. A college student who wants to watch a movie is recommended some movies according to parameters such as his available time, discount card, current location etc. During this recommendation process, the inference rules are executed on the knowledge base and the results are presented to the users. The accuracy of the results are not evaluated in the study. Moreover, the

study does not compare their approach with others. However, the proposed approach makes use of an ontology to recommend locations to the mobile users which makes it sensible within other approaches.

### 2.1.4 Other Location Recommendation Methods

In [30], Noulas *et al.* proposed a new model that depends on personalized random walks over a user-location graph. It combines location visit data with social network data and aims to recommend new unvisited locations to the users. They collect check-in data for 11 cities across the world. The check-in data is collected through Foursquare and Gowalla. They analyzed the collected data and found out that a large fraction of the visited locations are new locations. This fraction changes between 60% and 80% for different cities. Therefore, they claim that it is important to offer high quality recommendations of new locations to the users. The set of new places are identified by analyzing two consecutive months. The equation 2.4 gives the new locations visited by user $i$ in month $t$+1:

$$\Psi_i^t = \Theta_i^{t+1} \setminus \Theta_i^t \tag{2.4}$$

The new location recommendation problem is formally defined as follows: given a check-in dataset for month $t$, a set of users $U$, and a set of locations $L$, the problem is to predict the elements of the set defined in 2.4 which are the new locations that will be visited in next month $t$+1.

In this study, they identify the weak properties of the existing location recommendation methods. They claim that the most of the proposed approaches only capture the unique aspect of the data. For example, CF approaches exploit like-mindedness and location similarity, social filtering approaches only employ friends' data, and spatial filtering methods only consider physical distance. Therefore, the authors aim to develop a network that connects locations and users. After that they want to perform personalized random walks with restart over this network to predict new location recommendations for individual users.

While a random walker jumps across the nodes of the graph according to predefined transition probabilities, it will visit each node at different frequencies. After a while, the random walk will approach a steady-state. Then, the visit count of each node represents the relevance of that node to the start node. Random walk with restart has an additional feature, at any step there is constant probability of random walker to jump back to starting node. Therefore, the nodes that are closer to the start node tend to be ranked more highly than the other nodes, providing a personalized view of the underlying network [45].

In this study, two different graphs are constructed for random walk. The first one is an undirected graph whose nodes are users and locations. A user $i$ is connected to a location $j$ if $c_{ij}$ is not zero which means that user has at least one check-in at $j$. While traversing the graph with random walk, the steady-state probabilities are computed. Then each location is ranked in decreasing order according to these probabilities. The probability of restart keeps the random walker around the user's neighborhood which provides more biased recommendations towards locations that are more connected. The second version of the graph is weighted and directed where each link is given a weight that defines the transition probabilities. A link from user $i$ to user $k$ is weighted according to total number of friends of $i$. The weight of the link from user $i$ to location $j$ is proportional to user's check-ins to that location over the total number of check-ins of that user. The link from location $j$ to user $i$ is weighted using the check-in frequency of the user at that location over the total number of check-ins of that location.

In order to evaluate the proposed approach, the data is partitioned into multiple training/test pairs each of representing two consecutive months for obtaining cross validated results. The prediction algorithms generate a personalized ranked list of locations for each user. The evaluation metrics are precision@N, recall@N and Average Percentile Ranking (APR). They compare their random walk approach to Random, Popular, Activity [35], Home [37], SocialNet, kNN, PlaceNet and Matrix Factorization (MF) predictors. The most important result is that nearly all methods fail to outperform the popularity-based baseline. According to their experiment results, their two approaches are the only ones that outperform popularity. The undirected random walk (RW) achieves an improvement of 5% in Foursquare and 18% in Gowalla

dataset relative to popularity method. The results are similar for all three metrics. Moreover, the results obtained from two different datasets conform with each other.

### 2.1.5 Overview of Location Recommendation Methods

Table 2.1 lists and compares all the previously mentioned location recommendation algorithms in terms of used datasets, context-awareness, evaluation methodology and evaluation metrics.

Table 2.1: Overview of Location Recommendation Methods

| **Method** | **Class** | **Dataset** | **Context Awareness** | **Evaluation Method** | **Metrics** |
|---|---|---|---|---|---|
| CLR | CF-Based | Own Dataset | None | User Feedback | nDCG |
| CLAF | CF-Based | Own Dataset | None | 2-Fold Cross-Validation | Tensor \RMSE, AUC |
| PCLAF | CF-Based | Own Dataset | None | 2-Fold Cross-Validation | Tensor \RMSE, AUC |
| RPCLAF | CF-Based | Own Dataset | None | 2-Fold Cross-Validation | Tensor \RMSE, AUC |
| RMF Spot Recommendation | CF-Based | Gowalla | Social | 2-Fold Cross-Validation | MAE, RMSE |
| GeoSocial | CF-Based | Own Dataset | Social | 4-Fold Cross-Validation | Precision, Recall |

Table 2.1: (Continued)

| **Method** | **Class** | **Dataset** | **Context Awareness** | **Evaluation Method** | **Metrics** |
|---|---|---|---|---|---|
| LPES | Model-Based | Own Dataset | Location, Personal, Environment | 10-Fold Cross-Validation | MAE |
| BN Based Recommendation | Model-Based | Own Dataset | Location, Personal, Environment | 7-Fold Cross-Validation | - |
| UPOI-Mine | Model-Based | Gowalla | Location, Social, Personal | - | nDCG, MAE |
| Ontology Based Recommendation | Ontology-Based | Own Dataset | Location, Personal | - | None |
| Random Walk Around the City | Other | Gowalla, Foursquare | Social | Cross-Validation | Precision, Recall |
| Location-Based and Pref. Aware Rec. | CF-Based | Foursquare | Location, Personal, Social | 2-Fold Cross-Validation | Precision, Recall, Running Time |
| I'm Feeling Loco | Model-Based | Foursquare | Location, Personal | User Feedback | - |
| USG | CF-Based | Foursquare, Whirl | None | 2-Fold Cross-Validation | Precision, Recall |

## 2.2 Activity Recommendation Methods

In this section we give an overview of recent studies addressing activity recommendation in location-based social networks. Most of the works in the literature are focused on location recommendation. There are only limited number of proposed activity recommendation algorithms. However, location and activity recommendation algorithms are closely related. Therefore, we also provide some related work about location recommendation in addition to activity recommendation approaches.

In [27], Leung *et al*. proposed a collaborative location recommendation framework based on co-clustering (CLR). CLR converts users' GPS trajectory data to obtain a set of points of interest as candidate locations. Most of the collaborative filtering based approaches employ User-Location matrix. Unlike these methods, authors propose a Collaborative Location Model (CLM), which considers an additional entity type, activity, to capture the relations between users, location and activities. In CLR, CADC co-clustering algorithm is employed to cluster the CLM graph to produce location recommendations.

Zheng *et al*. propose a collaborative location and activity recommendation approach using users' GPS history data [55]. In addition to GPS data, they also employ point of interest category database and Web, to overcome the data sparsity problem in location-activity relations. Authors perform stay region extraction, location-activity information extraction, location feature extraction and activity-activity correlation during modeling phase. After having location-activity matrix, location-feature matrix and activity-activity correlation matrix, a recommender system is trained for location and activity recommendation. Based on the filled location-activity matrix, they rank the locations and activities and return the top $k$ locations/activities.

Sattari *et al*. inspired from the approach proposed in [55] and introduced a new activity recommendation method in [40]. In this study, they work on the same GPS dataset and employ the same location-activity matrix, location-feature matrix and activity-activity matrix. In both of the studies, the authors try to combine these three matrices to populate an integrated matrix. The basic difference of this study from [55] is the way integrated matrix is used for prediction and the application of Singular Value

Decomposition (SVD). They show that their prediction accuracy is higher than the technique given in [55].

Symeonidis *et al*. realized that the tensor decomposition based approach in [54], namely User Collaborative Location and Activity Filtering (UCLAF), do not update their system online as more users data is inserted into database. In [44], they proposed an online recommender system, called Geo-social recommender system, where users can get user, friend and activity recommendations. In location and activity recommendation, the data is modeled as 3-order tensor. They employ Higher Order Singular Value Decomposition (HOSVD) technique to perform latent semantic analysis and dimensionality reduction on tensor. They also propose an incremental tensor reduction approach to update the system as more data is accumulated to the system.

In [49], Ye *et al*. propose a framework that employs a mixed Hidden Markov Model (HMM) to predict the category of user's next activity. HMM models the dependencies between categories. HMM is trained by incorporating the temporal and spatial information to further improve the model accuracy. Moreover, they model user preferences using users' check-in activities. User preferences are also employed for improving the accuracy of the recommender. After predicting the category of activity, a location is recommended to user according to the estimated category distribution. An advantage of this approach is that it reduces the size of location prediction space by firstly determining the activity category.

The work proposed in [36], Collaborative Group Activity Recommender (CGAR), is different from other approaches, because they produce activity recommendations to groups, instead of individuals. They model the groups as a generative process to obtain the group dynamics. While the authors obtain group dynamics, user interactions are used for learning and inferring group preferences. They also employ location-temporal information of the LBSNs. The activities at a location are modeled using a topic model to capture its semantics. In order to perform the actual group-activity recommendation they use a collaborative filtering framework. They claim that their group generative process and collaborative filtering framework overcome data sparsity and cold-start problem in group recommendation.

W. Tu *et al*. introduce an activity-partner recommendation system in [46]. They realized that when recommending activities to users, it is also important to recommend suitable activity partners. They claim that recommending activity partners is likely to improve the success rate of activity recommendations. They utilize user attendance preference and social context (friendship relations) to implement activity-partner recommendation. They produce three different methods, which are Social-Closeness (SCAPR), Similar-Interest (SIAPR) and Also-Like (ALAPR) based activity recommendation. They also propose another activity-partner recommendation method based on past partner knowledge of users (CFAPR). This method employs collaborative filtering in recommendation. In experiments CFAPR outperforms the other methods proposed in the same work.

## 2.3 Friend Recommendation Methods

In this section we give an overview of friend recommendation methods for LBSNs. Most of the studies in literature are mostly focused on location recommendation. However, there are also a few number of friend recommendation methods for LBSNs. Friend recommendation is extensively studied for traditional social networks. LBSNs provide a new way of friend recommendation based on user location histories [5]. User location histories provide rich contextual information. Moreover, location histories and social behaviors are significantly correlated [13].

In [44], Symeonidis *et al*. propose a prototype system GeoSocial that is able to recommend locations, activities and friends to users. In friend recommendation they employ FriendLink algorithm [32]. In order to calculate the weights between links they use the average geographical distance between users' check-ins. In this paper, the authors state that the performance of FriendLink is not so high due to the data sparsity problem.

GeoLife2.0 is a social networking service that enables users to share their life experiences with other users. In [59], the authors say that GeoLife is capable of measuring the similarity between users based on the location histories. This similarity measure is employed for friend recommendation for individuals. The friend recommendation

system analyzes the location histories of the users and populates the potential friends list. Each individual's location history is modeled using a hierarchical graph based similarity measurement (HGSM).

In [11], Chu *et al*. propose a friend recommendation approach based on the similar interests of the users. Moreover, they employ the real-life location and dwell time in friend recommendation. After gathering these data, the proposed method analyzes the data using weighted Voronoi diagram and interest similarity. Affinity diagrams and matrices are populated according to the results of this analysis. Based on the affinity matrices and interest similarity the acceptable degree value is evaluated for candidate friends. If this value is greater than the predefined threshold for a user, then that user is recommended as a friend.

In [52], the authors propose a random walk based statistical framework for geo-friends recommendation (GEFR). It is a three-step approach and first, raw GPS data is analyzed and interesting and discriminative GPS patterns are extracted. The extracted geographical information and social network are combined in a heterogeneous information network. Random walk is applied on this network to provide friend recommendations. GEFR employs the patterns that are extracted from raw GPS data. However, actual check-in data can be more useful for identifying similar users in a social network rather than using the GPS patterns. Moreover, GEFR is not a context-aware algorithm and does not consider the current location of the user while recommending friends.

In [28], Li *et al*. introduce a three-layered friendship model that is used to evaluate the similarity between users in LBSNs. They employ social connections, user profiles and mobility patterns to find the correlation between users. The three layers are social graph layer, tag graph layer and location graph layer. In each layer the similarity of users are calculated based on the respective formula. The global friendship ranking is calculated by summing the ranking scores for each layer.

The work proposed in [42] is for recommending new friend links to the users. In this paper, the authors study to reduce the size of the prediction space. They show that even making the prediction space 15 times smaller by utilizing the place-friends, %66 of the future friend connections can still be preserved. The aim of the study is

to design a friend link prediction system which exploits data about user check-ins. The authors emphasizes the *sociological focus theory* which identifies that activity and interaction happening around physical places can produce social relationships between individuals, and it is correlated to the properties of the place itself [16]. In the experiments, they compare the performance of J48, Naive Bayes, model trees with linear regression on the leaves [17], and random forests [8]. The results indicates that their prediction method is more effective with random forests and model trees methods. The results also show that exploiting the place features in friend link recommendation leads to better performance than the methods that are purely based on social features.

Friendbook is a semantic-based friend recommendation system for social networks [48]. It employs user's life style information that is extracted from the data gathered from smartphone sensors. The authors model the daily lives of users as life documents. In order to extract life style information of users, they use the probabilistic topic model. In this topic model, the life styles are modeled as topics and the activities of users are modeled as words. Friendbook recommendation system is not proposed for LBSNs, it is designed for all types of social networks. However, it is capable of using user's current location to filter the friend recommendation list. Therefore, it can be regarded as a context-aware friend recommendation algorithm. On the other hand, it does not employ location history of the users in estimating the correlation between users.

# CHAPTER 3

# PROPOSED WORK

In this chapter, we describe the details of our recommendation algorithms. Since random walk is employed in all of the algorithms, firstly, the details of random walk is given in Section 3.1. After that LBSN data model is introduced in Section 3.2. Then location, activity and friend recommendation algorithms are described in detail in Section 3.3.

## 3.1 Random Walk

Random walk can be used to rank the nodes on a graph using the information encoded by links [30]. Random walk starts from a particular node and moves through the edges of the graph. This process is performed according to transition probabilities, which define the probability of moving from any node to another. In each transition of the random walk, the visit count of current node is incremented to be used for ranking the nodes of the graph. Random walk process continues for a while and approaches a steady state. The output of random walk is the visit counts of the nodes. This information can be utilized for ranking the nodes.

If graph has many nodes, then a random walk may move away from the starting node rapidly. This may cause moving out of the context and visiting less important nodes. This problem can be solved using a derivative of random walk, namely random walk with restart (RWR). In this method, in each transition, there is a constant probability of jumping back to starting node. Therefore, nodes that are closer to starting node tends to have more visits than distant nodes. RWR is a well-known method that

provides a good relevance score between two nodes in a graph. Therefore, RWR and related methods have been successfully used in numerous studies [45], such as automatic captioning of images, generalizations to the connection graphs, PageRank [31], personalized PageRank [20], SimRank [24], ObjectRank [3], RelationalRank [19] and more.

Formally, the transition probabilities of a random walk can be arranged in a matrix as in Equation 3.1. In this equation, $W$ is a matrix that represents the transition probabilities according to graph structure. $R$ is for modeling the random probability of moving back to starting node. $\alpha$ value is used for determining the weight between $W$ and $R$ to calculate the resultant matrix $Q$. Therefore, random walk behavior can be tuned by changing the $\alpha$ value.

$$Q = \alpha W + (1 - \alpha)R \tag{3.1}$$

$$p = pQ \tag{3.2}$$

In order to calculate the steady state rankings of the nodes, Equation 3.2 must be solved. Here $p$ is a vector of steady state probabilities, where $p_i$ denotes the $i^{th}$ node's probability. Resultant $p$ can be calculated by repeatedly iterating over Equation 3.2 until it converges [30].

In order to make personalized recommendations to users, we employ RWR on the underlying graph of the LBSN. The starting node of the random walk is the user that requests location, activity or friend recommendation. Our RWR approach is a specialized version of original RWR, since we have uniform transition probabilities. In other words, the probabilities of moving to each neighbor from a particular node is equal. This assumption lowers the computation cost, because it is a costly task to assign weights to each link on the graph. In this version of RWR, when new users and locations arrive, we do not need to update any weights. This is the major weakness of CF-Based recommendation methods, because such approaches need periodic updates. Moreover, our approach is more extensible since different types of nodes can easily be adopted into the same graph.

During graph traversal in every visit, current node's visit count is incremented by 1. Since it is an RWR, in every transition it is possible to jump back to starting node. After, RWR converges, we sort the locations, activities or users by visit counts and produce the recommendation results.

### 3.2 LBSN Data Model

LBSN data basically consists of users, locations, activities and relationship data. Representation of our LBSN model is given in Figure 3.1. An instance of this model is an undirected unweighted graph that defines the relationships between users, locations and activities. Formally, this graph, $G$, is a tuple $G < V, E >$ where $V$ is a set of nodes $v$ and $E$ is a set of edges $e$. $V = U \cup L \cup A$ where $U$, $L$ and $A$ are disjoint sets and $U$ is a set of users, $L$ is a set of locations and $A$ is a set of activities.

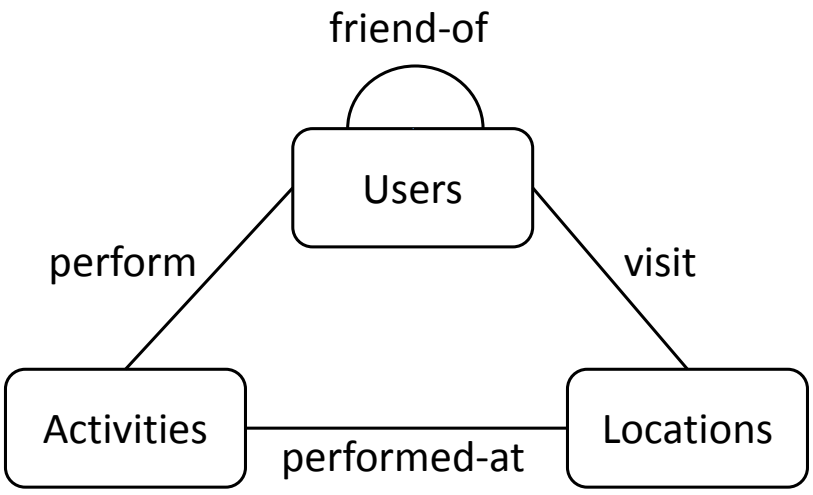



Figure 3.1: LBSN Data Model

A particular user is an ordinary user or an expert. Experts are users that have more location/activity knowledge about a particular region. Similarly, there are two different types of locations/activities which are ordinary and popular location/activities. Popular locations/activities and experts are identified together using a HITS-based [9] [25] algorithm. The details of this algorithm is given in Section 3.3.1.2 and Section

3.3.2.2.

$E$ is a set of edges that represent relationships between the elements of $V$. There are four types of relationships as depicted in Figure 3.1:

- **friend-of**: There is an edge between two users if they are friends.
- **visit**: There is an edge between a user and location if the user has visited the location at least once.
- **perform**: There is an edge between a user and an activity, when the user has performed the activity at least once.
- **performed-at**: There is an edge between an activity and a location if the activity could be performed at that location.

We propose two different location recommendation algorithms. In the first location recommendation algorithm, our LSBN model does not contain activity nodes. Therefore, in this algorithm location recommendation graphs do not contain perform and performed-at relationships. However, in the second algorithm, activity nodes are included in the model. Our activity recommendation algorithm considers all types of relations in this model. On the other hand, our friend recommendation algorithm considers visit and friend-of links.

## 3.3 Proposed Approach

We propose three different novel recommendation algorithms for LBSNs. The first one is a location recommendation algorithm (CLoRW). There is another version of CLoRW, namely CLoRW_A, which additionally considers activities. The second algorithm is an activity recommendation algorithm (RWCAR). The last one is a friend recommendation algorithm (RWCFR). The details of these algorithms are given in the following sections.

### 3.3.1 Context-Aware Location Recommendation with Random Walk

In this section, we describe our proposed location recommendation algorithm in detail. CLoRW, is a context-aware location recommendation algorithm that considers personal, social and location (spatial) contexts. CLoRW consists of two phases, which are subgraph construction and location recommendation phases.

In our random walk approach, our graph traversal is not performed on the entire graph that represents whole LBSN. Instead of this, we construct a subgraph according to current context of the user that requests location recommendation. The details of subgraph construction is given in Section 3.3.1.2. After subgraph construction phase we perform the actual recommendation using random walk. The details of recommendation phase is given in Section 3.3.1.3. Before explaining the details of the algorithm we define the location recommendation problem.

#### 3.3.1.1 Location Recommendation Problem

Location recommendation problem can be formulated as follows: Let graph $G$ be an instance of the LBSN model introduced in Section 3.2. Given $G$ and user's current location, for each user $u$ in $U$, we aim to predict the future locations that can be visited within a predefined radius $r$. These locations can be represented as a recommendation list which is an ordered list of locations that contains $N$ elements of $A$ where $N$ is the desired number of recommendations. The challenge is to populate this location list with highest accuracy, in other words, with minimal errors relative to user's future visited locations in $r$.

#### 3.3.1.2 Subgraph Construction

In subgraph construction phase, a subgraph is constructed for a particular user using his/her current context. In the second phase a random walk is performed on this graph to sort the location nodes for recommendation. When a particular user requests recommendations at a specific location, CLoRW firstly constructs a subgraph using the following items:

- locations that user previously visited in vicinity (personal spatial context)
- friends and their previously visited locations in vicinity (social spatial context)
- experts and their previously visited popular locations in vicinity (social spatial context)

The details of user subgraph construction algorithm is given in Algorithm 1. In this algorithm, *userid* and *currentLoc* represent the id of the user and the current location of the user, respectively. We model the vicinity as a rectangular region. In order to define this region we employ the *radius* parameter. A circle is drawn from user's current location and the bounding rectangle of this circle is used as recommendation region. We call this region as *vicinity*. *GetUserLocationsInVicinity* procedure retrieves the previous check-ins of the currents user in this region. Similary, *GetFriendLocationsInVicinity* procedure selects the friends that have check-ins in this region and their locations. *GetExpertLocationsInVicinity* procedure finds the experts and popular locations in recommendation region. After experts are identified, the relationship between all users and experts are retrieved using the *GetFriendRelationships* procedure.

*expertCount* and *popLocCount* parameters limit the number of expert and popular location counts in the subgraph, respectively. In order to obtain experts and popular locations, similar to works in [4][61], we employ a HITS-based [9][25] algorithm in which locations are authority and users are hub nodes. Each user's hub score denotes the knowledge of that user in recommendation area. Similarly, authority score of a location represents the popularity of that location in this area. According to HITS, people who visit many high quality locations in a region have rich knowledge about the region. Similarly, if a location is visited by many experts, it is more likely to be a quality location [4]. In order to calculate the location and user scores in expert and popular location estimation, our HITS-based algorithm iterates up to a predefined count. In each iteration, location scores are updated according to previously calculated user scores. In the same iteration newly obtained location scores are employed for estimating the user scores. At the end of each iteration, location and user scores are normalized. When the iteration count is reached, location and user scores are sorted. Our location recommendation algorithm selects the desired number of experts

**Algorithm 1** Subgraph Construction Algorithm for Location Recommendation

1: Initialize $G<V,E>$ {Subgraph of the user}
2: $userLocs \leftarrow GetUserLocationsInVicinity()$
3: $friendLocs \leftarrow GetFriendLocationsInVicinity()$
4: $expertLocs \leftarrow GetExpertLocationsInVicinity()$
5: Create new user vertex $u$ for the current user
6: $V \leftarrow V \cup u$
7: **for all** $loc$ in $userLocs$ **do**
8: $v_l \leftarrow$ new location vertex for $loc$
9: $V \leftarrow V \cup v_l$
10: **end for**
11: **for all** $friendLoc$ in $friendLocs$ **do**
12: $v_f \leftarrow$ new user vertex for $friendLoc.user$
13: $v_l \leftarrow$ new location vertex for $friendLoc.location$
14: $V \leftarrow V \cup \{v_f, v_l\}$
15: $e_f \leftarrow$ new edge between $u$ and $v_f$
16: $e_l \leftarrow$ new edge between $v_f$ and $v_l$
17: $E \leftarrow E \cup \{e_f, e_l\}$
18: **end for**
19: **for all** $expertLoc$ in $expertLocs$ **do**
20: $v_f \leftarrow$ new user vertex for $expertLoc.user$
21: $v_l \leftarrow$ new location vertex for $expertLoc.location$
22: $V \leftarrow V \cup \{v_f, v_l\}$
23: $e_f \leftarrow$ new edge between $u$ and $v_f$
24: $e_l \leftarrow$ new edge between $v_f$ and $v_l$
25: $E \leftarrow E \cup \{e_f, e_l\}$
26: **end for**
27: $V_f \leftarrow$ friends and experts vertices
28: $friendships \leftarrow GetFriendRelationships(V_f)$
29: **for all** $friendship$ in $friendships$ **do**
30: $e_f \leftarrow$ new edge between $v_{friendship.f_1}$ and $v_{friendship.f_2}$
31: $E \leftarrow E \cup \{e_f\}$
32: **end for**

and popular locations from these sorted lists. Since we filter whole LBSN graph and construct a subgraph, the number of users and locations in the vicinity are not so high. Since HITS is a rapidly converging algorithm and the size of the bi-partite graph is small, we do not have a convergence problem in our experiments. We use a constant value (i.e. 10) for iteration count.

#### 3.3.1.3 Recommendation using RWR

After subgraph of a given user is constructed by using Algorithm 1, the output of this algorithm is given as an input to Algorithm 2. Algorithm 2 performs the actual location recommendation using random walk. In this algorithm, *recCount* denotes the requested number of recommendations. *iterCount* and *restartProb* represent random walk iteration count and restart probability at each move. *curNode* stores the node currently being visited by the random walk algorithm. In every iteration *curNode* changes and the visit count of that node is incremented by 1. When random walk iteration count is reached the algorithm sorts the nodes by visit count. Then it selects the first *recCount* locations from the sorted list and terminates.

Since our LBSN graph is an unweighted graph, transition probabilities are equal for each node in every random walk iteration. *curNode* selects whether to move to starting node or its neighbors. All of its neighbors have the same chance to be the next *curNode*. *restartProb* parameter corresponds to the $\alpha$ value that determines the restart behavior of the random walk.

An example run of the algorithm for New York City (NYC) Brightkite dataset is given in Figure 3.2. In this example, the whole Brightkite dataset is filtered and only the locations around the NYC are considered. Here, *User_1* requests a personalized location recommendation list at a particular point. The configuration of the run is given in Table 3.1.

The result of the example run is listed in Table 3.2. The recommended locations are sorted according to their visit counts. *PopularLoc_1* has the highest visit count and recommended in the first place. On the other hand, the visit count of *PopularLoc_3* is 0 although it is a popular location. The reason of this situation is that it is not

**Algorithm 2** Random Walk Recommendation Algorithm

1: $recCount \leftarrow$ number of requested recommendations
2: $iterCount \leftarrow$ iteration count of random walk
3: $restartProb \leftarrow$ probability of restart in each move
4: $G < V, E > \leftarrow$ subgraph of the user
5: $u \leftarrow$ vertex of the current user in $V$
6: $curNode \leftarrow u$
7: **for** $i < iterCount$ **do**
8: $p \leftarrow$ rand(0,1)
9: **if** $p < restartProb$ **then**
10: $curNode \leftarrow u$
11: **else**
12: $curNode \leftarrow SelectNextNode(curNode)$
13: $curNode.visitCount \leftarrow curNode.visitCount + 1$
14: **end if**
15: $i \leftarrow i + 1$
16: **end for**
17: $sortedNodes \leftarrow SortNodesByVisitCount(G<V,E>)$
18: $result \leftarrow SelectFirstKNodes(sortedNodes, recCount)$

Table 3.1: Location Recommendation Configuration for a Given User

| **Parameter** | **Value** |
|---|---|
| Current Location Latitutude | 40.7772483825684 |
| Current Location Longitude | 73.8726119995117 |
| Radius | 3000 m |
| Expert Count | 3 |
| Popular Location Count | 3 |
| DBSCAN Eps | 3000 m |
| DBSCAN Minimum Cluster Points | 1 |
| Random Walk Restart Probability | 0.05 |
| Random Walk Iteration Count | 10000 |

Table 3.2: Location Recommendation Results

| **Location** | **VisitCount** |
|---|---|
| PopularLoc_1 | 594 |
| Location_3 | 263 |
| Location_4 | 206 |
| PopularLoc_2 | 199 |
| Location_6 | 199 |
| Location_2 | 190 |
| Location_1 | 179 |
| Location_5 | 153 |
| PopularLoc_3 | 0 |

connected to a user or an expert in the subgraph.


Figure 3.2: Sample Graph for Location Recommendation

#### 3.3.1.4 Context-Aware Location Recommendation with Random Walk using Activities

In the previous section, we introduce our location recommendation algorithm, namely CLoRW. In this section we describe a different version of CLoRW, **C**ontext-Aware

**Lo**cation Recommendation with **R**andom **W**alk using **A**ctivities (CLoRW_A), in detail. CLoRW_A is an extended version of CLoRW. CLoRW algorithm is a context-aware location recommendation algorithm that considers personal, social and spatial contexts. In addition to this, CLoRW_A takes activities into consideration. In our model, activity is used a as a general term that defines actions, categories, tags that are related to a user or a location. In most of the popular LBSNs, locations have one or more categories (e.g.French Restaurant, Coffee Shop). A particular user or a location can be related to one or more categories. These categories correspond to activity nodes in our LBSN model. When a particular user requests recommendations at a specific location, CLoRW_A firstly constructs a subgraph using the following items:

- Locations that user previously visited in vicinity
- Friends and their previously visited locations in vicinity
- Experts and their previously visited popular locations in vicinity
- Activities that user performed before
- Activities that can be performed in vicinity locations
- Vicinity locations that are related to an activity that the current user performed before

When all the related nodes are connected, random walk is performed on the subgraph as in the previous section. Similarly, when random walk iteration count is reached the algorithm sorts the nodes by visit count. Then it selects the desired number of locations from the sorted list and terminates.

#### 3.3.1.5 Context-Aware Location Recommendation with Weighted Random Walk

CLoRW employs an undirected unweighted graph model to recommend locations. We also try to develop the weighted version of CLoRW, **C**ontext-Aware **Lo**cation Recommendation with **W**eighted **R**andom **W**alk (CLoWRW). CLoWRW is identical to CLoRW except the weighting strategy between user-location edges. In CLoRW, there is a single edge between a user and a location independent from the number

of check-ins to that location if that user has at least one check-in at that location. However, in CLoWRW the number of edges between a user and a location is equal to number of check-ins. In other words, we assign the check-in count as weight to user-location edges.

We also try two different parameters for assigning weights to user-location edges. These parameters are visit count of locations and temporal check-in rate. When the visit count parameter is considered, if we want to select a neighbor location from a user node, the location nodes that have higher visit counts have higher probability of becoming the next node in next iteration. On the other hand, temporal check-in rate parameter pays more attention to recent check-ins rather than the older ones. This means if a location has more recent check-ins then it is more probable for user to select that location in next iteration.

In addition to these parameters, we also employ two different strategies for next node selection. In the first strategy, all neighbor nodes have equal probability to participate to node selection. In the second strategy, first the node type is selected with equal probability and then the next node is selected from the nodes that have the selected type. Finally, we have two different parameters and two different strategies, hence four different cases.

### 3.3.2 Context-Aware Activity Recommendation with Random Walk

In this section, we describe our proposed activity recommendation algorithm in detail. RWCAR is a context-aware activity recommendation algorithm that employs RWR for prediction of activity ranks. RWCAR considers three different contexts which are personal, social and location (spatial).

In our random walk approach, our graph traversal is not performed on the entire graph that represents whole LBSN. Instead of this, we construct a subgraph according to current context of the user that requests activity recommendation. The details of subgraph construction is given in Section 3.3.2.2. After subgraph construction phase we perform the actual recommendation using random walk. The details of recommendation phase is given in Section 3.3.2.3. Before explaining the details of

RWCAR, we define the activity recommendation problem.

#### 3.3.2.1 Activity Recommendation Problem

Activity recommendation problem can be formulated as follows: Let graph $G$ be an instance of the LBSN model introduced in Section 3.2. Given $G$ and user's current location, for each user $u$ in $U$, we aim to predict the future activities that can be performed within a predefined radius $r$. These activities can be represented as a recommendation list which is an ordered list of activities that contains $N$ elements of $A$ where $N$ is the desired number of recommendations. The challenge is to populate this activity list with highest accuracy, in other words, with minimal errors relative to user's future performed activities in $r$.

#### 3.3.2.2 Subgraph Construction

In order to perform RWR, we need a subgraph that is built according to user's current context. Here, user is the person requesting activity recommendation. This graph is constructed using the following items:

- Previously visited locations of user in vicinity (personal spatial context)
- Previously performed activities of user (personal context)
- Friends and their previously visited locations and performed activities in vicinity (social spatial context)
- Experts and their previously performed popular activities in vicinity (social spatial context)
- Locations in vicinity that activities added to the subgraph can be performed at (spatial context)
- Activities that can be performed at locations that are added to the subgraph (spatial context)

The links between added nodes are set based on the LBSN model introduced in Section 3.2. Additionally, current user and experts are connected using friend-of relationships. Vicinity is bounded by a rectangular region based on a predefined radius. A circle is drawn from user's current location. The bounding rectangular region of that circle is employed as recommendation region. Vicinity refers to that rectangular region. Algorithm 3 presents the details of subgraph construction process.

In this algorithm, *GetUserActivities* procedure retrieves the previously performed activities of the current user. *GetUserLocationsInVicinity* method is used for retrieval of vicinity locations that the user previously checked-in. Similarly, *GetFriendLocationsInVicinity* selects the friends' vicinity locations. *GetExpertActivitiesInVicinity* procedure finds the experts and popular activities in recommendation region.

In order to find experts and popular activities, we apply a HITS-based [9][25] algorithm. In this algorithm, users are hub nodes and activities are authority nodes. Each user's hub score represents the activity knowledge of that user in vicinity. On the other hand, each activity's authority score denotes the popularity of that activity in recommendation region. The main idea behind this approach is that users who perform high quality activities in vicinity, have rich knowledge about the vicinity. Equivalently, if an activity is performed by many experts, then it is more probable for that activity to be a quality activity. A similar approach is introduced in [4][61] for location recommendation.

In order to find hub and authority scores, our algorithm iterates until it converges. In each iteration, activity scores are calculated depending on the previous iteration's user scores. In the same iteration, this activity scores are employed for predicting user scores. Before moving to next iteration, calculated activity and user scores are normalized. This process continues up to convergence of the algorithm. After that, activity scores and user scores are sorted to find out desired number of popular activities and experts. Since we employ filtered subgraphs in recommendation, the number of activities and users that our algorithm considers are not so high. Therefore, the size of the HITS graph is small and algorithm converges rapidly.

**Algorithm 3** Subgraph Construction Algorithm for Activity Recommendation

Initialize $G<V,E>$ {Subgraph of the user}
$userActs \leftarrow GetUserActivities()$
$userLocs \leftarrow GetUserLocationsInVicinity()$
$friendLocs \leftarrow GetFriendLocationsInVicinity()$
$expertActs \leftarrow GetExpertActivitiesInVicinity()$
Create new user vertex $u$ for the current user
$V \leftarrow V \cup u$
**for all** $act$ in $userActs$ **do**
    $v_a \leftarrow$ new activity vertex for $act$
    $V \leftarrow V \cup v_a$
**end for**
**for all** $loc$ in $userLocs$ **do**
    $v_l \leftarrow$ new location vertex for $loc$
    $V \leftarrow V \cup v_l$
**end for**
**for all** $friendLoc$ in $friendLocs$ **do**
    $v_f \leftarrow$ new user vertex for $friendLoc.user$
    $v_l \leftarrow$ new location vertex for $friendLoc.location$
    $V \leftarrow V \cup \{v_f, v_l\}$
    $e_f \leftarrow$ new edge between $u$ and $v_f$
    $e_l \leftarrow$ new edge between $v_f$ and $v_l$
    $E \leftarrow E \cup \{e_f, e_l\}$
**end for**
**for all** $expertAct$ in $expertActs$ **do**
    $v_f \leftarrow$ new user vertex for $expertAct.user$
    $v_a \leftarrow$ new activity vertex for $expertAct.activity$
    $V \leftarrow V \cup \{v_f, v_a\}$
    $e_f \leftarrow$ new edge between $u$ and $v_f$
    $e_a \leftarrow$ new edge between $v_f$ and $v_a$
    $E \leftarrow E \cup \{e_f, e_a\}$
**end for**

**Algorithm 4** Subgraph Construction Algorithm cont'd

$V_l \leftarrow$ all location vertices
**for all** $v_l$ in $V_l$ **do**
    $locActs \leftarrow GetLocationActivities(v_l)$
    **for all** *locAct* in $locActs$ **do**
        $v_a \leftarrow$ new activity vertex for *locAct*
        $V \leftarrow V \cup v_a$
        $e_a \leftarrow$ new edge between $v_l$ and $v_a$
        $E \leftarrow E \cup \{e_a\}$
    **end for**
**end for**
$V_a \leftarrow$ all activity vertices
**for all** $v_a$ in $V_a$ **do**
    $actLocs \leftarrow GetActivityLocations(v_a)$
    **for all** *actLoc* in $actLocs$ **do**
        $v_l \leftarrow$ new location vertex for *actLoc*
        $V \leftarrow V \cup v_l$
        $e_l \leftarrow$ new edge between $v_a$ and $v_l$
        $E \leftarrow E \cup \{e_l\}$
    **end for**
**end for**
$V_f \leftarrow$ friends and experts vertices
$friendships \leftarrow GetFriendRelationships(V_f)$
**for all** $friendship$ in $friendships$ **do**
    $e_f \leftarrow$ new edge between $v_{friendship.f_1}$ and $v_{friendship.f_2}$
    $E \leftarrow E \cup \{e_f\}$
**end for**

#### 3.3.2.3 Recommendation using RWR

After subgraph is constructed using the procedure that is explained in detail in Section 3.3.2.2, that subgraph is given as an input to the activity recommendation algorithm. RWCAR operates on this graph using RWR to rank activities. Random walk starts from the current user's node and iterates over the graph. When an activity node is encountered the visit count of that node is incremented by 1. At the end of the random walk process, the activity nodes are sorted according to number of visits. After that recommendation results are populated depending on the number of desired number of recommendations. More details about random walk algorithm is given in Algorithm 2.

An example subgraph is given in Figure 3.3. In this example, *User_1* requests activity recommendation at a specific location. A graph is populated by using the subgraph construction algorithm. This algorithm identifies *Expert_1* and *Expert_2* as local activity experts. Moreover, user's location and activity history in vicinity is added onto graph. In addition to this, *Friend_1* and *Friend_2* and related links are added.

After building the subgraph, it is given as an input to actual activity recommendation algorithm. This algorithm performs a random walk on that subgraph and populates the results shown in Table 3.3.

Table 3.3: Activity Recommendation Results

| **Activity** | **Visit Count** |
|---|---|
| Activity_1 | 140 |
| Activity_2 | 115 |
| Activity_4 | 97 |
| Activity_3 | 50 |
| Activity_5 | 47 |

### 3.3.3 Context-Aware Friend Recommendation with Random Walk

In this section, we describe our friend recommendation algorithm in detail. RWCFR is a context-aware friend recommendation algorithm that considers personal, social and spatial context. It employs RWR for estimating the user rankings for friend rec-

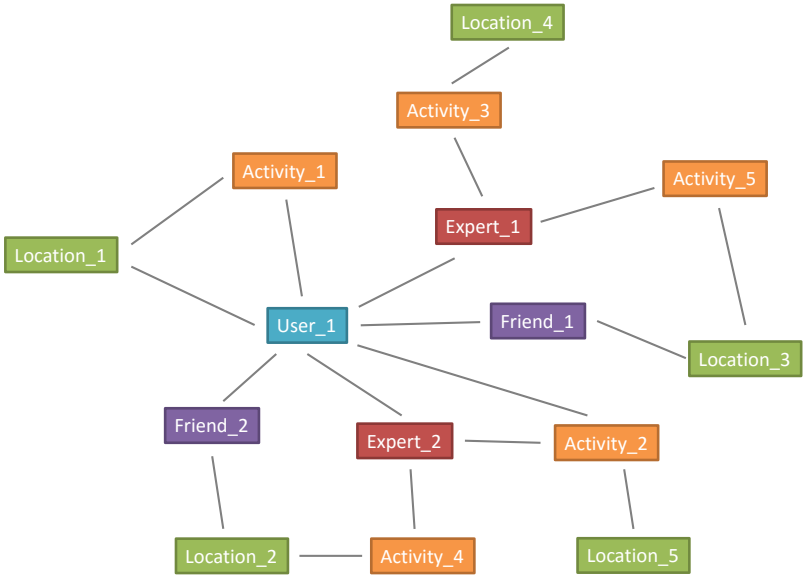



Figure 3.3: Sample Graph for Activity Recommendation

ommendation.

First a subgraph is constructed for the user that requests friend recommendation according to his/her current context. The details of construction of this subgraph is explained in Section 3.3.3.2. RWR is performed on this subgraph to rank the candidate friends for the current user. The actual friend recommendation is done according to these rankings. The details of recommendation phase is given in Section 3.3.3.3. Before describing the details of RWCFR, we define the friend recommendation problem.

#### 3.3.3.1 Friend Recommendation Problem

Friend recommendation problem can be formulated as follows: Let graph $G$ be an instance of the LBSN model introduced in Section 3.2. Given $G$ and user's current location, for each user $u$ in $U$, we aim to predict the future friends. These users can be represented as a recommendation list which is an ordered list of users that contains $N$ elements of $U$ where $N$ is the desired number of recommendations. The challenge

is to populate this user list with highest accuracy, in other words, with minimal errors relative to user's future friends.

#### 3.3.3.2 Subgraph Construction

In order to recommend friends to user, first we build a subgraph according to user's current context. This graph is constructed using the following items:

- Previously visited locations of user in vicinity (personal spatial context)
- Friends and their previously visited locations in vicinity (social spatial context)
- Experts and their previously visited popular locations in vicinity (social spatial context)
- Friends of friends (social context)
- Users that visited locations that the current user previously visited (social spatial context)

This subgraph is an instance of the LBSN data model introduced in Section 3.2. Vicinity is defined by a rectangular region based on a constant radius. Vicinity is also called as recommendation region. The subgraph construction procedure for friend recommendation is given in Algorithm 5.

*GetUserLocationsInVicinity* procedure retrieves the previously visited locations of the user in vicinity. Similarly, *GetFriendLocationsInVicinity* method is used for retrieval of the locations of friends in recommendation region. *GetExpertLocationsInVicinity* procedure identifies the local experts and popular locations. The process of finding experts and popular locations is identical with the process that is described in Section 3.3.1.2.

*GetFriendsOfFriends* method is used for finding the second degree friends of the current user. Since friends of friends of the current user could be considered as potential future friends, we add these users to our subgraph.

The users visiting the same places are likely to become friends. Therefore, the users that have check-ins at the locations that the current user previously visited are also potential future friends of the current user. *GetUsersThatVisitedLocation* procedure retrieves the users that checked-in at a location that the current user previously visited. This procedure is called for each vicinity location that the current user checked-in before. New users are added to the graph as vertices and links are constructed between these users and corresponding locations.

#### 3.3.3.3 Recommendation using RWR

After constructing the subgraph for friend recommendation, we employ RWR to rank potential friends. The details about RWR is provided in Algorithm 2. Therefore, we do not provide more details here.

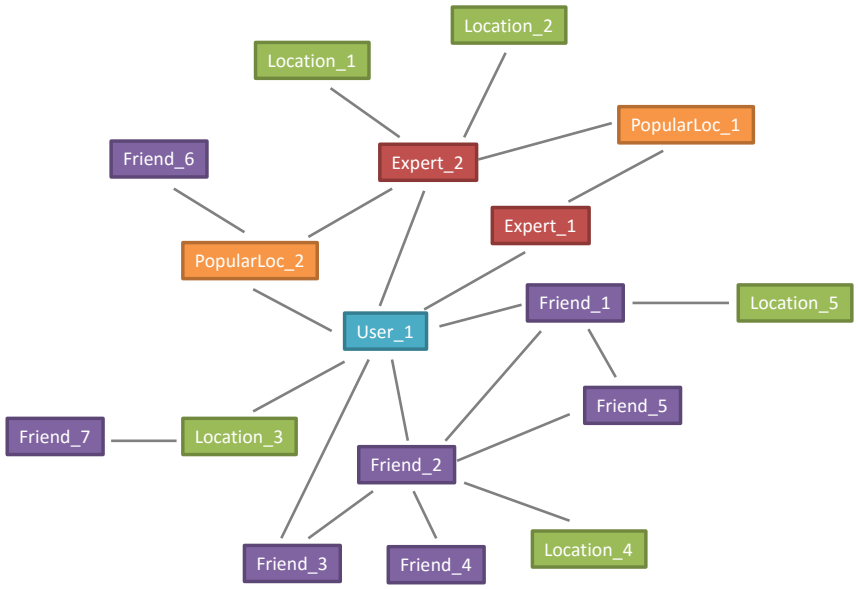



Figure 3.4: Sample Graph for Friend Recommendation

An example friend recommendation subgraph is fiven in Figure 3.4. Here, *User_1* requests friend recommendation at a particular location. This subgraph is constructed

**Algorithm 5** Subgraph Construction Algorithm for Friend Recommendation

1: Initialize $G<V,E>$ {Subgraph of the user}
2: $userLocs \leftarrow GetUserLocationsInVicinity()$
3: $friendLocs \leftarrow GetFriendLocationsInVicinity()$
4: $expertLocs \leftarrow GetExpertLocationsInVicinity()$
5: Create new user vertex $u$ for the current user
6: $V \leftarrow V \cup u$
7: **for all** $loc$ in $userLocs$ **do**
8: $v_l \leftarrow$ new location vertex for $loc$
9: $V \leftarrow V \cup v_l$
10: **end for**
11: **for all** $friendLoc$ in $friendLocs$ **do**
12: $v_f \leftarrow$ new user vertex for $friendLoc.user$
13: $v_l \leftarrow$ new location vertex for $friendLoc.location$
14: $V \leftarrow V \cup \{v_f, v_l\}$
15: $e_f \leftarrow$ new edge between $u$ and $v_f$
16: $e_l \leftarrow$ new edge between $v_f$ and $v_l$
17: $E \leftarrow E \cup \{e_f, e_l\}$
18: **end for**
19: **for all** $expertLoc$ in $expertLocs$ **do**
20: $v_f \leftarrow$ new user vertex for $expertLoc.user$
21: $v_l \leftarrow$ new location vertex for $expertLoc.location$
22: $V \leftarrow V \cup \{v_f, v_l\}$
23: $e_f \leftarrow$ new edge between $u$ and $v_f$
24: $e_l \leftarrow$ new edge between $v_f$ and $v_l$
25: $E \leftarrow E \cup \{e_f, e_l\}$
26: **end for**

**Algorithm 6** User Subgraph Construction Algorithm cont'd

$friendsOfFriends \leftarrow GetFriendsOfFriends(V_f)$
**for all** $friendOfFriends$ in $friendsOfFriends$ **do**
  **for all** $ff$ in $friendOfFriends.friends$ **do**
    $v_{ff} \leftarrow$ new user vertex for $ff$
    $V \leftarrow V \cup v_{ff}$
  **end for**
**end for**
**for all** $loc$ in $userLocs$ **do**
  $v_l \leftarrow$ location vertex for $loc$
  $locUsers \leftarrow GetUsersThatVisitedLocation(loc)$
  **for all** $locUser$ in $locUsers$ **do**
    $v_{lu} \leftarrow$ new user vertex for $locUser$
    $V \leftarrow V \cup v_{lu}$
    $e_{lu} \leftarrow$ new edge between $v_{lu}$ and $v_l$
    $E \leftarrow E \cup e_{lu}$
  **end for**
**end for**
$V_f \leftarrow$ friends and experts vertices
$friendships \leftarrow GetFriendRelationships(V_f)$
**for all** $friendship$ in $friendships$ **do**
  $e_f \leftarrow$ new edge between $v_{friendship.f_1}$ and $v_{friendship.f_2}$
  $E \leftarrow E \cup \{e_f\}$
**end for**

using Algortihm 5. *Expert_1* and *Expert_2* are identified as local experts. *PopularLoc_1* and *PopularLoc_2* are the popular locations. Friends and friends of friends such as *Friend_4* and *Friend_5* are also depicted on the graph. Moreover, users that have check-ins at the locations that *User_1* visited before (e.g. *Friend_7*) are also added onto the graph.

RWR algorithm operates on the constructed subgraph and the recommendation results are populated. The recommendation results that are sorted by visit count are shown in Table 3.4.

Table 3.4: Friend Recommendation Results

| **Friend** | **Visit Count** |
|---|---|
| Friend_3 | 104 |
| Friend_5 | 65 |
| Friend_7 | 51 |
| Friend_4 | 47 |
| Friend_6 | 45 |

### 3.3.4 Complexity Analysis of Algorithms

At this point, it is needed to consider that the proposed recommendation methods involve two phases: graph construction and random walk based recommendation generation. For the first phase, we employ graph database queries for subgraph construction. The details as to the complexity for internal logic of query engine is not available. Therefore, here we provide the complexity analysis of the second part of our recommendation algorithm.

The complexity of random walk depends on the iteration count. The number of desired iteration count changes according to graph size and it depends on the edge count of the underlying graph. In the worst case, the maximum number of edges on a graph is $C(|V|, 2)$, where $C$ is the combination function and $|V|$ represents the total number of vertices. Let $k$ be the average number of expected transitions for each edge. All edges are not used evenly in transitions, but we define $k$ as the average number of transitions for each edge to bound the number of iterations. Hence, the complexity of random walk is calculated as $C(|V|, 2) \times k$, which is $O(|V|^2)$. After random walk, we

sort vertices and the sorting complexity is $O(|V| \log |V|)$. However, sorting complexity is asymptotically smaller than random walk complexity. Therefore, the overall complexity of our algorithm is $O(|V|^2)$.

It is important to note that this algorithm runs on the subgraph which corresponds to the user's current context. Therefore, $|V|$ is expected to have much smaller values in comparison to the whole graph.

# CHAPTER 4

# EVALUATION

In this chapter, we compare the performance of CLoRW, RWCAR and RWCFR with other algorithms. Before presenting the evaluation results, we first introduce the datasets that are used in experiments. After that we describe the evaluation methodology and metrics. Then location, activity and friend recommendation results are presented, respectively.

## 4.1 LBSN Datasets

We employ three different datasets in our experiments, which are Brightkite [10], Gowalla [10] and Foursqaure [18] datasets. All of these datasets contain users' location check-in data. In the experiments, we employ the filtered versions of these datasets for New York City. The characteristics of these datasets are given in Table 4.1.

Table 4.1: Dataset Characteristics

| | **Brightkite** | **Gowalla** | **Foursquare** |
|---|---|---|---|
| Users | 6013 | 9768 | 4906 |
| Locations | 42494 | 55181 | 28095 |
| Activities | - | - | 331 |
| Friendships | 27330 | 44300 | 26522 |
| Total Check-ins | 252200 | 262004 | 451263 |
| Check-ins per User | 41.94 | 26.82 | 91.98 |
| Friends per User | 4.55 | 4.54 | 5.4 |

As seen in Table 4.1, all these datasets have different check-ins per user. On the other hand, Brightkite and Gowalla have nearly the same number of friends per user. Foursquare has a higher value of friends per user. Original Foursquare dataset does not contain activity information. We crawled category information of locations in the original dataset using Foursquare API to collect activity information. For each location coordinate in Foursquare dataset, we queried whether such a location exists in Foursquare database. If such a location exists, then we use its category as activity information. Finally, we obtained an updated dataset that contains activity information in addition to user and location information. We analyze whether the characteristics of datasets affect the recommendation accuracy or not in our experiments.

## 4.2 Evaluation Methodology and Metrics

CLoRW, RWCAR and RWCFR are context-aware recommendation algorithms and they consider spatial context. They populate subgraphs for random walk according to current location of user that requests recommendation. Therefore, we need to determine current location of user for each run in evaluation. In order to find out such points for each user in the dataset, we employ DBSCAN (Density-based spatial clustering of applications with noise) clustering algorithm [14]. DBSCAN algorithm clusters each user's check-in data and each cluster's center is used as current location of that user in corresponding run of the algorithm. Since DBSCAN algorithm decides on the number of clusters itself, each user has different number of clusters hence different number of runs. In order to determine the number of clusters DBSCAN employs two parameters, $Eps$ and minimum number of points in a cluster. $Eps$ is the radius that defines the *Eps-neighborhood* of a point. This algorithm is able to detect noisy points according to *Eps-neighborhood* of a point. If a point does not have sufficient number of neighbors, then it is skipped during clustering process.

It is important to note that DBSCAN algorithm is only used in processing of datasets to be used in our evaluation. When our recommendation algorithms are used in real life, they directly use the current location of the user, hence does not need DBSCAN or any other clustering algorithm.

The output of DBSCAN algorithm is a set of clusters for each user. Each cluster contains check-in data for corresponding user around the center point of the cluster. Each check-in stores a timestamp for the time of visit. Since our recommendation algorithms aim to predict the future locations, activities and friends of a user, user's check-in data is sorted according to time and partitioned into training and test datasets. This partition is done according to *partition ratio* parameter. This parameters defines the ratio between number of check-ins in training and test datasets.

After we obtain training and test datasets, recommendation algorithm operates on training datasets of each user and produces the recommendation results. These results are validated using the corresponding test datasets. We employ two well-known metrics, precision and recall, while evaluating the results. In order to evaluate precision and recall values together, we use another metric, F-Measure. There are different variations of F-Measure, but we use the traditional one which is the harmonic mean of precision and recall. The formulas that are used for calculation of precision, recall and F-Measure are given in Equation 4.1, Equation 4.2, and Equation 4.3, respectively.

$$Precision = \frac{\#Recommended\,Ground\,Truths}{\#Recommendations} \tag{4.1}$$

$$Recall = \frac{\#Recommended\,Ground\,Truths}{\#Ground\,Truths} \tag{4.2}$$

$$F - Measure = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{4.3}$$

Each experiment is performed for all users in Brighkite, Gowalla and Foursquare datasets in location and friend recommendation experiments. On the other hand, activity recommendation algorithms are only performed on Foursquare dataset because other datasets do not contain activity information. For each dataset, the average precision and recall values are calculated for all of the algorithms.

## 4.3 Location Recommendation Experiments

There are several location recommendation algorithms in the literature as summarized in Section 2. We identified the common approaches employed by these algorithms and used them as baselines in our evaluation. These algorithms are popularity-based, friend-based and expert-based baseline location recommendation algorithms and user-based CF algorithm. We also compare our algorithm with a well-known multi-criteria location recommendation algorithm, namely USG [50]. In this section, we evaluate our location recommendation algorithm and compare the results with these algorithms.

Popularity-based location recommendation suggests the most visited locations in a geospatial range. Friend-based baseline algorithm recommends the most visited locations by friends in vicinity. On the other hand, expert-based baseline suggests the most visited locations by experts.

In user-based CF approach, the general intuition is that similar users visit similar locations. In this approach a User-Location matrix is constructed. In this matrix, each entry represents the probability of a particular user to visit a particular location. Since it is a user-based CF algorithm, as the first step, similarity scores between users are calculated. This calculation is performed using cosine similarity. Then location scores for each user are calculated using these similarity scores.

USG is a multi-criteria algorithm that considers user preference (user-based CF), social influence (friend-based CF) and geographical influence for location recommendation [50]. It fuses user preference, social and geographical influence using a linear model to predict location scores.

The common configuration parameters and their values for location recommendation experiments are given in Table 4.2.

### 4.3.1 CLoRW Parameter Settings Experiments

In the following sections, we aim to analyze the accuracy of CLoRW under changing number of recommendations, dataset partition ratio and minimum number of check-

Table 4.2: Common Configuration of Parameter Setting Experiments

| **Parameter** | **Value** |
|---|---|
| Number of Recommendations | 5 |
| Radius | 3000 m |
| Expert Count | 3 |
| Popular Location Count | 3 |
| DBSCAN Eps | 3000 m |
| DBSCAN Minimum Cluster Points | 1 |
| Random Walk Restart Probability | 0.1 |
| Dataset Partition Ratio | 1 |
| Minimum Number of Check-ins | 6 |

ins. The precision, recall and F-Measure values are calculated for different values of these parameters. In each experiment only one of the parameters is changed while the others remain constant, and the effect of that parameter is analyzed.

#### 4.3.1.1 Number of Recommendations

This experiment aims to analyze the change of precision and recall of CLoRW with respect to number of recommendations. In this experiment, it is also aimed to find the optimal number of recommendations for CLoRW. The results of this experiment are shown in Figure 4.1.

As shown in Figure 4.1, the recall of CLoRW increases rapidly while the precision drops as the number of recommendations increases. CLoRW achieves the best performance when the number of recommendations is 2 ($F - measure = 0.421102798$) according to F-measure. However, in other experiments we select the number of recommendations as 5, because in most of the cases desired number of recommendations are more than 2.

It is expected that when the number of recommendations increases, recall should also increase. However, in this experiment and other experiments, if a particular algorithm cannot make desired number of recommendations, then that test result data is not considered in the evaluation result. For example, if an algorithm cannot suggest more than 2 locations for a user at a particular location, then when the required

number of recommendations is 3 or more, these results are not considered in the evaluation. Therefore, the dataset becomes smaller as the number of recommendation increases. This situation causes drop in the recall value as the required number of recommendations increases.

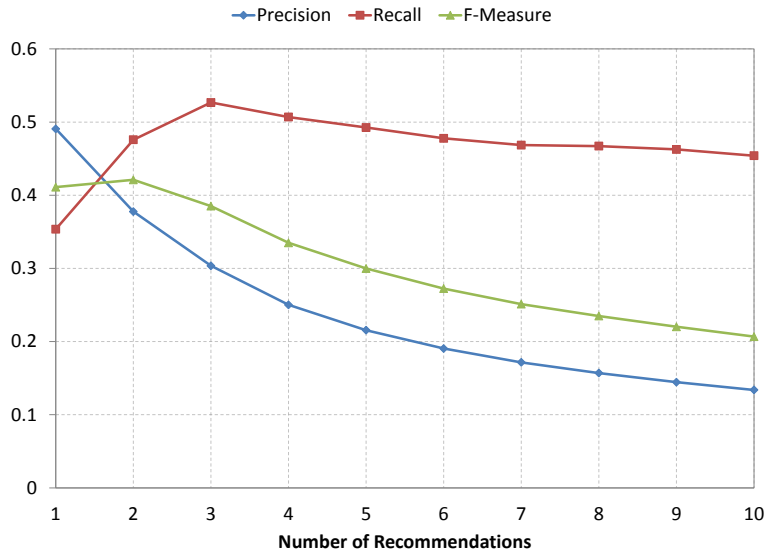



Figure 4.1: Precision, Recall and F-Measure Values vs. Number of Recommendations

#### 4.3.1.2 Dataset Partition Ratio

This experiment aims to find the behavior of our algorithm when the dataset partition ratio is set as 1, 2 and 3. In this experiment, we want to analyze whether CLoRW could construct a usable model for different amounts of check-in data. Here the number of recommendations are fixed to 5 and does not change throughout the experiment.

The results of this experiment are given in Figure 4.2. CLoRW has the best performance when partition ratio is set to 1 and have the lowest performance when it is 3. Hence, the optimal value for this parameter can be considered as 1. This shows that in our algorithm, it is not necessary to have a large training data to obtain a successful prediction model. Another observation is that the precision decreases as the dataset partition ratio parameter increases. This is due to the fact that as the size of the test

dataset decreases, it fails to represent user behavior.

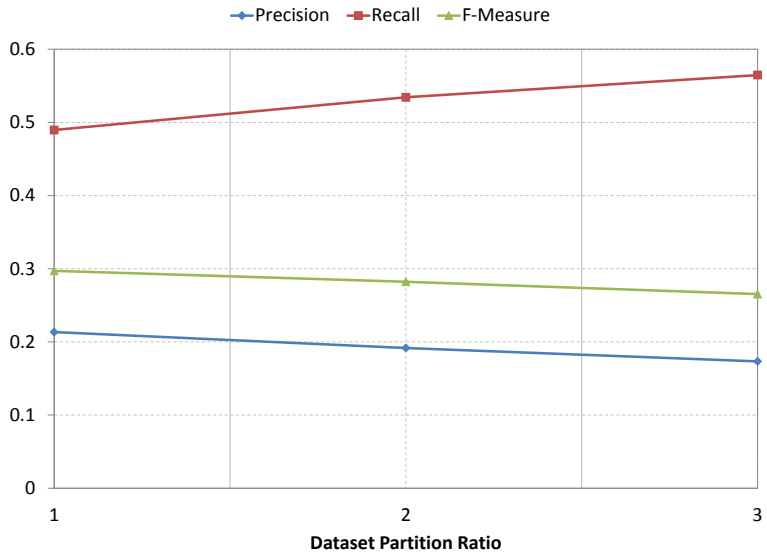



Figure 4.2: Precision, Recall and F-Measure Values vs. Dataset Partition Ratio

#### 4.3.1.3 Minimum Number of Check-ins

Our proposed algorithm considers user's previous check-ins for recommendation. Therefore, the number of check-ins may affect the efficiency of CLoRW. In this experiment, we test the accuracy of CLoRW under the changing number of minimum check-ins. If a particular user's check-in count in a cluster is lower than the minumum number of check-ins threshold, then the results of that cluster is not taken into account. In this experiment, we set the minimum number of check-ins to 4, 6, 8 and 10. The number of recommendations and dataset partition ratio are fixed to 5 and 1, respectively. The results are illustrated in Figure 4.3.

As shown in Figure 4.3, the performance of CLoRW increases as the value of minimum number of check-ins increases. Our algorithm achieves the best performance when the value of minimum number of check-ins is 10. This concludes that the more locations that a user has visited, the more accurate are the recommendation results for our algorithm. However, in other experiments minimum number of check-ins is

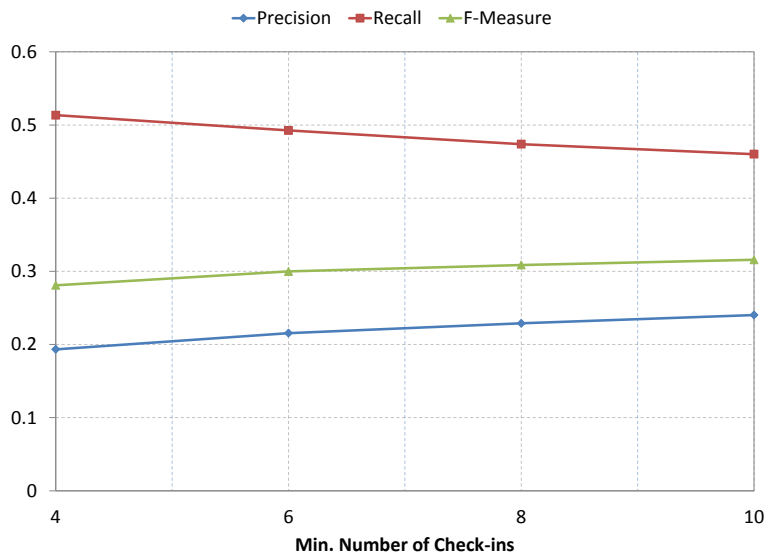



Figure 4.3: Precision, Recall and F-Measure Values vs. Min. Number of Check-ins

chosen as 6 because the performance of CLoRW changes slightly when the number of check-ins is increased, and we do not want to lose much data from the original dataset.

### 4.3.2 Comparison with Baselines and Similar Approaches

In order to evaluate our proposed algorithm, CLoRW, we compare it with baselines, user-based CF location recommendation and USG algorithm [50] using Brightkite, Gowalla and Foursquare datasets. The baseline location recommendation algorithms are popularity, friend and expert based location recommendation algorithms. In baseline tests, we aim to find if our extra considerations from baselines increase location recommendation accuracy or not. In this experiment the expert count is selected as 3. In the next section, we further analyze the accuracy of expert baseline and CLoRW as the number of experts increases. CF-based approaches are widely used in location recommendation algorithms. In this experiment, we also analyze the performance of CLoRW in comparison to user-based CF location recommendation algorithm. We conducted experiments for performance evaluation in comparison to a similar work in the literature, USG algorithm. USG is a multi-criteria algorithm that considers

user preference, social influence and geographical influence. Our algorithm is also a multi-criteria algorithm that considers popular locations, friend locations, local expert's locations and user's own locations to generate recommendations.

The results of the experiments for Brightkite, Gowalla and Foursquare datasets are given in Figure 4.4, Figure 4.5, Figure 4.6, Figure 4.7, Figure 4.8, Figure 4.9, and Figure 4.10, Figure 4.11, Figure 4.12, respectively. The results clearly indicate that CLoRW outperforms all the baselines and CF-based location recommendation according to precision, recall and F-Measure metrics. This shows that it is not sufficient to consider only friend, popular or expert locations disjointly to generate location recommendations, instead it is better to recommend locations using a graph structure that combines all of them. Moreover, CLoRW's consideration of user's location history also increases the accuracy because it is more likely for a user to visit same locations in the future.

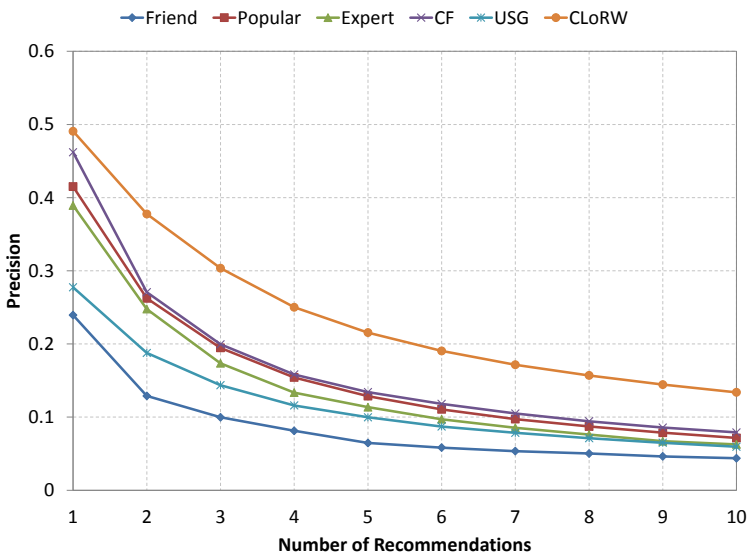



Figure 4.4: Precision Values of Baselines, CF, USG, and CLoRW vs. Number of Recommendations for Brightkite Dataset

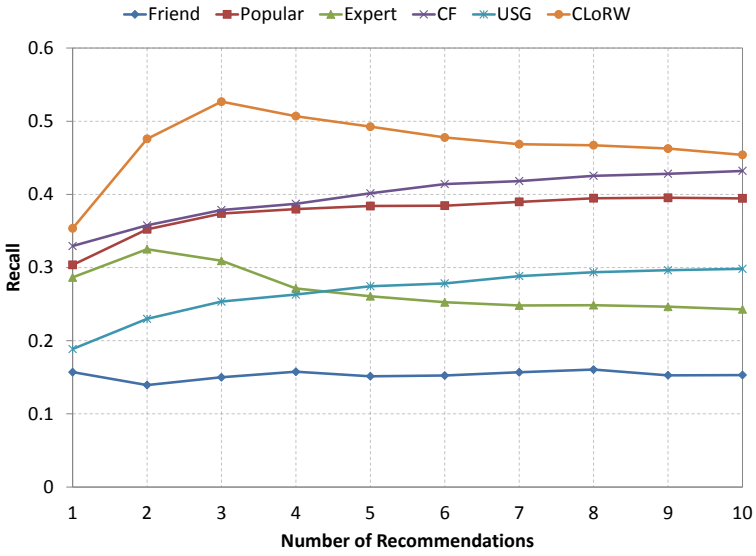



Figure 4.5: Recall Values of Baselines, CF, USG, and CLoRW vs. Number of Recommendations for Brightkite Dataset

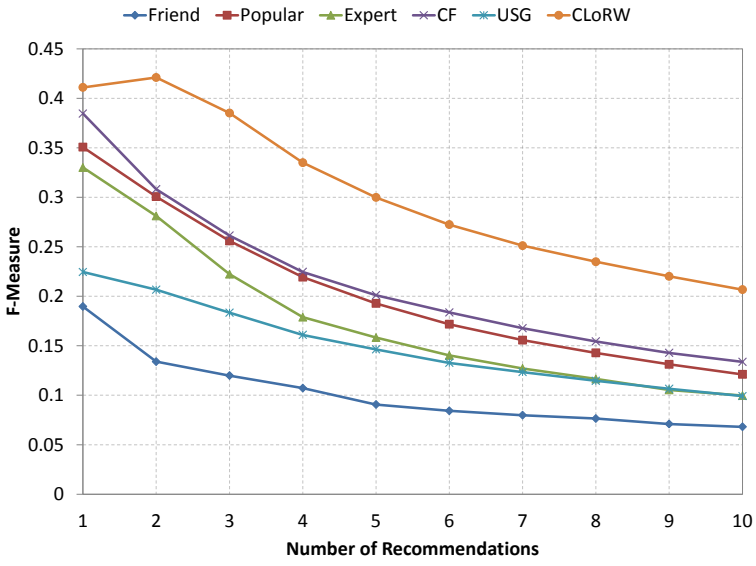



Figure 4.6: F-Measure Values of Baselines, CF, USG, and CLoRW vs. Number of Recommendations for Brightkite Dataset

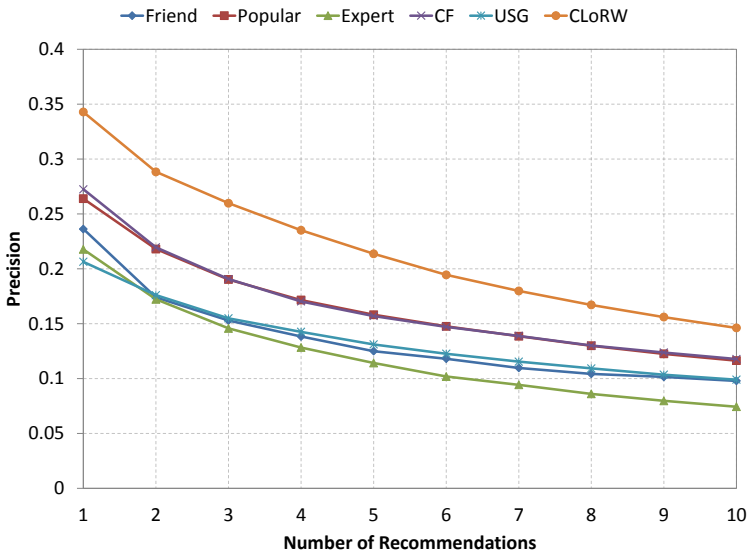



Figure 4.7: Precision Values of Baselines, CF, USG, and CLoRW vs. Number of Recommendations for Gowalla Dataset

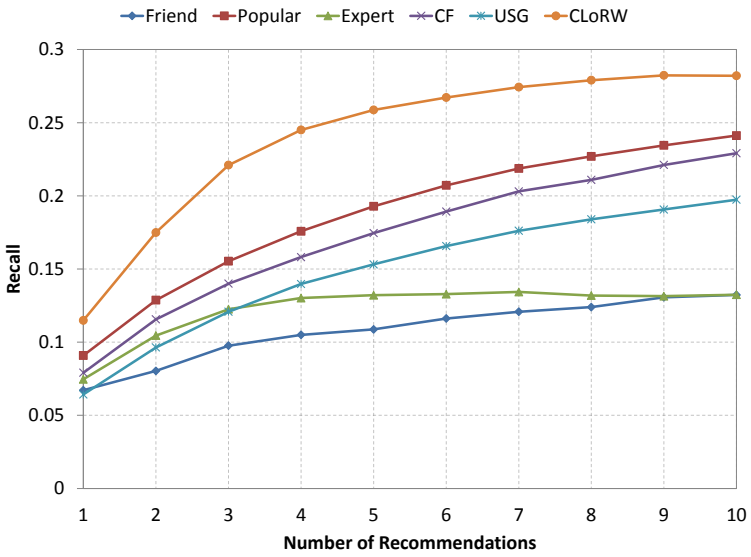



Figure 4.8: Recall Values of Baselines, CF, USG, and CLoRW vs. Number of Recommendations for Gowalla Dataset

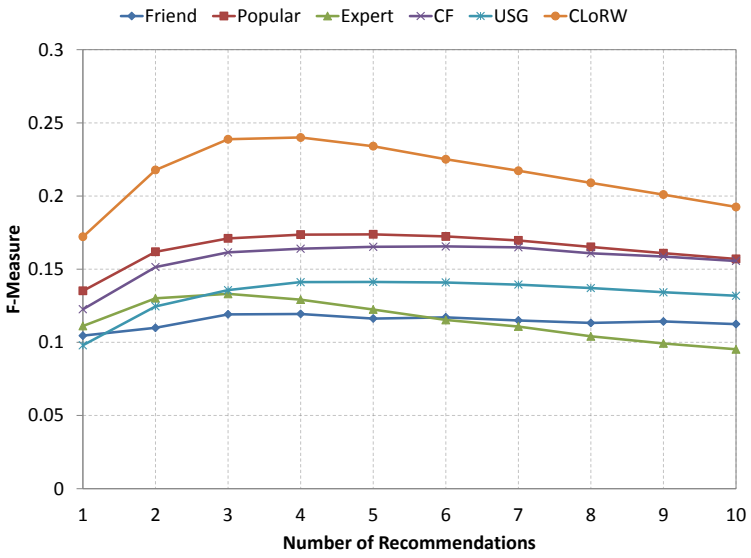



Figure 4.9: F-Measure Values of Baselines, CF, USG, and CLoRW vs. Number of Recommendations for Gowalla Dataset

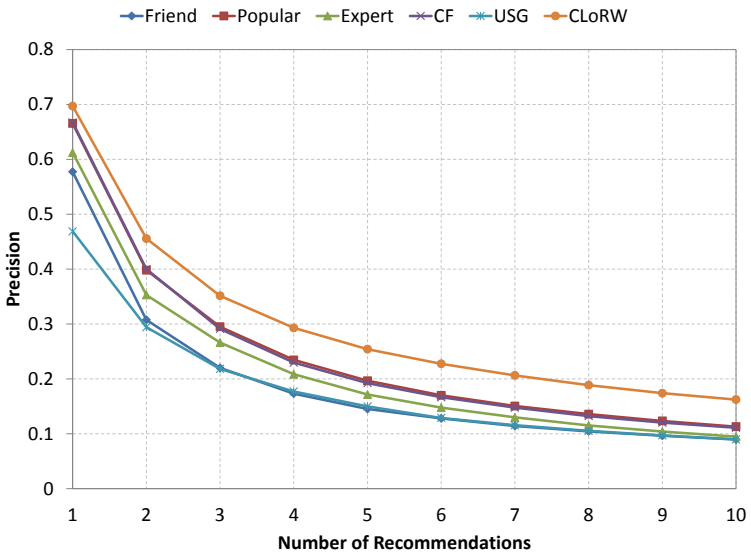



Figure 4.10: Precision Values of Baselines, CF, USG, and CLoRW vs. Number of Recommendations for Foursquare Dataset

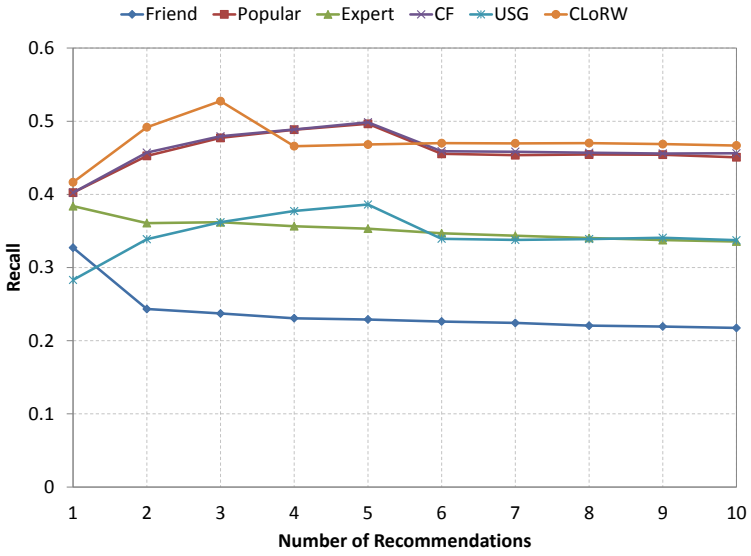



Figure 4.11: Recall Values of Baselines, CF, USG, and CLoRW vs. Number of Recommendations for Foursquare Dataset

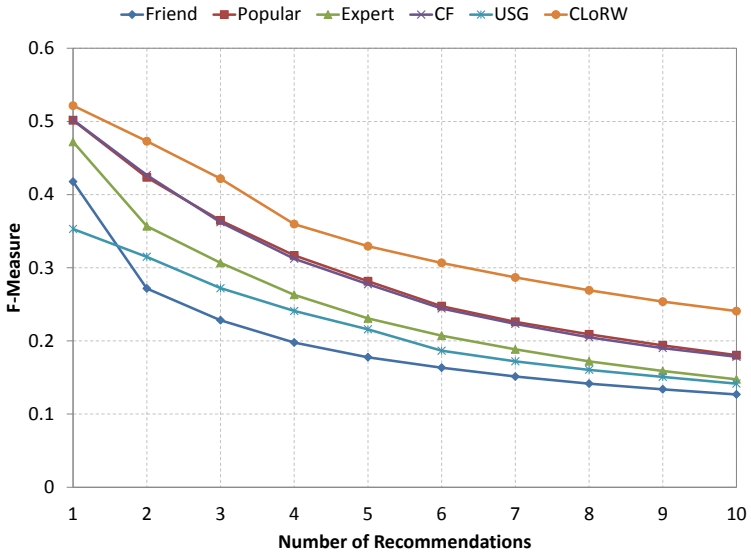



Figure 4.12: F-Measure Values of Baselines, CF, USG, and CLoRW vs. Number of Recommendations for Foursquare Dataset

Friend-based location recommendation algorithm has very poor performance. It has the lowest recommendation accuracy in nearly all the experiments. This result indicates that the friends of a user cannot span sufficient locations for user's current context. If a particular user has few friends, then it is difficult to recommend locations to that user by using friend-based algorithm. Therefore, we can conclude that user's friends themselves are not sufficient for location recommendation.

Expert-based algorithm has higher recommendation accuracy than friend-based baseline. However, in Gowalla experiments, there are a few cases that friend baseline outperforms expert baseline. On the other hand, even recommending popular locations produce more accurate results than expert baseline. This is because only locations of local experts cannot span enough locations in current location context.

Popularity baseline have nearly the same recommendation accuracy with CF-based approach. In Gowalla experiments, popularity baseline produce more accurate results while CF-based approach has higher accuracy in Brightkite experiments. In Foursquare experiments, the results are nearly the same. Popularity-based baseline algorithm is widely used in location recommendation. The logic behind this is that if a location is popular, then it is more probable for other users to visit that location. However, users have social relationships and their personal preferences. Therefore, friends and user's location history may affect user's future locations. Moreover, opinions of local experts may also influence other users' decisions. The results in these experiments also confirm the fact that popularity itself is not sufficient for location recommendation.

Each baseline considers location recommendation in one dimension. On the other hand, CLoRW takes friends, popularity, experts and location history into account and outperforms all the baselines in all of the experiments.

The results also notably show that CLoRW has better performance than user-based CF location recommendation algorithm. Moreover, if we compare these results with the baseline results, we see that CF-based approach even cannot outperform the popularity baseline. Although CF-based approach is widely used in recommendation, we can conclude that it is not a good candidate for location recommendation.

Although USG algorithm considers three different criteria for recommendation, its performance is even worse than user-based CF algorithm. USG also employ CF for estimating user preference and social preference. Moreover, it considers the distance between locations as geographical influence. Our algorithm also considers personal, social and spatial context. Consideration of multiple influences is important, but it is more important to combine them using a suitable fusion framework. USG uses a linear model to fuse final results. On the other hand, we employ a graph model for fusion of input data (i.e. users, locations, experts) not the results. The experiment results show that our methodology of fusion is more effective than the methodology used in USG.

The results obtained across three datasets agree with one another. It is an important result because it clearly indicates that CLoRW is not a dataset specific location recommendation algorithm. Although, all there datasets have different characteristics such as check-ins per user and friends per user, CLoRW achieves the best performance in all of the test cases.

The results also show that check-ins per user parameter affects the recommendation accuracy. The recommendation accuracy increases as the number of check-ins per user increases. This is because, if we have higher check-in per user values, we have large number of check-ins in training and test datasets.

**Changing the Number of Experts**

In this set of experiments, we further elaborate on the number of experts, as it affects the accuracy of expert-based baseline algorithm and CLoRW. The configuration parameter values of this experiment is same as the baseline experiments except the expert count parameter. In this experiment, the expert count is increased up to 10 and the accuracy results of the algorithms are compared. The results of this experiment are depicted in Figure 4.13, Figure 4.14 and Figure 4.15.

CLoRW outperforms expert baseline for all values of number of experts. The precision values of expert baseline and CLoRW nearly do not change as the number of experts increases. The recall value increases as the expert count increases for both of the algorithms. However, when the precision and recall values are evaluated together

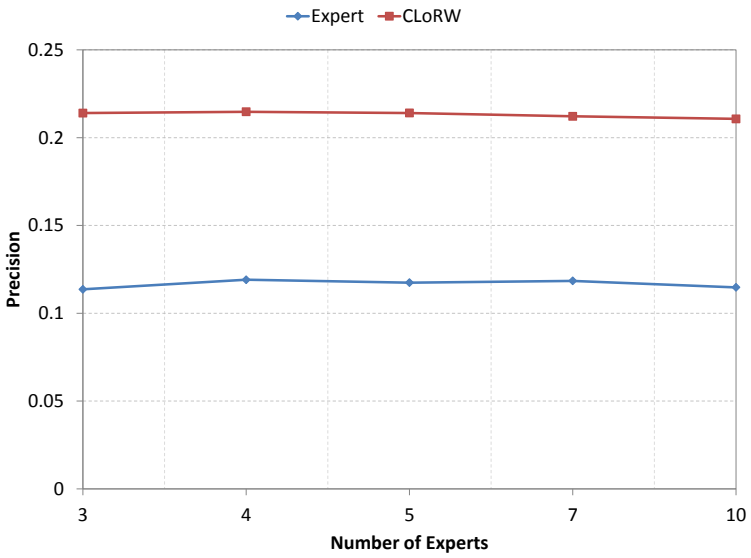



Figure 4.13: Precision Values of Expert Baseline and CLoRW

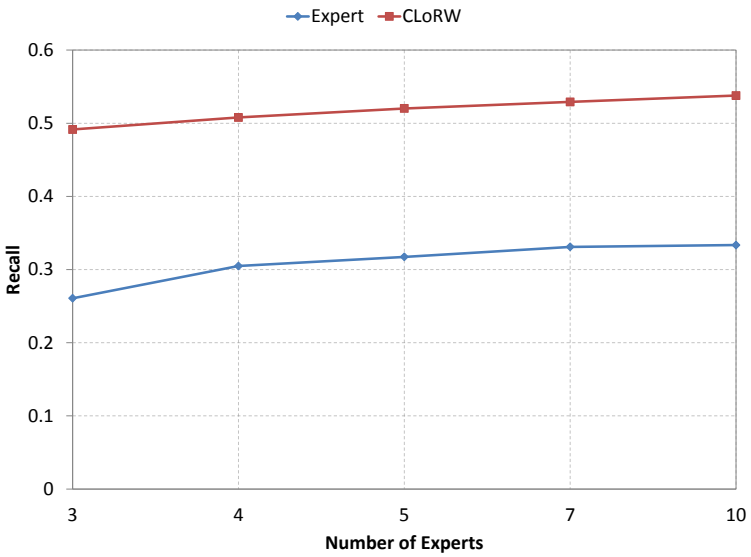



Figure 4.14: Recall Values of Expert Baseline and CLoRW

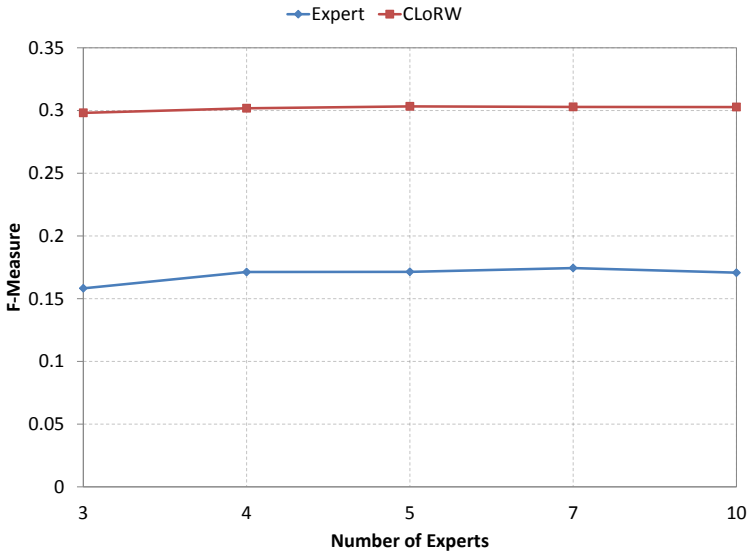



Figure 4.15: F-Measure Values of Expert Baseline and CLoRW

using F-Measure, we see that the performance of expert baseline and CloRW does not change when the number of experts is increased. This indicates that it is better to represent a locality with fewer but reasonable number of experts.

### 4.3.3 Comparison between CLoRW and CLoRW_A

CLoRW_A employs activities in location recommendation different from CLoRW. The datasets that we use in our experiments originally do not contain activity information. They only contain users, locations (latitude, longitude) and check-ins information. Brightkite and Gowalla services have become obsolete recently. Therefore, we do not have the chance of enriching these datasets with activity information through these services. Firstly, we try to enrich these datasets based on each location's coordinates. We employ Google Places service to match the coordinate information of each location with the locations that are stored in these services. However, we realized that there are several location candidates for a particular coordinate information in most of the cases. Therefore, it is hard to match our locations with the one in Google Places database.

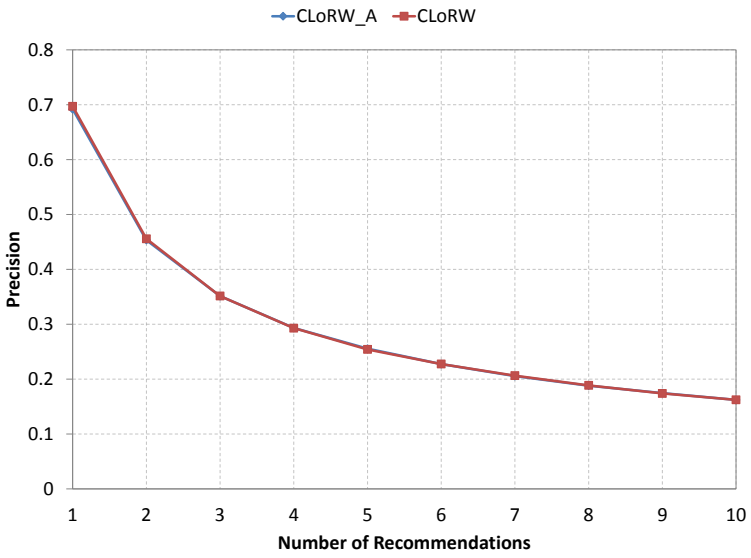



Figure 4.16: Precision Values of CLoRW and CLoRW_A for Foursquare Dataset

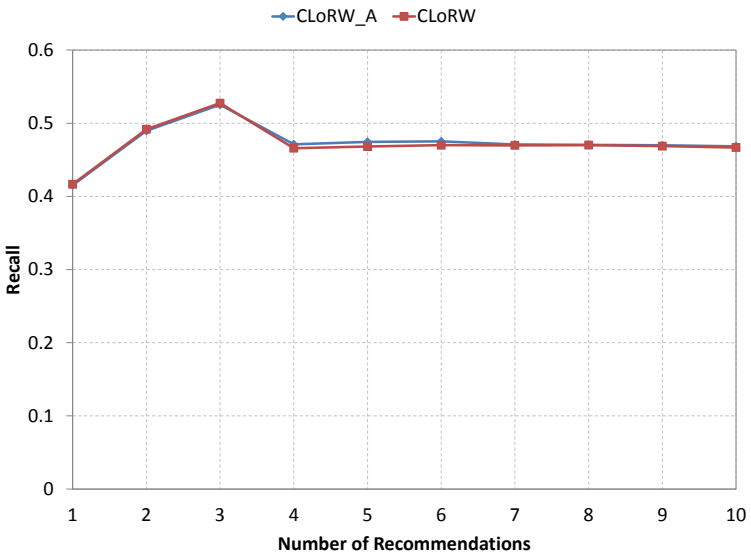



Figure 4.17: Recall Values of CLoRW and CLoRW_A for Foursquare Dataset

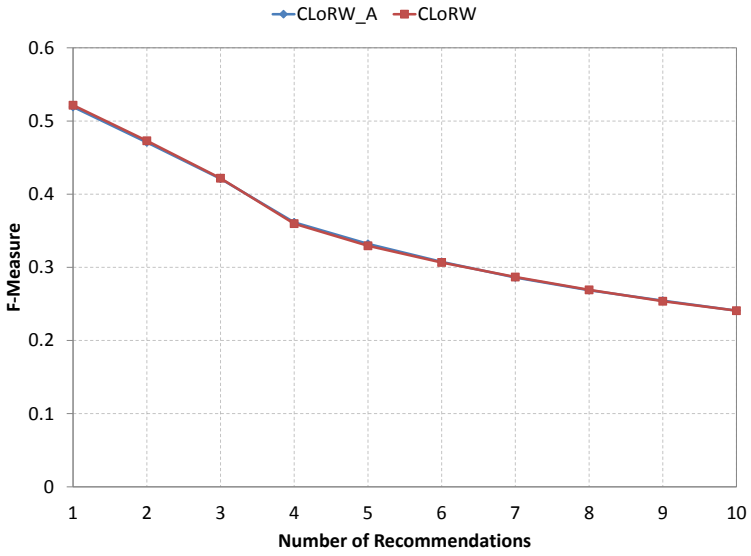



Figure 4.18: F-Measure Values of CLoRW and CLoRW_A for Foursquare Dataset

The Fourquare dataset that we employ in our experiments also only contain coordinate information for each location. However, Foursquare is an active service. Our test dataset is also crawled from Foursquare service and we think that location coordinates should match with the locations in Foursquare database. Therefore, as a second solution, we use Foursquare API and try to match our locations. We were able to match nearly 12% of locations in our dataset. Foursquare location categories are used as activities to enrich our original test dataset. This dataset is used in evaluation of CLoRW_A. The evaluation results are compared with the results of CLoRW. These results are given in Figure 4.16, Figure 4.17, and Figure 4.18.

The results show that adding activities to our model nearly does not affect location recommendation accuracy. However, CLoRW_A can recommend locations when CLoRW cannot produce any recommendation results, since new model contains more locations. We further analyze whether CLoRW_A can perform better than CLoRW when we only consider the cases that CLoRW can produce recommendation results. Therefore, we filter the CLoRW_A results to analyze the results deeply. The results are given in Figure 4.19, Figure 4.20, and Figure 4.21. The filtered version of CLoRW_A is renamed as CLoRW_A_F in result charts.

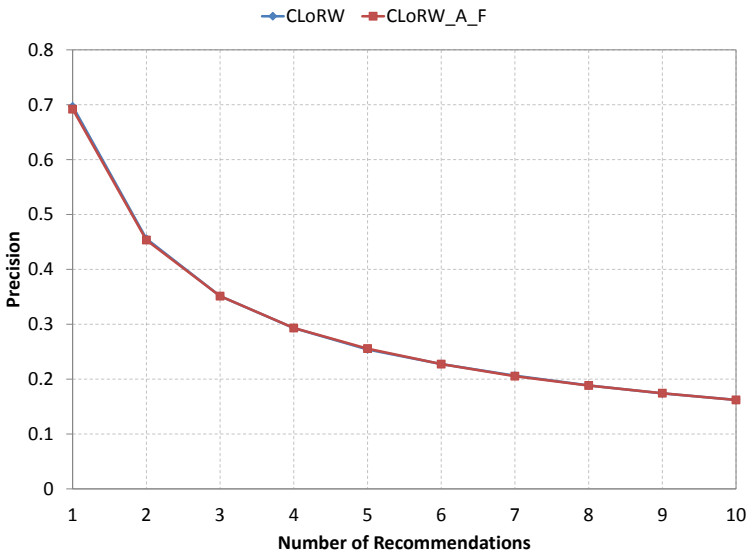



Figure 4.19: Precision Values of CLoRW and CLoRW_A_F for Foursquare Dataset

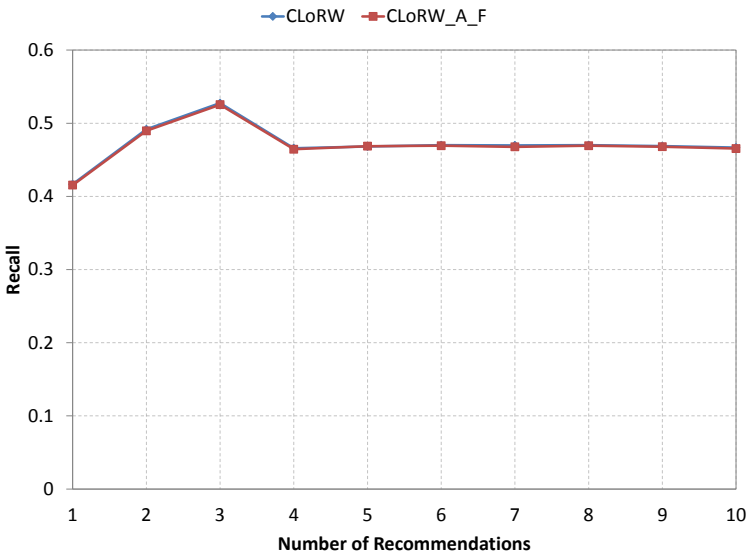



Figure 4.20: Recall Values of CLoRW and CLoRW_A_F for Foursquare Dataset

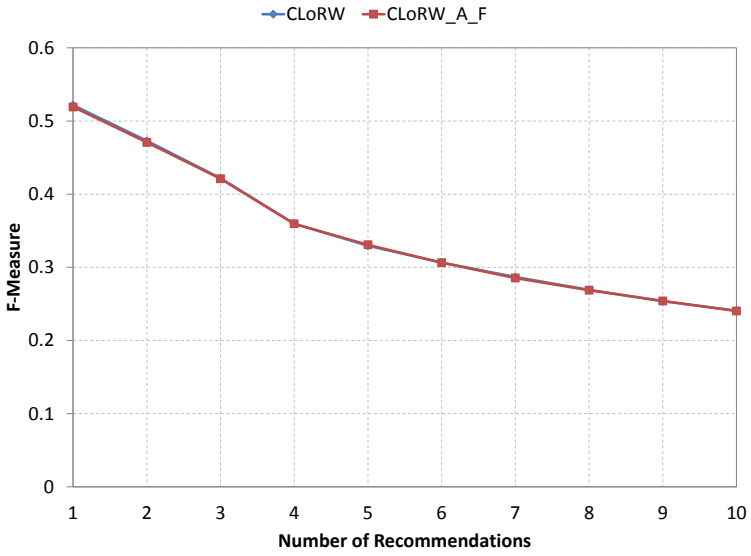


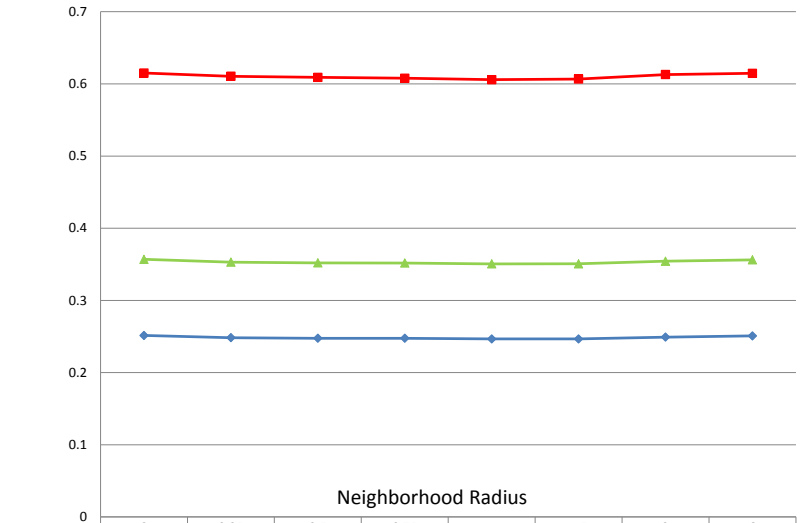

Figure 4.21: F-Measure Values of CLoRW and CLoRW_A_F for Foursquare Dataset

According to the results, CLoRW, CLoRW_A and CLoRW_A_F have nearly same recommendation accuracy. In other words, we are not able to improve our location recommendation accuracy. However, we increased the size of the set of locations that can be recommended in each case. Moreover, we can employ the newly constructed model that contains activity nodes in activity recommendation.

### 4.3.4 Location-Location Edges Experiments

In this experiment, we test whether adding location-location edges to our undirected unweighted graph model increases CLoRW's performance or not. In order to achieve this, we setup location-location edges using different values of neighborhood radius. If a location is in another location's neighborhood than an edge is added to graph. When the neighborhood radius is set to 0, the algorithm is identical to original CLoRW. The results of this experiment is given in Figure 4.22.

According to results in Figure 4.22, the best performance is achieved when neighborhood radius is 0 (i.e. no direct edges exist between locations). This result clearly indicates that consideration of distance between locations as a similarity measure worsens the performance of CLoRW. In other words, distance is not a good candidate




| | 0 | 0.25 | 0.5 | 0.75 | 1 | 1.5 | 3 | 6 |
|---|---|---|---|---|---|---|---|---|
| Precision | 0.251388544 | 0.24828141 | 0.24751084 | 0.24751084 | 0.2466657 | 0.2466657 | 0.249151407 | 0.250791974 |
| Recall | 0.615079718 | 0.610544306 | 0.609001813 | 0.60784015 | 0.6058566 | 0.606789984 | 0.612931568 | 0.614666819 |
| F-Measure | 0.356906309 | 0.35300946 | 0.351972734 | 0.351778458 | 0.350592688 | 0.350748794 | 0.354287852 | 0.356235343 |

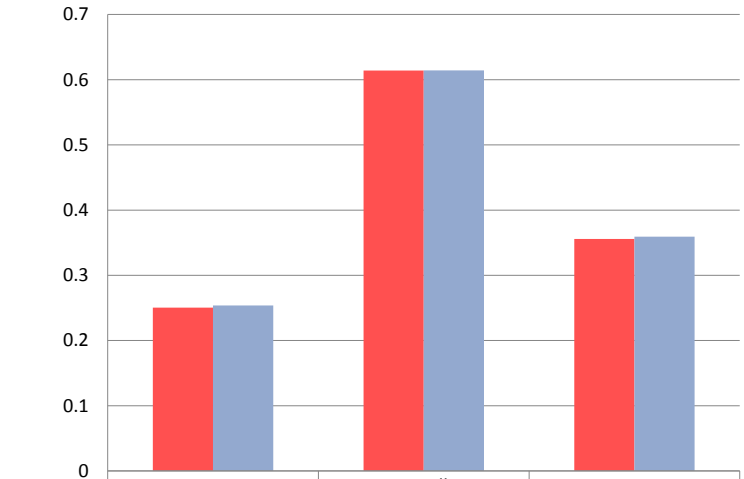
Figure 4.22: Precision, Recall and F-Measure Values versus Neighborhood Radius

measure for estimating the similarity between locations.

### 4.3.5 Weighted Location Recommendation Experiments

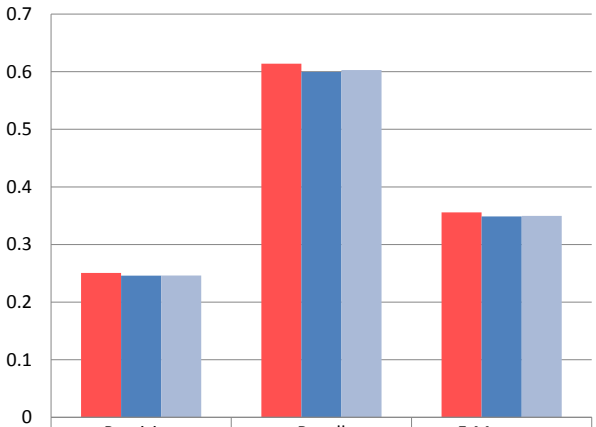
CLoRW employs an undirected unweighted graph model to recommend locations. We also try to develop the weighted version of CLoRW, **C**ontext-Aware **Lo**cation Recommendation with **W**eighted **R**andom **W**alk (CLoWRW). In this experiment we setup edges between a particular user and his/her locations for each check-in. In CLoRW, there is a single edge between a user and a location even if there are several check-ins to that location by that user. However, in this experiment there are number of check-in count of edges between a user and a location. This procedure is identical to assigning the check-in count as weight to user-location edges. The results of this experiment are given in Figure 4.23.

According to the results, CLoWRW produces slightly better results when compared to CLoRW for all three types of metrics. This result indicates that the check-in count of a user to a location is a good candidate similarity measure between a user and a location.



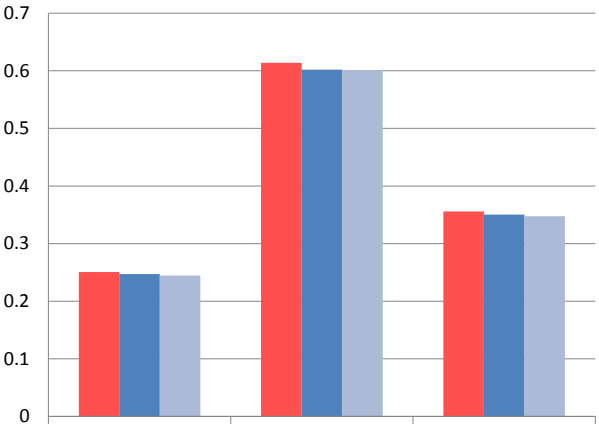

| | Precision | Recall | F-Measure |
|---|---|---|---|
| CLoRW | 0.250518546 | 0.61396315 | 0.355841324 |
| CLoWRW | 0.253810595 | 0.614184918 | 0.359187662 |

Figure 4.23: Precision, Recall and F-Measure Values of CLoRW and CLoWRW




| | Precision | Recall | F-Measure |
|---|---|---|---|
| CLoRW | 0.250518546 | 0.61396315 | 0.355841324 |
| CLoWRW-VisitCount | 0.245914543 | 0.599631956 | 0.348787958 |
| CLoWRW-VisitCount (NodeType) | 0.24623938 | 0.602814443 | 0.349651932 |

Figure 4.24: Precision, Recall and F-Measure Values of CLoRW and CLoWRW-VisitCount




| | Precision | Recall | F-Measure |
|---|---|---|---|
| CLoRW | 0.250518546 | 0.61396315 | 0.355841324 |
| CLoWRW-Temporal | 0.247063968 | 0.602058823 | 0.350354609 |
| CLoWRW-Temporal (NodeType) | 0.244515242 | 0.601353169 | 0.347666408 |

Figure 4.25: Precision, Recall and F-Measure Values of CLoRW and CLoWRW-Temporal

In later experiments, we try two different parameters for assigning weights to user-location edges. These are location node's visit count and temporal check-in rate. The details of these parameters are given in Section 3.3.1.5. As mentioned in that section, we have two different parameters and two different strategies, hence four different experiments. The results of these experiments are shown in Figure 4.24 and Figure 4.25.

As seen from the figures, neither visit count parameter nor temporal check-in parameter is able to produce better results than CLoRW for both of the strategies. Therefore, we decide not to employ these parameters in our location recommendation algorithm.

## 4.4 Activity Recommendation Experiments

We conduct several experiments to compare the performance of our activity recommendation algorithm (RWCAR) with three different algorithms. These algorithms represent the common approaches that are employed in location and activity recommendation studies in the literature. We use these approaches as baselines in our evaluation. The definitions of these baselines are given below:

Table 4.3: Activity Experiments Configuration

| **Parameter** | **Value** |
|---|---|
| Random Walk Restart Probability | 0.1 |
| Radius | 3000 m |
| Expert Count | 3 |
| Popular Activity Count | 3 |
| DBSCAN Eps | 3000 m |
| DBSCAN Minimum Cluster Points | 1 |
| Dataset Partition Ratio | 1 |
| Minimum Number of Check-ins | 6 |

**Popularity-Based Activity Recommendation (PBAR)**: Recommends the most performed activities in recommendation region.

**Friend-Based Activity Recommendation (FBAR)**: Considers the friend links and recommends the most performed activities by friends in vicinity.

**Expert-Based Activity Recommendation (EBAR)**: Considers the experts and their activities and recommends the most performed activities by experts in vicinity.

The common configuration parameters and their values for activity recommendation experiments are given in Table 4.3.

### 4.4.1 Comparison with Baselines

In this section, we compare RWCAR with other algorithms. The results of experiments that are conducted on Foursquare dataset are given in Figure 4.26, 4.27 and 4.28. The results are produced for different number of recommendations (i.e. 1,2,3,4 and 5). The results clearly indicate that RWCAR outperforms FBAR, EBAR and PBAR considering precision, recall and F-Measure metrics. RWCAR is a multi-criteria algorithm and considers friendships, experts and popularity together. Our algorithm fuses all these data by using an LBSN model. Moreover, RWCAR employs user's location and activity history as personal context. On the other hand, other algorithms consider friendships, expert users and popularity disjointly and do not provide a data fusion framework.

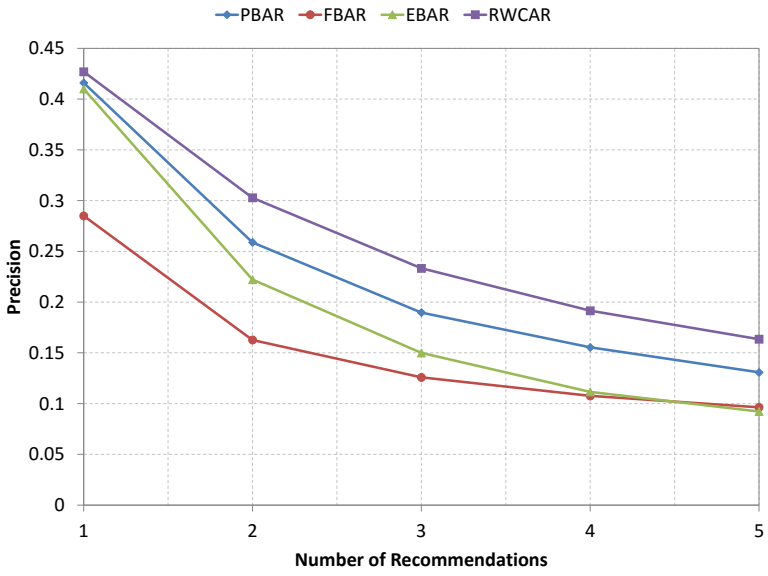



Figure 4.26: Precision Values of Baselines and RWCAR vs. Number of Recommendations for Foursquare Dataset

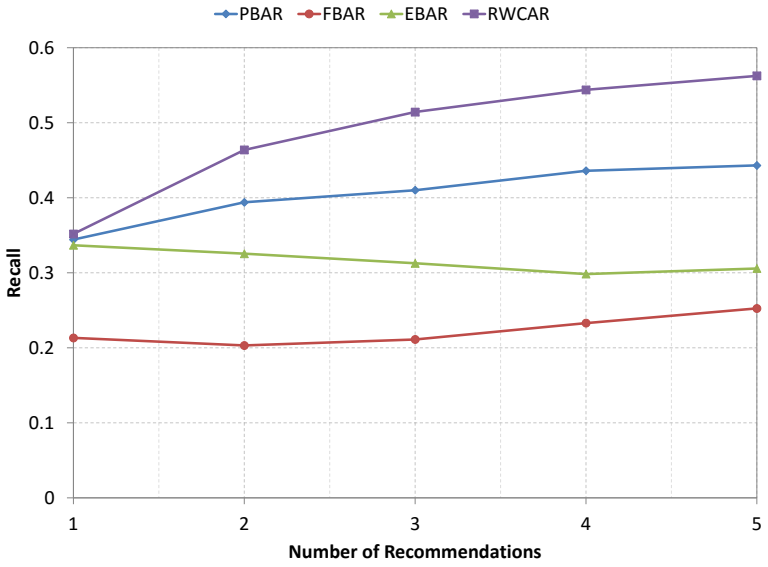



Figure 4.27: Recall Values of Baselines and RWCAR vs. Number of Recommendations for Foursquare Dataset

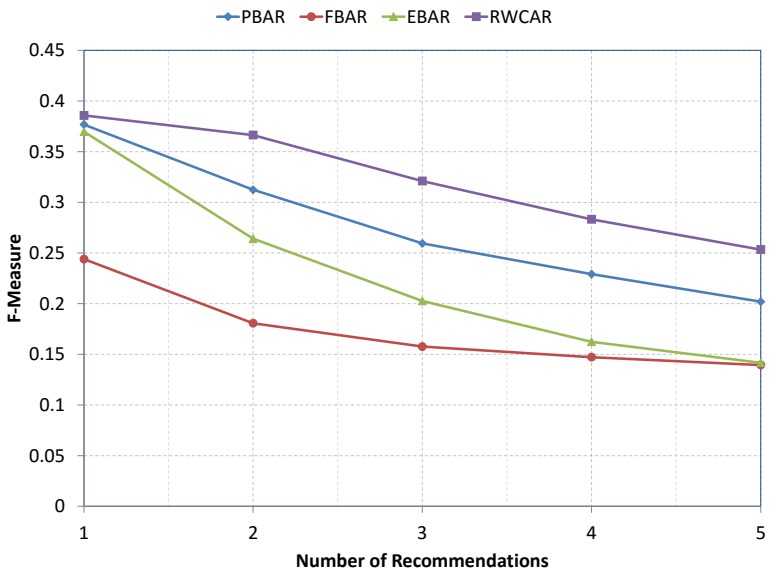



Figure 4.28: F-Measure Values of Baselines and RWCAR vs. Number of Recommendations for Foursquare Dataset

FBAR has the poorest performance among all the algorithms. When number of recommendations is 5, its precision value is greater than EBAR's. However, in the same case, when precision and recall values are considered together EBAR has higher recommendation accuracy. Therefore, we can conclude that FBAR has the lowest recommendation accuracy in all the experiments. This results shows that friends of a user itself is not sufficient for generating accurate activity recommendations. Moreover, FBAR is not a suitable algorithm who has few friends in the network.

EBAR has higher recommendation accuracy than FBAR, but PBAR's performance is higher than EBAR in all of the experiments. This shows that local activity experts are more useful than friends in activity recommendation. However, PBAR and RWCAR still have more recommendation accuracy than EBAR. This shows that EBAR is a better choice than FBAR, but it still cannot span enough activities in current spatial context.

Popularity-based techniques are widely used in recommendation. It is a simple approach, but it produces reasonable results because popular items (e.g. activities) are very strong candidates for recommendation. The logic behind PBAR algorithm is that if an activity is popular in vicinity, then it is more probable for other users to

perform that activity in the same region. However, PBAR does not consider users' social links (i.e. friends), personal preferences (i.e. user's location/activity history) and does not ask experts' opinions about the region. This lack of consideration puts PBAR behind RWCAR. However, PBAR's performance is significantly greater than EBAR and FBAR's performance. This shows that popularity is a valuable feature for activity recommendation, but popularity itself is not satisfactory for activity recommendation.

## 4.5 Friend Recommendation Experiments

In order to compare the performance of our friend recommendation algorithm, we conduct several experiments. The common configuration parameters and values for friend recommendation experiments are given in Table 4.4.

Table 4.4: Friend Experiments Configuration

| **Parameter** | **Value** |
|---|---|
| Random Walk Restart Probability | 0.1 |
| Radius | 3000 m |
| Expert Count | 3 |
| Popular Location Count | 3 |
| DBSCAN Eps | 3000 m |
| DBSCAN Minimum Cluster Points | 1 |
| Dataset Partition Ratio | 1 |
| Minimum Number of Check-ins | 6 |
| Minimum Number of Friends | 6 |

In these experiments we employ three different algorithms for comparison. These algorithms are common approaches that are used in location, activity and friend recommendation. These approaches are employed as baseline algorithms in experiments. The definitions of these baselines are given below:

**Popularity-Based Friend Recommendation (PBFR)**: Recommends the users that have check-ins in recommendation region and have the highest number of friends.

**Friend-Based Friend Recommendation (FBFR)**: Recommends the second degree friends (i.e. friends of friends) sorted by the number of friends.

**Expert-Based Friend Recommendation (EBFR)**: Recommends the friends of local experts that have the highest number of friends.

### 4.5.1 Comparison with Baselines

In this section we compare RWCFR with baseline friend recommendation algorithms. The experiments are conducted on three different datasets, which are Brightkite, Gowalla and Foursquare datasets. The results of the experiments are given in Figure 4.29, Figure 4.30, Figure 4.31, Figure 4.32, Figure 4.33, Figure 4.34, and Figure 4.35, Figure 4.36, Figure 4.37, respectively.

As reported by the results, it is clear that RWCFR outperforms PBFR, FBFR and EBFR considering precision, recall and F-Measure metrics. RWCFR is a multi-criteria algorithm and it considers popularity, friendships and local experts together. The proposed LBSN model fuses these data seamlessly. In addition to this, our friend recommendation algorithm also takes user's personal preference into consideration. However, other algorithms are based on a single consideration and they do not provide a data fusion model.

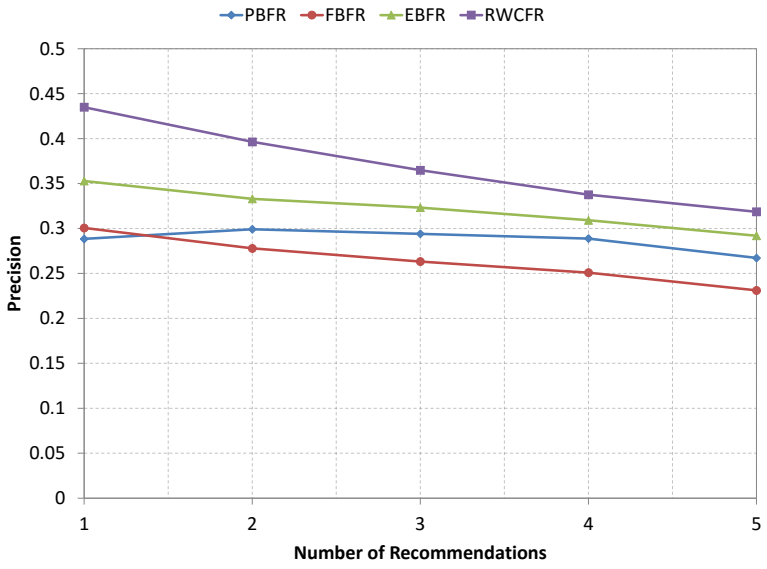



Figure 4.29: Precision Values of Baselines, and RWCFR vs. Number of Recommendations for Brightkite Dataset

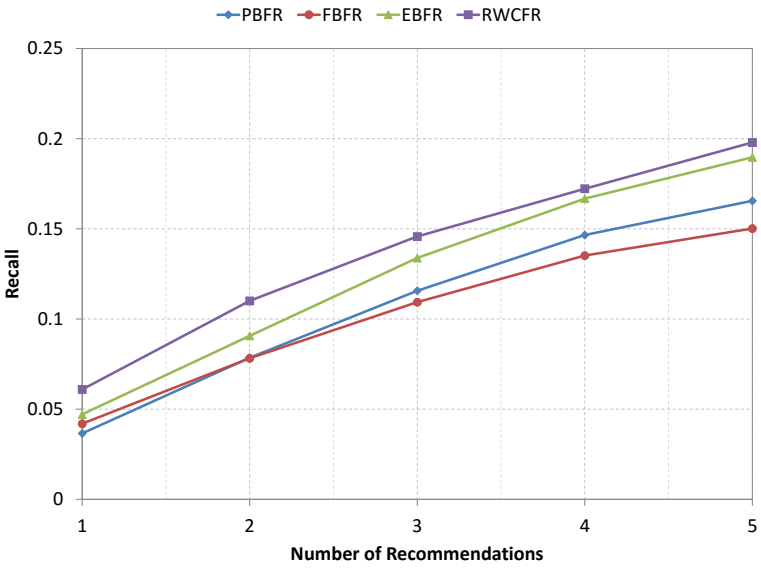



Figure 4.30: Recall Values of Baselines, and RWCFR vs. Number of Recommendations for Brightkite Dataset

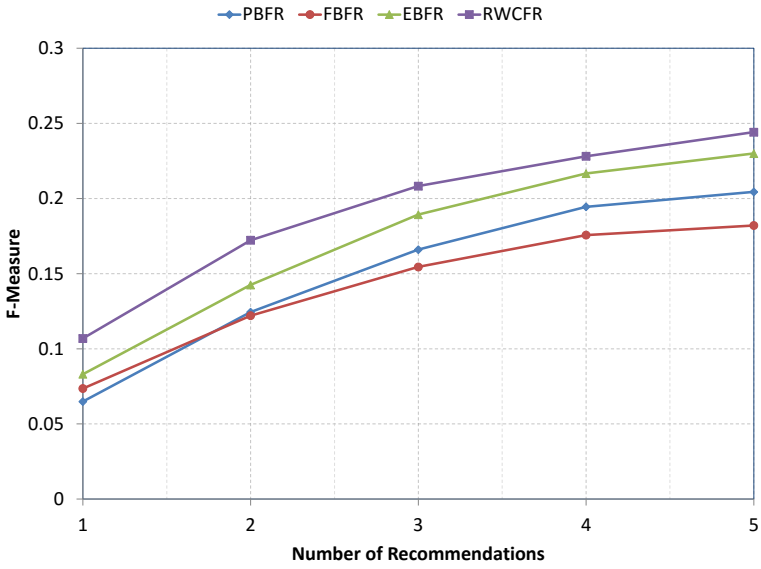



Figure 4.31: F-Measure Values of Baselines, and RWCFR vs. Number of Recommendations for Brightkite Dataset

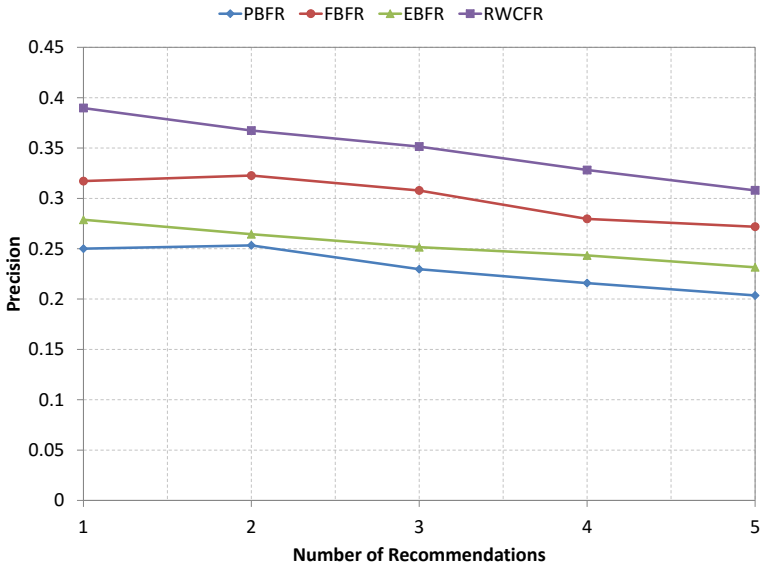



Figure 4.32: Precision Values of Baselines, and RWCFR vs. Number of Recommendations for Gowalla Dataset

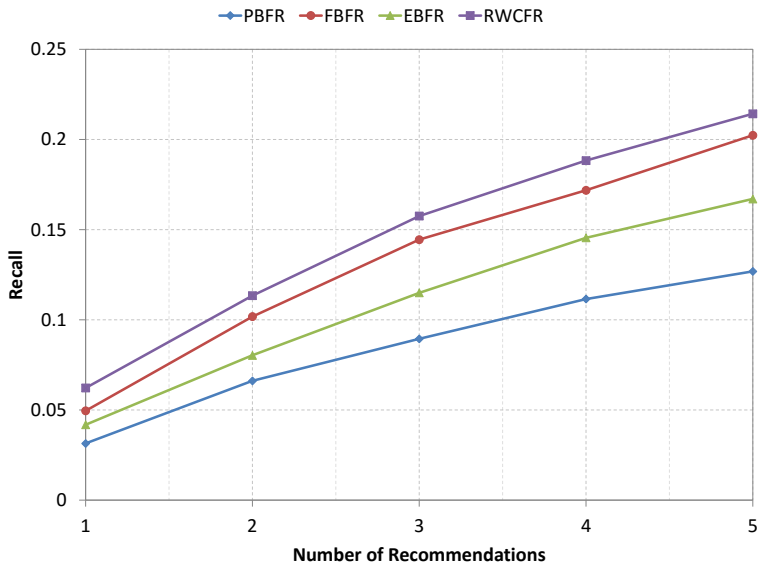



Figure 4.33: Recall Values of Baselines, and RWCFR vs. Number of Recommendations for Gowalla Dataset

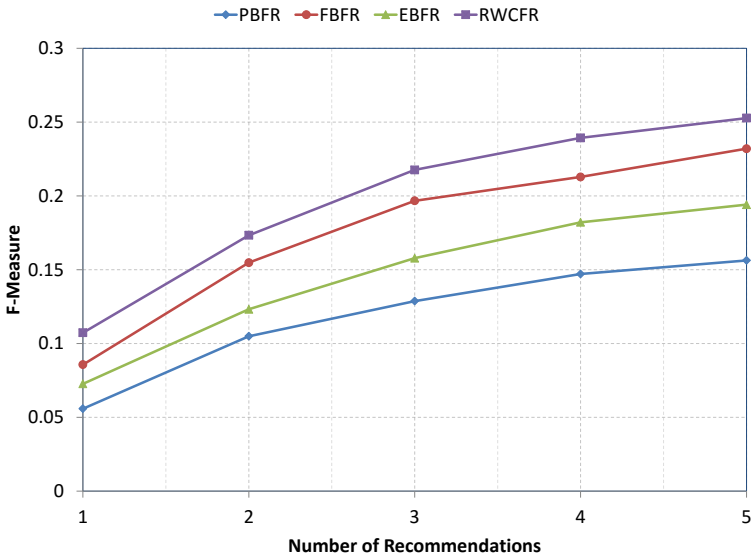



Figure 4.34: F-Measure Values of Baselines, and RWCFR vs. Number of Recommendations for Gowalla Dataset

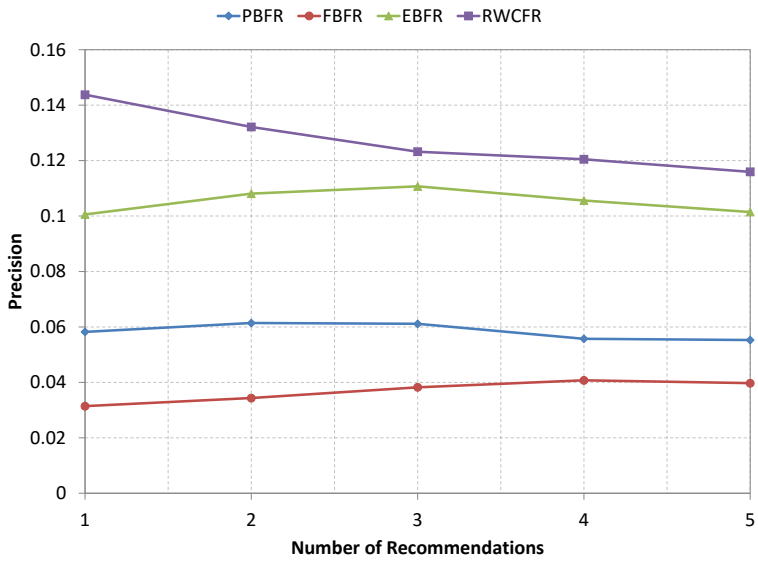



Figure 4.35: Precision Values of Baselines, and RWCFR vs. Number of Recommendations for Foursquare Dataset

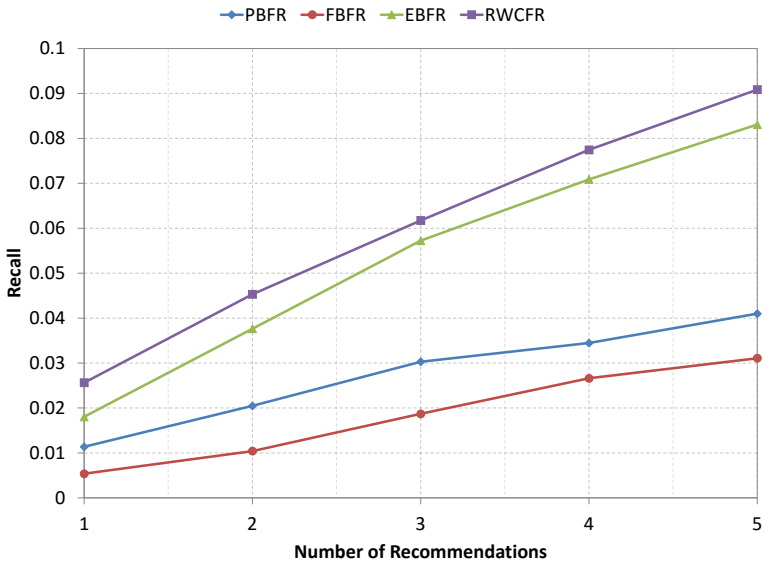



Figure 4.36: Recall Values of Baselines, and RWCFR vs. Number of Recommendations for Foursquare Dataset

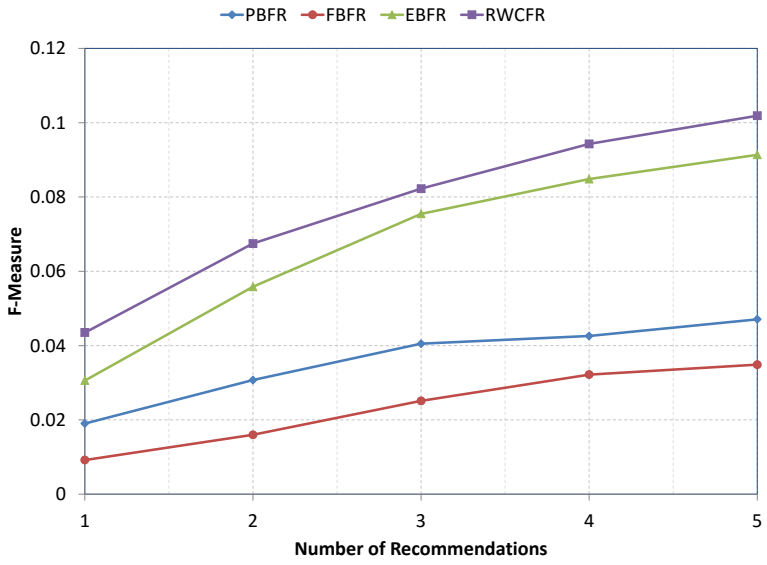



Figure 4.37: F-Measure Values of Baselines, and RWCFR vs. Number of Recommendations for Foursquare Dataset

In Brighkite and Foursquare experiments, FBFR has the lowest performance among all the algorithms. In Gowalla experiments, it is the second algorithm in terms of recommendation accuracy. However, it is still behind the performance of RWCFR. As mentioned before FBFR recommends the second degree friends. If a user has few friends, it is difficult for FBFR to recommend friend of friends. In other words, FBFR cannot span enough number of potential friends. Moreover, in the cases FBFR produces enough number of recommendations, the accuracy of it is very low. Hence, we can conclude that friend of friends are useful for friend recommendation but it is not sufficient for friend recommendation.

PBFR has the lowest performance in Gowalla experiments. In other experiments it is slightly better than FBFR but it still has lower performance than EBFR and RWCFR. Popularity-based techniques are simple and widely used in recommendation. Although it is a simple approach, the results produced by popularity-based approaches are reasonable because popular items target the majority of the audience. PBFR recommends the users that have check-ins in vicinity and have the highest number of friends. However, it does not consider social context (i.e. friendship links) of the user. Moreover, it does not ask the opinion of local experts. It also does not consider the personal preferences of the user. All these reasons put PBFR behind RWCFR. Popularity is still a reasonable friend recommendation approach because it produces better results than FBFR in most of the cases. However, popularity should be combined with other approaches to produce more accurate results, as in RWCFR.

EBFR asks the opinions of the local experts for friend recommendation. Local experts are very useful in location and activity recommendation. The friend recommendation results show that experts are also good at recommending friends to the users. It has the highest performance after RWCFR in Brightkite and Foursquare experiments. In Gowalla experiments it is the third algorithm in terms of recommendation accuracy. EBFR is a better choice in recommendation compared to FBFR and PBFR. However, EBFR is still not able to span enough friends for recommendation.

RWCFR considers popularity, local experts and second degree friends in friend recommendation. Moreover, it also take local history and place friends into consideration. All these data are combined with the help of proposed LBSN model and with the

power of random walk RWCFR produces more accurate results than all the baselines. RWCFR has the highest performance for each of the datasets. This results clearly indicates that RWCFR is a stable friend recommendation algorithm for LBSNs.

# CHAPTER 5

# CONCLUSION AND FUTURE WORK

The data collected from LBSNs can be utilized for building recommendation systems for users. In our study, we aim to build a recommendation system for LBSNs which can suggest locations, activities and friends to users according their current context. We propose a graph model to represent LBSN data. This model includes user, location and activity nodes and relationships between these nodes. Our LBSN model represents an LBSN as an undirected unweighted graph. By applying a random walk approach on the personalized versions of these graphs, we can rank locations, activities and users. Recommendation lists for each type can be populated based on these rankings. We introduce three novel context-aware recommendation algorithms, CLoRW, RWCAR and RWCFR for LBSNs. They recommend locations, activities and friends to users, respectively.

We evaluate our location recommendation algorithm, CLoRW, by comparing it with popularity-based, friend-based, expert-based baselines and CF-based location recommendation approach using three different LBSN datasets. Moreover, we compare CLoRW with a well-known location recommendation algorithm, USG. In the experiments, we use the subset of Brightkite, Gowalla and Foursquare datasets that are filtered for New York City. In order to measure the effectiveness of the algorithms, we employ precision and recall metrics. We also calculate the F-measure to compare precision and recall values.

Our proposed location recommendation algorithm outperforms all the baselines, CF-based approach and USG in all of the experiments. This shows that consideration of friends and experts, which defines the social context, yields better results. More-

over, when the previous location history is employed for location recommendation as in our algorithm, the prediction accuracy increases. According the results USG experiments, we realized that consideration of multiple influences is important, but it is more important to combine them using suitable a fusion framework. USG uses a linear model to fuse final results. On the other hand, we employ a graph model for fusion of input data (i.e. users, locations, experts) not the results. The experiment results show that our methodology of fusion is more effective than the methodology used in USG.

In order to evaluate our activity recommendation algorithm, RWCAR, we compare it with friend-based, expert-based and popularity-based activity recommendation approaches. We use the Foursquare LBSN dataset in evaluation. We further enrich this dataset with activity information using the Foursquare API. According to the results of the experiments, RWCAR outperforms all the baselines. This clearly indicates that consideration of user's location and activity history, social relations and local activity experts together in activity recommendation yields better results.

Similar to RWCAR evaluation, we compare our friend recommendation algorithm, RWCFR, with friend-based, expert-based and popularity-based friend recommendation baselines. In experiments, we employ Brightkite, Gowalla and Foursqaure datasets. RWCFR performs better than all the baselines for all of the datasets. This is because of that RWCFR is a multi-criteria algorithm similar to CLoRW and RWCAR. It considers personal, spatial and social context together and fuses this data using the proposed LBSN model.

One of the major strength of our recommendation methods is that they are suitable for dynamic environments. CF-based and model based recommendation approaches need to update their models and structures (e.g. tensors) periodically due to inclusion of new check-in data. On the other hand, CLoRW, RWCAR and RWCFR construct their local recommendation graph online using efficient graph queries and does not need to keep static models or structures that need to be updated periodically. Therefore, our recommendation algorithms can easily be employed in online systems without bringing extra processing overhead to server side.

All the results clearly point out the correctness of the LBSN model. Our proposed

LBSN model is easy to extend. Therefore, other LBSN elements can easily be added to this model. The results also indicate that random walk is a good choice for ranking the connected items on a graph.

As a future work, our work can be extended to provide recommendations for groups by extracting the common features of group members and adding those features as contextual information to our subgraph. In addition to this, our proposed recommendation algorithms can easily be employed in mobile applications which allows collecting more contextual information about the users. This contextual information can be used in subgraph construction to provide more accurate recommendation results.

# REFERENCES


[1] Hakan Bagci and Pinar Karagoz. Context-aware location recommendation by using a random walk-based approach. *Knowledge and Information Systems*, pages 1–20, 2015.

[2] Hakan Bagci and Pinar Karagoz. Random walk based context-aware activity recommendation for location based social networks. In *Data Science and Advanced Analytics (DSAA), 2015. 36678 2015. IEEE International Conference on*, pages 1–9, Oct 2015.

[3] Andrey Balmin, Vagelis Hristidis, and Yannis Papakonstantinou. Objectrank: Authority-based keyword search in databases. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 564–575. VLDB Endowment, 2004.

[4] Jie Bao, Yu Zheng, and Mohamed F Mokbel. Location-based and preference-aware recommendation using sparse geo-social networking data. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*, pages 199–208. ACM, 2012.

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

[6] B. Berjani and T. Strufe. A recommendation system for spots in location-based online social networks. In *Proceedings of the 4th Workshop on Social Network Systems*, pages 1–6. ACM, 2011.

[7] M. Brand. Incremental singular value decomposition of uncertain data with missing values. *Computer Vision-ECCV 2002*, pages 707–720, 2002.

[8] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[9] Soumen Chakrabarti, Byron Dom, Prabhakar Raghavan, Sridhar Rajagopalan, David Gibson, and Jon Kleinberg. Automatic resource compilation by analyzing hyperlink structure and associated text. *Computer Networks and ISDN Systems*, 30(1):65–74, 1998.

[10] E. Cho, S.A. Myers, and J. Leskovec. Friendship and mobility: user movement in location-based social networks. In *Proceedings of the 17th ACM SIGKDD*

*international conference on Knowledge discovery and data mining*, pages 1082–1090. ACM, 2011.

[11] Cheng-Hao Chu, Wan-Chuen Wu, Cheng-Chi Wang, Tzung-Shi Chen, and Jen-Jee Chen. Friend recommendation for location-based mobile social networks. In *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2013 Seventh International Conference on*, pages 365–370. IEEE, 2013.

[12] Rinner Claus and Raubal Martin. Personalized multi-criteria decision strategies in location-based decision support. *Geographic Information Sciences*, 10(2):149–156, 2004.

[13] Justin Cranshaw, Eran Toch, Jason Hong, Aniket Kittur, and Norman Sadeh. Bridging the gap between physical location and online social networks. In *Proceedings of the 12th ACM international conference on Ubiquitous computing*, pages 119–128. ACM, 2010.

[14] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, volume 96, pages 226–231, 1996.

[15] A. Etemad-Shahidi and J. Mahjoobi. Comparison between m5' model tree and neural networks for prediction of significant wave height in lake superior. *Ocean Engineering*, 36(15):1175–1181, 2009.

[16] S.L. Feld. The focused organization of social ties. *American journal of sociology*, pages 1015–1035, 1981.

[17] E. Frank, Y. Wang, S. Inglis, G. Holmes, and I.H. Witten. Using model trees for classification. *Machine Learning*, 32(1):63–76, 1998.

[18] Huiji Gao, Jiliang Tang, and Huan Liu. gscorr: Modeling geo-social correlations for new check-ins on location-based social networks. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 1582–1586. ACM, 2012.

[19] Floris Geerts, Heikki Mannila, and Evimaria Terzi. Relational link-based ranking. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 552–563. VLDB Endowment, 2004.

[20] Taher H Haveliwala. Topic-sensitive pagerank. In *Proceedings of the 11th international conference on World Wide Web*, pages 517–526. ACM, 2002.

[21] J.L. Herlocker, J.A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 230–237. ACM, 1999.

[22] M. Jamali and M. Ester. Trustwalker: a random walk model for combining trust-based and item-based recommendation. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 397–406. ACM, 2009.

[23] M. Jang and J.C. Sohn. Bossam: An extended rule engine for owl inferencing. *Rules and Rule Markup Languages for the Semantic Web*, pages 128–138, 2004.

[24] Glen Jeh and Jennifer Widom. Simrank: a measure of structural-context similarity. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 538–543. ACM, 2002.

[25] Jon M Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46(5):604–632, 1999.

[26] B.H. Lee, H.N. Kim, J.G. Jung, and G.S. Jo. Location-based service with context data for a restaurant recommendation. In *Database and Expert Systems Applications*, pages 430–438. Springer, 2006.

[27] K.W.T. Leung, D.L. Lee, and W.C. 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*, pages 305–314, 2011.

[28] Nan Li and Guanling Chen. Multi-layered friendship modeling for location-based mobile social networks. In *Mobile and Ubiquitous Systems: Networking & Services, MobiQuitous, 2009. MobiQuitous' 09. 6th Annual International*, pages 1–10. IEEE, 2009.

[29] C.D. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval*, volume 1. Cambridge University Press Cambridge, 2008.

[30] A. Noulas, S. Scellato, N. Lathia, and C. Mascolo. A random walk around the city: New venue recommendation in location-based social networks. In *2012 ASE International Conference on Social Computing*, 2012.

[31] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford University, 1998.

[32] A. Papadimitriou, P. Symeonidis, and Y. Manolopoulos. Friendlink: Link prediction in social networks via bounded local path traversal. In *Computational Aspects of Social Networks (CASoN), 2011 International Conference on*, pages 66–71. IEEE, 2011.

[33] A. Papadimitriou, P. Symeonidis, and Y. Manolopoulos. Geo-social recommendations. In *ACM Recommender Systems 2011 (RecSys) Workshop on Personalization in Mobile Applications*, 2011.

[34] M.H. Park, J.H. Hong, and S.B. Cho. Location-based recommendation system using bayesian user preference model in mobile devices. *Ubiquitous Intelligence and Computing*, pages 1130–1139, 2007.

[35] M. Pazzani and D. Billsus. Content-based recommendation systems. *The adaptive web*, pages 325–341, 2007.

[36] Sanjay Purushotham, C-C Jay Kuo, Junaith Shahabdeen, and Lama Nachman. Collaborative group-activity recommendation in location-based social networks. In *Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Crowdsourced and Volunteered Geographic Information*, pages 8–15. ACM, 2014.

[37] D. Quercia, N. Lathia, F. Calabrese, G. Di Lorenzo, and J. Crowcroft. Recommending social events from mobile phone location data. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 971–976. IEEE, 2010.

[38] Sasank Reddy, Min Mun, Jeff Burke, Deborah Estrin, Mark Hansen, and Mani Srivastava. Using mobile phones to determine transportation modes. *ACM Transactions on Sensor Networks (TOSN)*, 6(2):13, 2010.

[39] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Incremental singular value decomposition algorithms for highly scalable recommender systems. In *Fifth International Conference on Computer and Information Science*, pages 27–28. Citeseer, 2002.

[40] Masoud Sattari, Murat Manguoglu, Ismail H Toroslu, Panagiotis Symeonidis, Pinar Senkul, and Yannis Manolopoulos. Geo-activity recommendations by using improved feature combination. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, pages 996–1003. ACM, 2012.

[41] Norma Saiph Savage, Maciej Baranski, Norma Elva Chavez, and Tobias Höllerer. I am feeling loco: A location based context aware recommendation system. In *Advances in Location-Based Services*, pages 37–54. Springer, 2012.

[42] S. Scellato, A. Noulas, and C. Mascolo. Exploiting place features in link prediction on location-based social networks. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1046–1054. ACM, 2011.

[43] N. Srebro, T. Jaakkola, et al. Weighted low-rank approximations. In *MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-*, volume 20, page 720, 2003.

[44] Panagiotis Symeonidis, Alexis Papadimitriou, Yannis Manolopoulos, Pinar Senkul, and Ismail Toroslu. Geo-social recommendations based on incremental tensor reduction and local path traversal. In *Proceedings of the 3rd*

*ACM SIGSPATIAL International Workshop on Location-Based Social Networks*, pages 89–96. ACM, 2011.

[45] H. Tong, C. Faloutsos, and J.Y. Pan. Fast random walk with restart and its applications. In *Data Mining (ICDM), 2006 IEEE 6th International Conference on*, pages 613–622. IEEE, 2006.

[46] Wenting Tu, David W Cheung, Nikos Mamoulis, Min Yang, and Ziyu Lu. Activity-partner recommendation. In *Advances in Knowledge Discovery and Data Mining*, pages 591–604. Springer, 2015.

[47] J. Wang, A.P. De Vries, and M.J.T. Reinders. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 501–508. ACM, 2006.

[48] Zhibo Wang, Jilong Liao, Qing Cao, Hairong Qi, and Zhi Wang. Friendbook: a semantic-based friend recommendation system for social networks. *Mobile Computing, IEEE Transactions on*, 14(3):538–551, 2015.

[49] Jihang Ye, Zhe Zhu, and Hong Cheng. What's your next move: User activity prediction in location-based social networks. In *Proc. of SIAM International Conference on Data Mining (SDM)*, 2013.

[50] 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.

[51] J.J.C. Ying, E.H.C. Lu, W.N. Kuo, and V.S. Tseng. Urban point-of-interest recommendation by mining user check-in behaviors. In *Proceedings of the ACM SIGKDD International Workshop on Urban Computing*, pages 63–70. ACM, 2012.

[52] Xiao Yu, Ang Pan, Lu-An Tang, Zhenhui Li, and Jiawei Han. Geo-friends recommendation in gps-based cyber-physical social network. In *Advances in Social Networks Analysis and Mining (ASONAM), 2011 International Conference on*, pages 361–368. IEEE, 2011.

[53] Y.H. Yu, J.H. Kim, K. Shin, and G.S. Jo. Recommendation system using location-based ontology on wireless internet: An example of collective intelligence by using 'mashup' applications. *Expert systems with applications*, 36(9):11675–11681, 2009.

[54] V.W. Zheng, B. Cao, Y. Zheng, X. Xie, and Q. Yang. Collaborative filtering meets mobile recommendation: A user-centered approach. In *Proceedings of the 24rd AAAI Conference on Artificial Intelligence*, volume 10, pages 236–241, 2010.

[55] V.W. Zheng, Y. Zheng, X. Xie, and Q. Yang. Collaborative location and activity recommendations with gps history data. In *Proceedings of the 19th international conference on World wide web*, pages 1029–1038. ACM, 2010.

[56] V.W. Zheng, Y. Zheng, X. Xie, and Q. Yang. Towards mobile intelligence: Learning from gps history data for collaborative recommendation. *Artificial Intelligence*, 2012.

[57] Y. Zheng. Location-based social networks: Users. *Computing with Spatial Trajectories*, pages 243–276, 2011.

[58] Y. Zheng. Tutorial on location-based social networks. *WWW2012*, 2012.

[59] Y. Zheng, Y. Chen, X. Xie, and W.Y. Ma. Geolife 2.0: a location-based social networking service. In *Mobile Data Management: Systems, Services and Middleware, 2009. MDM'09. Tenth International Conference on*, pages 357–358. IEEE, 2009.

[60] Y. Zheng, X. Xie, and W.Y. Ma. Geolife: A collaborative social networking service among user, location and trajectory. *IEEE Data Engineering Bulletin*, 33(2):32–40, 2010.

[61] Y. Zheng, L. Zhang, X. Xie, and W.Y. Ma. Mining interesting locations and travel sequences from gps trajectories. In *Proceedings of the 18th international conference on World wide web*, pages 791–800. ACM, 2009.

# CURRICULUM VITAE

## PERSONAL INFORMATION

**Surname, Name:** Bağcı, Hakan
**Nationality:** Turkish (TC)
**Date and Place of Birth:** 07.04.1984, Ankara
**Marital Status:** Married
**Email:** hbagci@gmail.com

## EDUCATION

| Degree | Institution | Year of Graduation |
|---|---|---|
| M.S. | Middle East Technical University | 2010 |
| B.S. | Bilkent University | 2006 |

## PROFESSIONAL EXPERIENCE

| Year | Place | Enrollment |
|---|---|---|
| 2006-Present | TÜBİTAK BİLGEM İLTAREN | Senior Researcher |

## PUBLICATIONS

### Journal Publications

1. Hakan Bagci, Pinar Karagoz, Context-aware location recommendation by using a random walk-based approach, Knowledge and Information Systems, First Online: 15 July 2015, Online ISSN 0219-3116, 2015.

2. Seyyit Alper Sert, Hakan Bagci, Adnan Yazici, MOFCA: Multi-objective fuzzy clustering algorithm for wireless sensor networks, Applied Soft Computing, Volume 30, May 2015, pages 151-165, ISSN 1568-4946, 2015.

3. Hakan Bagci, Adnan Yazici, An energy aware fuzzy approach to unequal clustering in wireless sensor networks, Applied Soft Computing, Volume 13, Issue 4, April 2013, pages 1741-1749, ISSN 1568-4946, 2013.

**International Conference Publications**

1. Hakan Bagci, Pinar Karagoz, Random walk based context-aware activity recommendation for location based social networks, in Data Science and Advanced Analytics (DSAA), 2015 IEEE International Conference on, pages 1-9, Oct 2015.

2. Hakan Bagci, Adnan Yazici, An energy aware fuzzy unequal clustering algorithm for wireless sensor networks, in Fuzzy Systems (FUZZ), 2010 IEEE International Conference on, pages 1-8, July 2010.