TRUST-AWARE LOCATION RECOMMENDATION IN LOCATION-BASED SOCIAL NETWORKS

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

BY

DENIZ CANTÜRK

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

AUGUST 2021

Approval of the thesis:

TRUST-AWARE LOCATION RECOMMENDATION IN LOCATION-BASED SOCIAL NETWORKS

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

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Halit Oğuztüzün
Head of Department, **Computer Engineering** _____

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

Examining Committee Members:

Prof. Dr. Ali Doğru
Computer Engineering, METU _____

Prof. Dr. Pınar Karagöz
Computer Engineering, METU _____

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

Prof. Dr. Osman Abul
Computer Engineering, TOBB ETU _____

Assist. Prof. Dr. Engin Demir
Computer Engineering, 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, Surname: Deniz Cantürk

Signature :

ABSTRACT

TRUST-AWARE LOCATION RECOMMENDATION IN LOCATION-BASED SOCIAL NETWORKS

Cantürk, Deniz
Ph.D., Department of Computer Engineering
Supervisor: Prof. Dr. Pınar Karagöz

August 2021, 104 pages

Users can share their location with other social network users through location-embedded information in LBSNs (Location-Based Social Network). LBSNs contain useful resources, such as user check-in activities, for building a personalized recommender system. Trust in social networks is another important concept that has been integrated into a recommendation system in various settings. In this thesis, we propose two novel techniques for location recommendation, *TLoRW* and *SgWalk*, to improve recommendation performance through integrated trust information. In both of the algorithms, the elements of LBSN and their relationships (user-user, user-location) are represented by using a graph model. For trust modeling, we develop a method to generate trust scores of LBSN users. With the developed method, the global trust score of a user is predicted with respect to the check-in history. The trust model is integrated into the LBSN graph model to be used within the proposed location recommendation algorithms. The first algorithm, *TLoRW*, generates location recommendations based on the user's current location by exploiting the friendships, experts, and trusted users traversing the region of user's spatial context through a random walk based algorithm. This region is constructed as the subgraph of the user according to the current loca-

tion. In the second recommendation algorithm, *SgWalk*, we consider user subgraph as a heterogeneous information network and propose a novel HIN embedding technique. The location recommendation is generated by the proximity between users and locations based on their corresponding node embedding. *SgWalk* is differentiated from the previous node embedding techniques relying on meta-path or bi-partite graphs by utilizing the user subgraphs generated based on spatial context. By this way, it is aimed to capture the relationship between the entities with respect to the spatial context.The recommendation performance of *TLoRW* and *SgWalk* is analyzed through extensive experiments conducted on benchmark datasets by evaluating the accuracy in top-k location recommendations. The experiments reveal that trust information has a significant effect on improving the location recommendation performance. The performance evaluation results show a substantial improvement compared to baseline techniques and the state-of-the-art trust-aware recommendation and heterogeneous graph embedding techniques in the literature.

Keywords: Location-based Social Networks, Location Recommendation, Heterogeneous Information Network Embedding, Information Fusion, Trust Prediction, Trust-aware Recommendation, Random Walk

ÖZ

KONUM TEMELLİ SOSYAL AĞLARDA GÜVEN FARKINDA KONUM ÖNERİSİ

Cantürk, Deniz
Doktora, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. Pınar Karagöz

Ağustos 2021 , 104 sayfa

Kullanıcılar, LBSN'lerde (Konum Tabanlı Sosyal Ağ) konuma gömülü bilgiler aracılığıyla konumlarını diğer sosyal ağ kullanıcılarıyla paylaşabilmektedir. Kullanıcıların konumlar için yaptıkları giriş bilgilerinden oluşan sonuç veri seti kişiselleşmiş konum öneri sistemi inşa etmek için kullanılmaktadır. Güven, sosyal ağlara çeşitli ortamlarda öneri sistemlerine entegre edilmiş bir başka önemli kavramdır. Bu tezde, öneri performansını iyileştirmek için konum önerisi için iki yeni teknik olan *TLoRW* ve *SgWalk* öneriyoruz. İleri sürülen yaklaşımlarda LBSN öğeleri (kullanıcı- kullanıcı, kullanıcı-konum) ve aralarındaki ilişkiler çizge modeli kullanılarak temsil edilmektedir. Güven modellemesi için, LBSN kullanıcılarının güven puanlarını tahmin eden bir yöntem geliştirdik. Geliştirilen yöntemle, bir kullanıcının check-in geçmişine göre global güven puanı tahmin edilmektedir. Güven modeli, önerilen konum öneri algoritmalarında kullanılmak üzere LBSN grafik modeline entegre edilmiştir. İlk algoritma, *TLoRW*, rastgele yürüyüş tabanlı bir algoritma olup kullanıcının uzamsal bağlam bölgesini dolaşarak arkadaşlıklardan, uzmanlardan ve güvenilir kullanıcılardan yararlanarak kullanıcının mevcut konumuna dayalı konum önerileri üretir. Bu uzamsal bağlam bölge,

mevcut konuma göre kullanıcının altçizgesi olarak oluşturulur. İkinci öneri algoritmasında, *SgWalk*, kullanıcı altçizgesini heterojen bir bilgi ağı olarak ele alarak yeni bir HIN yerleştirme tekniği öneriyoruz. Konum önerileri, kullanıcılar ve konumlar arasındaki yakınlığa bağlı olarak yapılan düğüm yerleştirmelerine dayalı olarak oluşturulmaktadır. *SgWalk*, uzamsal bağlama dayalı olarak oluşturulan kullanıcı altçizgelerini kullandığı için meta-path veya iki parçalı çizgelere dayanan önceki düğüm yerleştirme tekniklerinden farklıdır. Bu sayede, varlıklar arasındaki ilişkinin mekânsal bağlam açısından yakalanması amaçlanmaktadır *TLoRW* ve *SgWalk*'un öneri performansı, ilk k konum önerilerindeki doğruluk değerlendirilerek bilinen veri setleri üzerinde gerçekleştirilen kapsamlı deneyler aracılığıyla analiz edildi. Deneyler, güven bilgilerinin konum önerisi performansını iyileştirmede önemli bir etkiye sahip olduğunu ortaya koymaktadır. Performans değerlendirme sonuçları, literatürde temel teknikler ve son zamanlarda yayınlanan güvene duyarlı öneri ve heterojen grafik gömme tekniklerine kıyasla önemli bir gelişme olduğunu göstermektedir.

Anahtar Kelimeler: Konum Tabanlı Sosyal Ağlar, Konum Önerisi, Heterojen Bilgi Ağı Gömme, Bilgi Füzyonu, Güvenilirlik Tahminlemesi, Güven-Farkında Konum Önerisi, Rastgele Yürüyüş

To my precious wife Zeynep CANTURK and kids

ACKNOWLEDGMENTS

I want to start with sincere thanks to my supervisor, Prof. Dr. Pınar Karagöz, for her helpfulness, guidance, and suggestions throughout my thesis studies.

I would like to show my appreciation to Prof. Dr. Ali Doğru, Assist. Prof. Dr. Gönenç Ercan and Assist. Prof. Dr. Engin Demir, the prior and current member of the thesis monitoring committee, for their supports and feedback on my research studies.

I acknowledge special thanks to the members of defense jury Prof. Dr. Hakkı Toroslu and Assoc. Prof. Dr. Osman Abul, for reviewing and commenting on my thesis.

I am glad to mention that this work is funded by the Scientific and Technological Research Council of Turkey (TUBITAK) with grant number 118356.

I thank my daughter and son for their patience during my thesis studies.

I want to conclude by sending the most excellent thanks to my wife, Zeynep CAN-TÜRK, for her love, endless support, and encouragement during my graduate studies.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

ABBREVIATIONS	
CDL	Collaborative Deep Learning
CF	Collaborative Filtering
CLR	Collaborative Location Recommendation
DBSCAN	Density-based Spatial Clustering of Applications with Noise
ETAF	Extended Trust Ancestor Framework
EU	Expert User
GeoMF	Geographical Modeling and Matrix Factorization
GE	Graph Embedding
GPS	Global Positioning System
HIN	Heterogeneous Information Network
HITS	Hypertext Induced Topic Search
JLGE	Jointly Learn the Graph Embeddings
JUST	Jump or Stay
LBSN	Location Based Social Network
LINE	Large-scale Information Network Embedding
METU	Middle East Technical University
MF	Matrix Factorization
PL	Popular Location
PTE	Predictive Text Embedding
SDAE	Stacked Denoising Autoencoder
SH-CDL	Spatial-Aware Hierarchical Collaborative Deep Learning
SHINE	Signed Heterogeneous Information Network Embedding

SVD	Singular Value Decomposition
TAF	Trust Ancestor Framework
TECF	Trust-Enhanced Collaborative Filtering
TLoRW	Trust-aware Location Recommendation with Random Walk
TU	Trustable User

CHAPTER 1

INTRODUCTION

1.1 Motivation and Problem Definition

The rapid growth of the internet and mobile communication led to the development of social networks on the web, and advancements in positioning technology allow the utilization of location data through social networks in various ways, such as sharing an instant location with other users or exchanging travel experiences with friends. These opportunities foster sharp increases in LBSNs (location-based social networks), such as Foursquare[1] and GeoLife[2].

An LBSN stores the users' check-in history, which is a precious resource for a recommender system to recommend a variety of information, such as friends, locations, and events, according to the user's context. Given a check-in history from an LBSN as a collection of tuples consisting of (user, check-in date and time, location information) and the user's social network connections, a list of locations can be suggested by the location recommender for the LBSN to the target user. The quality of the recommendation can be improved with additional information from LBSNs, such as users' demographic information or the performed activity type[1].

In this thesis work, LBSN is considered as the basis and the problem of location recommendation is challenged within this setting. More specifically, it is aimed to provide accurate location recommendations with respect to the current location (spatial context) of a given user. Furthermore, trust concept is modeled for LBSN setting and integrated into LBSN model to improve recommendation accuracy. Consequently,

[1] https://www.foursquare.com/
[2] https://www.microsoft.com/en-us/research/publication/geolife-gps-trajectory-dataset-user-guide/

we propose two approaches for location recommendation. In the first approach, we represent LBSN data (the users and the locations visited) as vertices and visiting relations as edges of a graph model and rank the nodes on the graph for generating recommendations. In this approach, random walk is applied on the generated subgraph to estimate the ranking value of the locations. Then, locations are sorted according to their ranking values, and the top-n locations are recommended to a user. In the second approach, we construct a user subgraph that considers a wide range of information sources, such as personal, social, spatial, and trustworthiness contexts, and generated random walk sequences over the user subgraph. Then, we learn the graph embeddings of nodes in the LBSN and generate a location recommendation list from the vector representation of the nodes.

1.2 Proposed Methods and Models

Trust is a phenomenon that has been extensively studied in several fields and utilized in various contexts as a basis for decision-making. Although these fields define trust differently, the problems they attempt to solve share the common objective of accurately assessing trust as a sound basis for decision-making, where an incorrect estimation of trust may lead a trustor to set a wrong trust to a trustee, resulting in betrayal by the trustee or loss of opportunities for good collaborators [2, 3]. Because all required knowledge is unavailable in several problem areas, crucial decisions are generally made with insufficient, incomplete, uncertain, and conflicting information. This vagueness leads a decision-maker to risk negative output from fallacious choices because of the possibility of misplaced trust in another object.

The dictionary definition of trust is "to have confidence in someone; to believe that someone is good, sincere, honest, etc."[3]. Simply, trust is a relationship in which an object, usually called a trustor, relies on something or someone called a trustee based on some conditions. Since trust is an interdisciplinary concept, the term has been used in various fields to model varying types of relationships. To derive an interdisciplinary definition of trust that includes these concepts, we review the definitions of trust in various fields and reach a common definition of trust.

[3] https://www.oxfordlearnersdictionaries.com/us/definition/english/trust_2

The social sciences work on the trust relationships of people in a general social setting. In this setting, people have expectations about the behavior of others. In a classic definition from sociology, [4] defines trust as the trustor's subjective probability of whether the trustee (or trustees) will perform a particular action to the trustor's advantage. Probability is assessed before the trustee takes action according to the uncertain conditions about the trustee. In philosophy, trust is an individual, internal phenomenon that helps to preserve ethical relationships between people, and breach of trust is an obvious violation of ethical behavior that leads to distrust [5]. In psychology, trust is described as a cognitive construct by Rotter [6] that a person learns from social experiences as the outcomes of trusting behavior in a positive or negative manner. It is concluded that a person who has had negative experiences of greater trust in the past is unlikely to trust in the future, and vice versa. Trust propensity is a recognized concept that reveals differences in the degrees of trust among people in the same circumstance. In particular, trust propensity is strongly influenced by who or what the trustee is. In computing, trust is an important concept and is applied to many different areas, including artificial intelligence, telecommunications, and cyber/network security. As a subjective judgment, an agent's trust is dependent on their belief that another object will behave reliably in a given context. According to the agent's experience, it can maximize its interests (or utility) or reduce its risks [7, 8].

As a result of the common themes across disciplines, the concept of trust can be summarized as follows:

Trust is a relationship where an individual takes a risk, based on a subjective belief that a trustee will involve in reliable behavior to maximize their interests under ambiguous, conflicting, and incomplete information of a given circumstance based on the cognitive assessment of actions taken according to the trustee. [2]

We used trust in our thesis research in the meaning of the above definition.

Trust is a relationship between trustor and trustee, and the strength of the relationship is determined by the decision that results in positive or negative outcomes is called trustworthiness, which represents objective trust based on the observed outcome. Conversely, overall trust can be formed by combining subjective and objective trust. Basically, trustworthiness refers to trust that is validated by evidence. Zarghami

et al. [9] introduced a metric, the T-index, to estimate a user's trustworthiness as a recommendation maker in the same way as the h-index is used by researchers community. In our research, we do not have trust relationships between users and calculate the *global trustworthiness* scores of users and identify globally *trustable users (TUs)*

In addition to the trust relationship, *trustworthiness* of the users also has the potential to improve the recommendation quality. To identify *TUs*, we employed users' relevant properties to determine their trustworthiness by utilizing feature engineering techniques [10]. According to computational social science investigations [11], which analyze online information characteristics, the loyalty of a user's attitude can be inferred as follows. When a user who does not have prejudgements maintains an objective attitude (i.e., objectivity) and provides a consistent perspective (i.e., consistency), loyalty tends to increase [12]. In addition, when a user frequently communicates with other users by accessing the data produced by him or her (i.e., activity), the reliability tends to increase as well. Therefore, *activity, objectivity* and *consistency* can be considered the basic features to recognize TUs.

The trust concept has been studied in recommendation systems with the opinion that each user's attitude is influenced by the user's trust connections, and hence, various trust-aware recommendation approaches have been proposed [13, 14, 15, 16, 17, 18, 19, 20, 21, 22]. According to the theoretical and experimental results of these proposed approaches, it has been reported that making use of trust relationships improves recommendation performance. In the studies considering *explicit trust*, the trust score explicitly declared by users is utilized [13, 14, 15, 16, 19, 21, 22]. However, for some other solutions, the *implicit trust* score is used in the form of friendship relations between the users [14, 15, 16, 17, 18]. In LBSN models explicit trust information is rarely available. In this thesis work, we model implicit trust in terms objectivity and consistency of TUs and incorporate into LBSN model. This extended LBSN model is used in both of the proposed algorithms in the form of a graph. The first algorithm, TLoRW, makes use of random walk on the trust extended graph to generate location recommendations. The same extended graph model is also used in the second algorithm, SgWalk, in order to generate node embeddings, which are to be used for recommending locations.

Given a check-in history of an LBSN and the user's social network connections, the location recommender for an LBSN attempts to recommend locations. As the numbers of users and locations in an LBSN are too high, processing and analyzing formed social networks requires considerable effort. To overcome this issue, network embedding has attained popularity in recent years with the idea of representing graph nodes (vertices) with vectors of low-dimensional space that preserve the graph structure and its fundamental properties by generating a continuous vector representation of nodes. Hence, *network embedding* can provide the capability to identify the social characters closer to each other according to their corresponding vector representations. This leads to the adoption of network embedding approaches in various graph analysis tasks by considering node distances (similarities).

Recent graph embedding approaches can be grouped as homogeneous or heterogeneous graph embedding into two categories, according to the types of vertices in the graphs[23]. In a homogeneous graph, which only contains nodes belonging to a single notion (such as users in an LBSN), graph embedding techniques use sequences of nodes obtained as random samples from a graph (by performing random walks through the graph [24]) to preserve the proximity between node pairs. However, a heterogeneous graph has a complicated structure that includes nodes from several notions and edges, both homogeneous and heterogeneous, for connecting nodes from the same notion or relating nodes from different notions. [25, 26]. For instance, an LBSN graph with user (U), location (L), and friend (F) notions contains the homogeneous edges between users (friendship) as well as heterogeneous edges linking a user and location (visits). These types of graphs are also called HINs (*heterogeneous information networks*). Moreover, HINs have richer semantics and embody more information than homogeneous graphs. This phenomenon presented a new viewpoint to heterogeneous data analysis and drew the attention of many researchers.

With the basic idea, a heterogeneous graph can be directly embedded by homogeneous graph embedding techniques, but it suffers from poor results [23]. Therefore, network embedding techniques for heterogeneous graphs need to consider the information fusing methodology and structure of the graph while generating node sequences. Many studies have been performed on the techniques for HIN embedding [27, 28, 29, 30, 25, 26, 19, 31] and have proven the usefulness of HIN embedding

in HIN analysis tasks. In such studies, some of the techniques rely on metapaths to preserve the similarity between nodes while generating node sequences in a random walk [25, 26, 31]. Other techniques break down the HIN into smaller networks (bipartite graphs) and then jointly learn the embeddings of each subnetwork [29, 30, 32]. Additionally, there are solutions that learn nonlinear mapping functions by employing neural network-based approaches for HIN embedding [27, 33, 19, 34]. HIN embedding learned by the aforementioned techniques has been used in many tasks, such as clustering [35], classification [36], recommendation [37, 38], and link prediction [39].

In this thesis, we propose two approaches for location recommendation. In the first approach, TLoRW, we represent LBSN data (the users and the locations visited) as vertices and visiting relations as edges of a graph model and rank the nodes on the graph for generating recommendations. In this approach, trust scores are also attached to user nodes, and all locations' ranking values are estimated by employing a random walk on the generated subgraph. Then, locations are sorted according to estimated ranking values, and desired top-n locations are recommended to a user. Second approach, SgWalk, includes a novel use of node embedding model for HINs. From the LBSN graph, which is a HIN by its nature consisting of a variety of node and edge types, subgraphs are generated based on spatial contexts within the data. Node embeddings are learned by using walk sequences obtained on subgraphs. The motivation behind using such contextual subgraphs is to be able to capture the relationships between the entities (i.e. the edges) with respect to different spatial context. The similarities between a user node and location nodes in terms of their embeddings determine the locations to be recommended.

1.3 Contributions and Novelties

We can highlight the contributions of this thesis as follows:

- We propose two novel location recommendation techniques, *TLoRW* and *SgWalk*.
- We propose a trust model for LBSN users based on activities. Instead of ex-

plicit trust scoring or personal preferences, we develop a latent trust model, leveraging the features of objectivity, consistency and activity.

- We extend the LBSN user subgraph model by including trusted user nodes.
- We propose a random walk-based location recommendation algorithm on the trust-aware graph model, *Trust-aware Location Recommendation with random Walk (TLoRW)*, such that social connections, trustworthiness, user preferences and spatial context are employed together.
- To analyze the recommendation accuracy of the *TLoRW*, we conduct extensive experiments on four real-world datasets. The results show that *TLoRW* outperforms the state-of-the-art recommendation methods, such as collaborative deep learning and collaborative filtering.
- We propose,*SgWalk* (***S**ubg**r**aph **Walk***), a user subgraph-based graph embedding technique, to eliminate the links in the whole graph representing LBSNs that are not related to the user requesting a recommendation. Current embedding techniques in the literature use the whole graph to generate walk sequences that cause unrelated nodes to occur together in the embedding window and obtain low embedding performance. We solve this problem by constructing subgraphs specific to each user according to the user preferences and social connections.
- Based on the random walk sampling process on the user subgraph, we adopt the popular word embedding technique in a new setting for heterogeneous information network embeddings.
- We create heterogeneous information networks that consist of trustworthiness, user preferences, social connections, and spatial-context notions and fuse them successfully in learning graph embedding.
- *SgWalk* utilizes user subgraphs instead of bipartite subgraphs proposed in the literature and does not depend on metapaths.
- To analyze the recommendation accuracy of *SgWalk*, we perform extensive experiments on four real-world datasets. The results indicate that *SgWalk* outperforms the state-of-the-art graph embedding methods.

1.4 The Thesis Outline

The rest of the thesis is organized as follows. Related works are summarized in Section 2. We present the LBSN model, trust model and details of the proposed techniques in Section 3. We report our conducted experiments and the results in Section 4. The paper is concluded in Section 5 with final remarks.

CHAPTER 2

RELATED WORK

In this chapter, we provide a summary of the previous work related to location recommendation from LBSN data, the trustworthiness of users in recommendation systems and heterogeneous information network embedding.

2.1 Location Recommendation in LBSNs

Mainly, location recommendation approaches are built on top of collaborative filtering-based methods [40, 41, 42, 43, 44, 45]. Current developments in machine learning with deep neural network models allow studies to use deep learning methods in recommendation systems [27, 33, 46, 47, 18, 48]. In addition, other methods, such as random walks, decision trees, and Bayesian networks, are also applied in location recommendation [49].

Zheng et al. [42] used GPS trajectory data to recommend location and activity. In their approach, GPS history data are converted to a location-activity matrix, and then the matrix is factorized to recommend locations for a given activity, or vice versa. Additionally, in [43], the authors proposed a social networking service, namely, GeoLife 2.0, that provides an opportunity for sharing their travel experiences with other users. GeoLife aims to model GPS data, locations, and users and determine the similarity between users and locations. The predicted similarity values are employed for friend recommendations. Leung et al. [50] proposed the collaborative location recommendation (CLR) framework, which recommends locations by using user GPS trajectory data. In their work, the user-activity-location tripartite graph and a coclustering algorithm were employed to represent GPS trajectory data. Recommendation refinement

was performed by using clusters of similar locations for a particular activity issued by a specific user.

Cho et al. [41] investigated human mobility patterns on location-based social networks through temporally and geographically periodic movement with the social network structure. They developed a model of human mobility for predicting future locations by using the check-in similarity of friends. Additionally, Wang et al. [51] included a temporal cyclic effect with the sequential influence of user check-in times and the corresponding visited locations to improve recommendation accuracy.

Lian et al. [40] proposed a weighted matrix factorization-based location recommendation technique, namely, GeoMF. In [40], users' and locations' latent feature vectors were augmented with area vectors of user activities or location influence, respectively. Additionally, the GeoMF approach explains why integrating geographical influence into matrix factorization is beneficial to location recommender systems.

Wang et al. [27] proposed the collaborative deep learning (CDL) approach, which extracts the deep feature representation from content and captures the implicit relationship and proximity between users (and locations) by performing deep learning collaboratively. The learned representation is used for location recommendations. In [33], Yin et al. proposed a spatial-aware hierarchical collaborative deep learning model (SH-CDL), which jointly learns deep representation from heterogeneous location features and hierarchical additive representation of spatially-aware personal preferences. The authors aimed to overcome issues such as cold start and data sparsity that cause performance loss in collaborative filtering-based methods.

2.2 Trust Detection and Trust-aware Recommendation

Trust is a significant concept for enhancing recommendation quality. In recent studies, there have been many proposed approaches for trust-aware recommendation in different settings. In a social network that contains trust relations, the *trustor* is a user who trusts another user, and the *trustee* is a user who is trusted by another user. In a trust network, when explicit *distrust* is provided as well as explicit trust, such networks are considered signed trust networks. *Trust/distrust* is a one-directional re-

lationship between two users indicating that both agree/disagree on the feedback of the same item. In other words, a user has a similar/dissimilar impression on the item as the other user [15].

Guo et al. [14] proposed a trust-based matrix factorization method to address the cold start and data sparsity problems in the trust-aware recommendation system. To improve the accuracy, a trust propagation mechanism is incorporated into the model. In this method, explicit trust and the implicit influence of ratings are merged in the recommendation process. In [15], Ali et al. proposed a trust-aware recommendation method for utilizing both implicit and explicit trust relationships. The implicit trust relationship is inferred from rating information. In this way, the trust relationship sparsity problem is alleviated as well. In [52], multiple trustor and trustee relationships in trust networks were modeled to mine more information from social networks. In this approach, users are modeled in two separate models for the roles of trustors and trustees, and then the results from both models are incorporated to make recommendations. In [53], the trust-based competitive influence diffusion model was established to simulate the dynamic spreads of competitive influence. Trust and distrust relationships were used to model positive or negative influence, respectively, and trust values were estimated through generalized network flows.

Mayer et al. [54] proposed the *trust ancestor framework (TAF)*, which includes four trust factors for trust evaluation. These four factors are the trustee's *benevolence, ability, integrity* and the trustor's trust *propensity*. In [55], Guo et al. extended the TAF model (called *ETAF)* by using all interactions among users with the target trustees and obtaining the global trustworthiness. The ETAF model, as in TAF, is based on the four trust factors, but here, each factor can be formalized into local and global perspectives.

Oh and Kim [10] proposed an approach to identify and exploit trustworthy users with robust features in an online rating system. The candidate features to determine trust are the activity, objectivity, and consistency. The values for each of the features are normalized, and the influence of each feature is measured according to the correlation between the trustworthiness and the feature. In [56], trust was measured by using an uncertainty distribution variable that represents the trust relationship. In the approach,

single-path trust chains are established, and missing trust information is derived from the uncertainty distribution.

In the proposed work, we extend the concepts in Oh and Kim's [10] study to define trust within the LBSN context. However, the setting of our work is different. In [10], the authors focused on identifying trustworthy users, and they did not calculate the trust scores. Additionally, the authors did not employ the identified trustworthy users in recommendation generations. Guo et al. [55] used a similar methodology to our approach, such that they used formulas to find trust scores based on trust factors, mainly depending on the items' comments or ratings. However, in our setting, LBSN data do not include ratings for locations given by users. We calculate location ratings on the spot, so ETAF does not apply to our case. The problem setting of our work is different from those given in Ali et al.'s studies. Because, in our work, there are no available trust data in the dataset, we use generated trust scores in the recommendation. In contrast, Ali et al. employed existing trust data to fill the trust value gaps by utilizing the trust propagation approach. Furthermore, [18] and [57] used different techniques to estimate the trust values. To summarize, our approach is the first work that includes the trust node in an LBSN graph as an independent context and uses the trust node with a random walk technique for location recommendation.

2.3 Heterogeneous Information Network Embedding

The goal of network embedding is finding the network's low-dimensional vector representation by protecting the structure of the network and properties of the nodes in the network [58, 23, 47]. Hence, network embedding can be considered a dimension-reducing method. In early studies, [59, 60], dimensionality reduction techniques decomposed the network to learn the latent representations of vertices and edges in terms of low-dimensional vectors. For example, Ahmed et al. [61] represented the graph as a matrix with the entries representing the edges that connect graph nodes and applied matrix factorization for learning a low-dimensional graph representation. Nevertheless, decomposition-based models are not scalable due to the high computational cost and are not flexible enough to decompose a large-scale matrix [23]. Hence, decomposition-based methods are not practical to use in large networks for data min-

ing tasks.

Recently, deep learning-based models have taken place in network embedding to handle decomposition issues in large networks. Motivated by the word2vec [62] approach, Perozzi et al. [24] proposed DeepWalk, which learns network representations by feeding the generated random walk sequences to the skip-gram model. DeepWalk assumed nodes as "words" and random walk sequences as "sentences" and then maximized the co-occurrence probability of each node pair. Next, node2vec was proposed by Grover et al. [63], in which a biased random walk was performed on homogenous networks by employing depth-first and breadth-first sampling. In addition, with the LINE model, Tang et al. [28] employed both first- and second-order proximities among vertices by the edge-sampling algorithm for learning vectorial representations of nodes in large information networks. Furthermore, higher-order graph proximity for network representations was proposed in the GraRep model by Cao et al. [64]. Unfortunately, these state-of-the-art methods concentrated on learning node representations in a homogeneous network, but these methods cannot be directly used in heterogeneous networks.

Recently, heterogeneous information networks have gained popularity due to their powerful capability to model nodes from several aspects and their specific relations. HIN embedding attempts to embed the nodes from multiple notions into the shared low-dimensional space. HIN embedding methods have improved the ideas in homogenous graph embedding research to handle information graph heterogeneity. For example, metapath2vec [25], as an extension of DeepWalk, generates node sequences from metapath-based random walks to accommodate heterogeneous neighborhoods of a node and learns the representation of heterogeneous networks by applying the skip-gram model. HIN2vec [26] first discovers different relationship types between nodes by using a list of combined metapaths having shorter lengths than a specified value and employs them to generate better walk sequences. PTE [29] extends LINE by decomposing heterogeneous graphs into bipartite subgraphs and performs individual network embedding by using the LINE approach. Additionally, PTE is a text embedding technique. However, GE [30] and JLGE [32] extend the LINE approach for embedding location nodes. Both GE and JLGE decompose LBSNs into bipartite user-location subgraphs and then jointly perform representation learning for all node

pairs of subgraphs.

Last, deep neural network-based methods are imported to heterogeneous data embedding methods due to the power of deep models, such as autoencoders and graph neural networks, in modeling heterogeneous data. SHINE [34] borrows the autoencoder model to encode and decode the heterogeneous information in the social network to obtain the feature representation, and in [33, 48], the stacked denoising autoencoder (SDAE) was used as the deep learning model for feature representation. A graph neural network (GNN) is another model that learns graph representations using specifically designed neural layers and defines convolutions in the graph domain by aggregating the feature information of each node from the connected neighbors. GraphSAGE [65] is proposed as the seminal spatial-based GNN framework that is founded upon the general notion of aggregator functions for node embeddings. Zhang et al. [66] proposed a heterogeneous GNN model to consider heterogeneous graph structure and the heterogeneous contents together by collecting strongly correlated heterogeneous neighbors and aggregates feature information of the sampled neighbor nodes. Additionally, for better representation of heterogeneous graphs with rich node content features, Fu et al. [67] employed intrametapath aggregation for the content transformation of heterogeneous node attributes and intermetapath aggregation to generate node embeddings by applying the attention mechanism for every metapath.

Finally, we note that few research efforts can be found that employ subgraphs in the embedding process for recommending locations to the users. Our approach differs from previous works in the subgraph creation process. In these works, graphs are decomposed along with relationships or metapaths. However, we decompose the graphs according to the geographical attributes of the nodes such that location nodes around the user's current positions with the specified radius and user nodes (friend, expert, trusted) visiting these locations.

CHAPTER 3

TRUST-AWARE LOCATION RECOMMENDATION

In this chapter, we provide detailed explanations of our proposed location recommendation techniques. First, the main elements in the LBSN and the problem descriptions are stated in Section 3.3.1. Then, the models used in the proposed techniques, the LBSN model and the proposed trust model, are introduced in Section 3.1 and Section 3.2, respectively. After that, the *Trust-aware Location Recommendation with Random Walk: TLoRW* technique is described in detail in Section 3.3 and User Subgraph-based Graph Embedding technique *SgWalk* (***Sub**graph **Walk***) is described in detail in Section 3.4.

3.1 LBSN Model

We represent the particular LBSN with an unweighted and undirected graph model. This model, represented by $\mathcal{G}$, is a graph $\mathcal{G} < \mathcal{V}, \mathcal{E} >$, where $\mathcal{V}$ denotes the set of nodes v and $\mathcal{E}$ denotes the edges e between nodes. $\mathcal{V} = \mathcal{U} \cup \mathcal{L}$ where $\mathcal{U}$ is the set of users and $\mathcal{L}$ is the set of locations. $\mathcal{V}$ contains six different types of nodes:

- User node $u \in \mathcal{U}$, requesting recommendation.
- Friend nodes, denoting friends of the user u, such that each friend $f \in \mathcal{F}$.
- Expert nodes, denoting location experts in the user's vicinity, such that each expert $eu \in \mathcal{EU}$.
- Trusted user nodes, denoting trustworthy users in user u's vicinity, such that each trusted user, $tu \in \mathcal{TU}$.

- Location nodes, denoting locations visited in the user's vicinity either by user u or by a friend f, such that each location $l \in \mathcal{L}$.
- Popular location nodes, denoting popular locations in the user's vicinity, such that each popular location $pl \in \mathcal{PL}$.

The set of edges denoted by $\mathcal{E}$ is used for defining the links among the items in $\mathcal{V}$. $\mathcal{E}$ contains seven distinct edge types between the six nodes above, as shown in Figure 3.1.

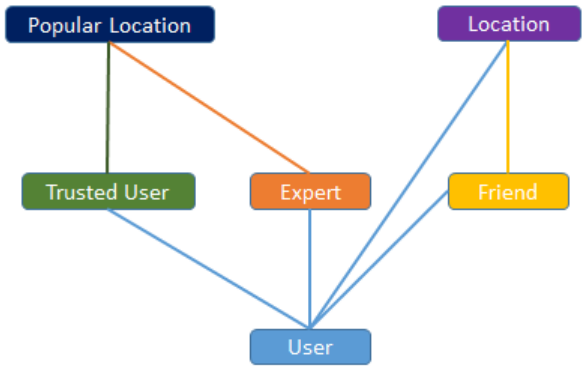


Figure 3.1: Vertex and edge types in the LBSN model.

3.2 Trust Model

Trust is widely recognized as an important component in human social relationships. In general, trust is a measure of confidence that an entity will behave in an expected manner, despite its inability to monitor or control the environment in which it operates [68]. Trust is a belief that does not necessarily presuppose past observed behavior, which is different from trustworthiness, which is a verified objective of trust through observed evidence [69]. Thus, trust includes both subjective and objective trust (i.e., trustworthiness). Trust is denoted in [70] simply as $\mathcal{T}(i, j, \alpha)$ and read as "i trusts j in a situation α."

In this thesis, we focus on the objective part of the trust and propose an approach to compute user trustworthiness scores. In our model, there is no available explicit trust relationship between users, and we employ the reliable behavior of trustee as stated in

the common definition of trust in the Section 1.2 in a global manner. To model a user's global trustworthiness score within an LBSN, we adopt the trust features described in [10]: *objectivity* and *consistency*. In our approach, the definition of these features and the way they are computed differ based on the LBSN environment and the data generated within the LBSN.

Definition 1 (*Objectivity*) *The objectivity of user u, denoted by O_u, is the normalized average of the objectivity scores of the visits by user u.*

The objectivity score of a visiting location l, O_l, depends on U_l, which is the *visiting status* of user u for location l, as given in Equation 3.1.

$$\mathcal{U}_l = \begin{cases} 1, & \{a | a \in \mathcal{A}, a.u = u \wedge a.l = l\} \neq \emptyset \\ 0, & \{a | a \in \mathcal{A}, a.u = u \wedge a.l = l\} = \emptyset \end{cases} \tag{3.1}$$

The *objectivity of a visit* for location l, $\mathcal{O}_l$, indicates that user u exhibits more objective behavior in visiting location l as $\mathcal{O}_l$ approaches 0. $\mathcal{O}_l$ is computed based on the location rating represented by $\mathcal{R}_l$ and the standard deviation, represented by σ_l as given in Equation 3.2. Note that $\mathcal{R}_l$ is the popularity score of location l calculated in 3.7.

$$\mathcal{O}_l = \mid \frac{\mathcal{U}_l - \mathcal{R}_l}{\sigma_l} \mid \tag{3.2}$$

The user's objectivity, represented by $\mathcal{O}_u$, is calculated as the mean of the objectivity scores of the user's visit, where $\mathcal{L}$ is the set of locations in the dataset (given in Equation 3.3). As the objectivity value approaches 0, the user is considered more objective.

$$\mathcal{O}_u = \frac{1}{\mid \mathcal{L} \mid} \sum_{l \in \mathcal{L}} \mathcal{O}_l \tag{3.3}$$

Definition 2 (*Consistency*) *A user is considered to be consistent if her or his check-in behavior conforms to the expected behavior in visiting locations. The consistency*

of a user, represented by C_u, can be defined as the deviations in the objectivity of her or his visits, as given in Equation 3.4.

$$\mathcal{C}_u = \mid \frac{\mathcal{O}_u - \mathcal{O}_m}{s_u} \mid \tag{3.4}$$

$\mathcal{C}_u$ is computed by using the average objectivity of all users, denoted by $\mathcal{O}_m$. If $\mathcal{C}_u$ is closer to 0, the user is considered to be more consistent. The consistency value is considered to provide the *trustworthiness* of the user. Finally, the trust score of a user, $\mathcal{T}_u$, is calculated by Equation 3.5.

$$\mathcal{T}_u = 1 - \Phi(\mathcal{C}_u) \tag{3.5}$$

In this equation, function $\Phi(x) = 1/(1 + exp(-x))$ is the sigmoid function, which limits $\mathcal{C}_u$ within the range [0, 1]. Given an LBSN, the global trust score of each user in the LBSN is calculated as shown in Algorithm 1.

Algorithm 1 Trustworthiness prediction algorithm

```
1: procedure PREDICTTRUST
2:     Initialize O, T
3:     users ← GetUsers ()
4:     for all u in users do
5:         O_u ← compute value according to Equation 3.3
6:     O_m ← FindAverage(O)
7:     for all u in users do
8:         T_u ← compute value according to Equation 3.5
```

3.3 Trust-aware Location Recommendation with Random Walk: TLoRW

We present our recommendation algorithm's details, *Trust-aware Location recommendation with Random Walk with restart (TLoRW)* in this section. As the main contribution of this algorithm, location recommendations for a given user with a specific spatial context are generated by considering the trusted users in the vicinity. The

random walk with restart approach is used for the creation of location recommendation lists. First, we explain the user subgraph algorithm. Following this, random walk execution details are given. Finally, a location recommendation with the proposed trust-aware location recommendation algorithm (TLoRW) is described in detail.

3.3.1 Preliminaries

In this section, definitions of the main elements in LBSNs and problem descriptions are given.

Definition 3 (*User*) *User is the essential human entity in a social network named user, member, etc. It is denoted by u and identified either by an anonymized id number or nickname in our datasets.*

Definition 4 (*Location*) *A location is a specific venue (such as a cafe or a cinema) that can be uniquely identified. A location has two attributes: a unique identifier (name) and geographical position. We use l to denote a location and l_c to represent its positional attribute as geographical coordinates in terms of longitude and latitude values.*

Definition 5 (*Check-in Activity*) *A check-in activity is in the form of a quadruple $a(u, l, l_c, \tau)$ such that for an activity a, location l is visited by user u on coordinates l_c at time τ.*

Definition 6 (*Vicinity*) *Vicinity is a circular region defined by the user's current location p_u and specifies the radius parameter ρ such that $|p_u - l_c| < \rho$. The obtained circular area is used as the recommendation region.*

Definition 7 (*Friend*) *Each user who takes part in an explicit direct relationship (friendship, following, etc.) in social networks is considered* friend*. A direct relationship between two users u_i and u_j is denoted as a tuple of (u_i, u_j)*

Definition 8 (*Trusted User*) *A user is considered trusted if the user's check-in activity behavior conforms to the most common behavior in visiting locations in a particular region.*

Trustworthiness score calculation of a user is performed according to the formulas stated in section 3.2.

Definition 9 (*Expert*) *An expert is a user who is supposed to have more knowledge about the locations in the given spatial context.*

Since LBSN data typically do not include expert information explicitly, in this work, we execute a HITS-based [71] algorithm to determine the experts in the vicinity. In the HITS-based method, people who visit the majority of the important locations in a region are regarded to have plenty of knowledge about the region. In the algorithm, users' expert scores are calculated by using Equation 3.6. The users are sorted in descending order according to the score, and the top-n% of the users are considered experts.

Definition 10 (*Popular Location*) *A location is considered popular if it is worth visiting in the given spatial context.*

As in expert information, LBSN datasets do not include any explicit information about the popularity of the locations. Therefore, we execute the same HITS-based [71] algorithm to find the popular locations in a vicinity. In the algorithm, if a venue is visited by many people, it should be considered an important location [72]. Location scores are calculated by using Equation 3.7. The locations are sorted in descending order according to the scores, and the top-n% of the locations are considered popular locations.

$$\mathcal{R}_{u_i} = \left\| \sum_{l_j \in \mathcal{A}_{u_i}} \mathcal{R}_{l_j} \right\| \tag{3.6}$$

$$\mathcal{R}_{l_j} = \left\| \sum_{u_i \in \mathcal{A}_{l_j}} \mathcal{R}_{u_i} \right\| \tag{3.7}$$

Expert user and popular location score calculation algorithms run simultaneously in an iterative manner, as depicted in equations 3.6 and 3.7. In the equations, $\mathcal{R}_{u_i}$ denotes the expert score of user u_i, $\mathcal{R}_{l_j}$ denotes the popularity score of location l_j and $\mathcal{A}_{u_i}, \mathcal{A}_{l_j}$ denotes the check-in activities of user u_i or location l_j. In each iteration, popularity scores are updated by using the previous iteration's expert scores, and then recently updated popularity scores are used to update the new expert scores within the same iteration. Popularity scores and expert scores are normalized at the end of each iteration. The algorithm stops once it converges.

Based on the definitions for the basic LBSN concepts, we define the problems that we focus on in this study as follows.

Problem 1 (Trustworthiness Score Prediction) *Given an LBSN history as a user check-in activity set $\mathcal{A}$ = {a_1,a_2,...,a_n}, a user u_i in $\mathcal{U}$, the task is to predict trust score $\mathcal{T}(i)$ of user u_i so that the user with highest score will conform to the most objective check-in behavior.*

Problem 2 (Location Recommendation) *Given a check-in activity set $\mathcal{A}$ = {a_1,a_2,...,a_n}, a target user u_q in $\mathcal{U}$ within the target region r_q, our objective is to generate a set of top-n locations { $l_1, ..., l_N$} as recommendations so that the recommendation set consisting of locations u_q will visit the next time with the highest accuracy.*

In this approach, we devise trust-aware location recommenders based on random walks, and our hypothesis is that incorporating trusted users improves recommendation accuracy. The main notations used in this thesis are given in Table 3.1.

3.3.2 User Subgraph Construction

We represent the particular subgraph with an unweighted and undirected graph model. This subgraph represented by $\mathcal{G}$ is a graph $\mathcal{G} < \mathcal{V}, \mathcal{E} >$, where $\mathcal{V}$ denotes the set of nodes v and $\mathcal{E}$ denotes the set of edges e. $\mathcal{V} \subset (\mathcal{U} \cup \mathcal{L})$ where $\mathcal{U}$ is the set of users and $\mathcal{L}$ is the set of locations. Note that $\mathcal{V}$ contains different subsets of $\mathcal{U}$ and $\mathcal{L}$. Moreover, it includes six different types of nodes: user, friend, expert, trusted user, location, and popular location.

Table 3.1: Definitions of Main Notations in TLoRW

Notation	Description
$\mathcal{U}$	Set of all users
$\mathcal{L}$	Set of all locations
$\mathcal{A}$	Set of check-in activities
$\mathcal{G}$	Graph of location based social networks
$\mathcal{V}$	Set of all vertices as union of users and locations
$\mathcal{E}$	Set of edges representing connections between $v \in \mathcal{V}$
$\mathcal{T}_{u_i}$	Trustworthiness score of a user i
$\mathcal{R}_{u_i}$	Expert score of a user i
$\mathcal{R}_{l_j}$	Popularity score of a location j
$\mathcal{A}_{u_i}$	Check-in activities of user u_i
$\mathcal{A}_{l_j}$	Check-in activities of location l_j
$\mathcal{F}$	Set of friends for a given user
$\mathcal{TU}$	Set of trusted users
$\mathcal{EU}$	Set of expert users
$\mathcal{PL}$	Set of popular locations
$\mathcal{U}_l$	Visited status of location by a user
$\mathcal{O}_l$	Objectivity of a visit for a location
$\mathcal{O}_u$	Objectivity of a user
$\mathcal{O}_m$	Mean objectivity of all users
$\mathcal{C}_u$	Consistency of a user

To recommend locations to a user at a specified position, TLoRW first constructs the subgraph using the following items:

- Locations in the check-in history of the user in the vicinity (personal and spatial contexts)
- Friends and their check-in history in the vicinity (social and spatial contexts)
- Trusted users and their check-in history for popular locations in the vicinity (trustworthiness and spatial contexts)

- Experts and their check-in history for popular locations in the vicinity (social and spatial contexts)

A user subgraph can be constructed incrementally according to the available data. The simplest user subgraph can be constructed by depending on only her or his own and friends' check-in history (Figure 3.2a). It can be further enriched by adding trusted users and popular locations visited by them (Figure 3.2b). Similarly, experts and their visited popular locations were added (Figure 3.2c). Finally, a complex user subgraph is constructed (Figure 3.2d) by using all introduced nodes, which are user, friend, location, expert, trusted user and popular locations. To analyze the effect of including additional types of nodes, we conduct accuracy performance analysis experiments by using each subgraph structure separately.

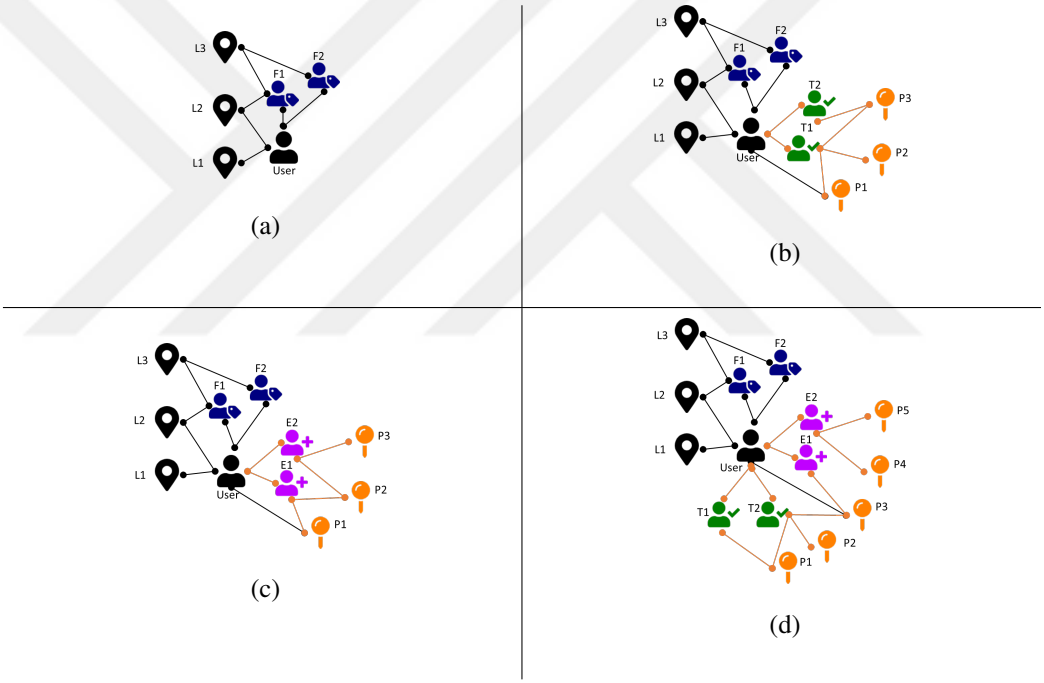


Figure 3.2: Incremental Construction of User Subgraph

The details of user subgraph construction adopted from [49] are given in Algorithm 2. In this algorithm, the target user for recommendation and the user's current position for vicinity are denoted by *usrId* and *crLocation*, respectively. In the algorithm, the *GetUserLocationsInVicinity* procedure obtains the check-in history of the current user in the vicinity. Similarly, the *GetFriendLocationsInVicinity* procedure retrieves the check-in activity of friends in the vicinity. Likewise, *GetExpertLocationsInVicin-*

ity and *GetTULocationsInVicinity* procedures find the check-ins of popular locations visited by experts and trusted users, respectively, in the recommendation region. Following this, the friends of the target user having a check-in history in the vicinity are retrieved in the *GetFriendsOfUser* method. In *GetExpertsInVicinity* and *GetTrustedUsersInVicinity* methods, top-n experts and trusted users having check-ins in the vicinity are fetched. Once all those users are collected, the relationships between the current user and these users (i.e., friends, experts, and trusted users) are included. Finally, the visited location nodes and the edges between users and these locations are added to the subgraph. A sample user subgraph is shown in Figure 3.3.

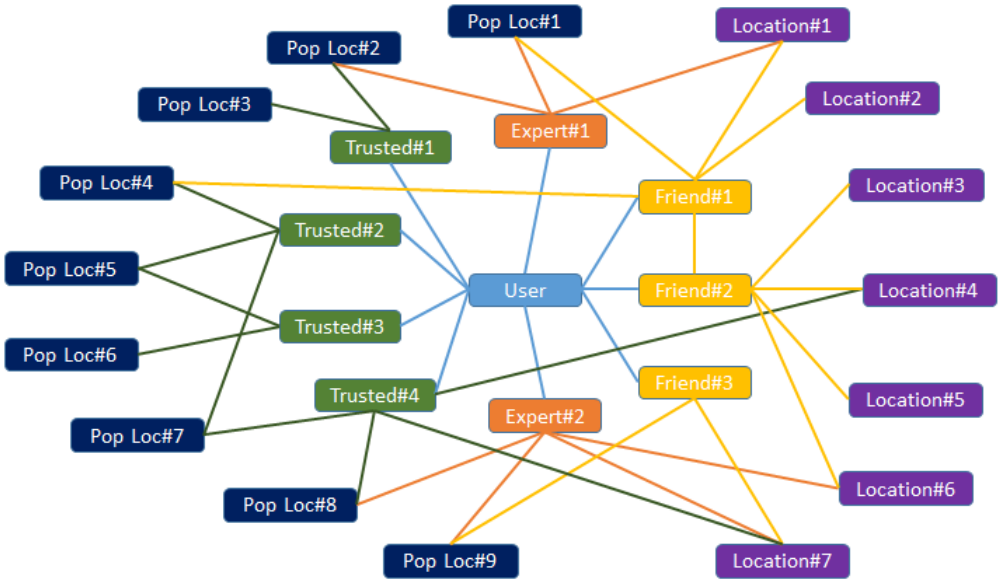


Figure 3.3: Sample user subgraph used for location recommendation.

In the algorithm, *tuCount*, *expCount* and *plCount* parameters restrict the number of trusted users, experts and popular locations in the user subgraph. Trusted users are obtained through the trust score calculation, as defined in Section 3.2. For experts and popular locations, the HITS-based [71] algorithm is adopted, in which user and location nodes correspond to authority and hub nodes, respectively. Score calculations of the user and location nodes for estimating the expert and popular location are performed by HITS-based algorithm iterations. User and location scores are ordered once iterating is complete, and the desired top-n experts and popular locations from the ordered lists are selected. A constant value (i.e., 40) is used as the iteration count during the work. Since we work on a subgraph constructed from the filtered LBSN

data, the count of locations and users in the vicinity is limited.

Algorithm 2 User subgraph construction algorithm

1: **procedure** GENERATESUBGRAPH($usrId, crLocation$)
2: tuCount ← max top-n percent of trustable user in the vicinity
3: expCount ← max top-n percent of experts in the vicinity
4: plCount ← max top-n percent of popular location in the vicinity
5: Initialize $G < V, E >$
6: $vicinity \leftarrow$ GetUserLocationsInVicinity ($usrId, crLocation$)
7: $vicinity \leftarrow vicinity \cup$ GetFriendLocationsInVicinity ($crLocation$)
8: $vicinity \leftarrow vicinity \cup$ GetExpertLocationsInVicinity ($crLocation, plCount$)
9: $vicinity \leftarrow vicinity \cup$ GetTULocationsInVicinity ($crLocation, plCount$)
10: $V \leftarrow currentuser$
11: $users \leftarrow$ GetFriendsOfUser ($usrId$)
12: $users \leftarrow users \cup$ GetExpertsInVicinity($crLocation, expCount$)
13: $users \leftarrow users \cup$ GetTrustedUsersInVicinity($crLocation, tuCount$)
14: **for all** $user$ in $users$ **do**
15: $E \leftarrow E \cup$ new edge connecting $currentuser$ and $user$
16: $V \leftarrow V \cup$ users
17: **for all** $data$ in $vicinity$ **do**
18: $V \leftarrow V \cup$ new location node for $data.location$
19: $E \leftarrow E \cup$ new edge connecting $data.user$ and $data.location$

3.3.3 Random Walk with TLoRW

Edges used to connect the nodes on a graph can be utilized to rank the nodes efficiently [16]. The random walk starts traversing the graph from a particular node, and the traversal continues through the links according to the edges' transition probabilities. The random walk traverses the graph until it reaches a stationary state. The visit counts per node are used to rank nodes of the graph during the graph traversal. Hence, the random walk's output is the vector of probabilities of each node in the stationary state. A random walk may go away from the initial node, which may result in context loss. As a solution to this issue, a random walk *with restart* is considered, where a

random probability that lets jump back to the starting node is available in every transition. As a result, the random walk traverses mostly the vicinity of the starting node without losing the location context.

The transition probability of the links can be presented in a matrix $Q = \alpha W + (1 - \alpha)R$ as specified in [73] to perform a random walk with restart. Here, W denotes the probability of transition by the links among nodes. R regulates the likelihood of resetting back to the initial node, and the α value adjusts the nature of the random walk.

The random walk is performed with a restart option, as described in Section 3.3.4, on the underlying graph of LBSN to recommend locations to users. The proposed random walk process initiates from the target user node and traverses over the location and user nodes over the graph. Whenever a location node is visited, 1 is added to the location's visit count. A constant probability of jumping back to the initial node is considered before shifting to the next node in each movement. When random walk iteration is completed, locations are sorted by visit counts, and the location recommendation results are provided. It is worth mentioning that the random walk is not done on the graph of the entire LBSN data. Instead, the proposed algorithm traverses over the subgraph of the user, which is constructed according to the spatial context of the user.

3.3.4 Location Recommendation with TLoRW

We can define the problem of trust-aware location recommendation as follows. Given a graph $\mathcal{G}$, which models the trust-aware LBSN, a user u in $\mathcal{U}$, and the current location of the user, we aim to generate a set of locations $\{ rl_1, ..., rl_j, ..., rl_k \}$ as recommendations, such that each rl_j is in $\mathcal{L}$ in the vicinity. The challenge in this problem is generating the recommendation list with the highest accuracy.

To fulfill this challenge, we propose the *Trust-aware Location Recommendation with Random Walk (TLoRW)* algorithm. TLoRW considers social, personal, trustworthiness, and positional (spatial) contexts. The algorithm is composed of two stages: construction of the subgraph and recommendation location. In the *subgraph construction*

stage, a user subgraph is formed for a particular user according to the user's spatial context, i.e., current location, and the random walk is performed on this subgraph as the second stage to generate recommendations.

Location recommendation list generation is performed according to Algorithm 3. The subgraph of a given user is constructed using Algorithm 2 and provided as an input to Algorithm 3 to perform the location recommendation by a random walk with restart. In the location recommendation algorithm, the *recmCount* parameter represents the desired recommendation count, *itCount* parameter denotes the random walk iteration bound and *rstProb* parameter represents the restart probability of jumping back. *crUser* and *crLocation* store the user and location nodes, respectively, currently being visited by the random walk algorithm. During the iteration, next user is selected randomly in *SelectNextUser* method from the users that visited *crLocation*. Similarly, next location is selected randomly from the locations visited by *crUser* in *SelectNextLocation* method and 1 is added to the node's visit count. The random walk stops after iterating up to the specified number, and location nodes are ordered by the algorithm according to the visit counts. Then, the top *recmCount* locations from the ordered list are selected, and the algorithm returns the recommendations to the user. The proposed LBSN model is an unweighted and undirected graph so that the movements can be done to both sides with equal probabilities in the random walk iterations. In other words, all of the neighbor location nodes of the user node have the same possibility of being the next *crLocation*. In addition, the user node decides whether to move to the initial location or its neighboring location nodes to facilitate the restart option according to the restart probability parameter

3.4 Location Recommendation with User Subgraph-based Graph Embedding: SgWalk

In this section, we introduce our method SgWalk, a heterogeneous graph embedding method using random walks on user subgraphs rather than using the whole social graph. Basically, we develop a graph embedding-based solution combining random walks with a skip-gram-like model, which has been widely adopted in the literature and provides high accuracy performance on different tasks [24, 63, 25, 26, 46]. Our

Algorithm 3 Location recommendation algorithm

```
recmCount ← desired recommendations count
itCount ← random walk iteration count
rstProb ← jumping back probability in each transition
G < V, E > ← subgraph of the user
crLocation ← null
while i < itCount do
    if rand(0, 1) < rstProb then
        crLocation ← null
    else
        crUser ← SelectNextUser(crLocation)
        crLocation ← SelectNextLocation(crUser)
        crLocation.visitCount ← crLocation.visitCount + 1
    i ← i + 1
sortedNodes ← SortNodesByVisitCount(G < V, E >)
result ← SelectFirstKNodes(sortedNodes, recmCount)
```

proposed method first constructs a user subgraph for user preferences, which helps to eliminate unintended users and uninterested locations, and then performs a random walk over a heterogeneous input graph to generate walk sequences that are fed into a skip-gram model for learning the node embedding. In the rest of the section, we first present our user subgraph construction strategy, followed by the node embedding learning process using skip-gram.

We can define the personalized location recommendation problem as follows. Given a graph G corresponding to a particular LBSN, a user $u \in \mathcal{U}$ and the current location of the user, we aim to generate a set of locations $\{r_{l_1}, ..., r_{l_i}, ..., r_{l_n}\}$ that have not been visited by u as recommendations, such that each $r_{l_i} \in \mathcal{L}$ is in the vicinity. The challenge is to populate this recommendation set with the highest accuracy. We follow four steps to fulfill this challenge: user subgraph construction, random walk generation, graph embedding, and location recommendation.

3.4.1 Preliminaries

Definition 11 (*User*) *User is the essential human entity in the social networks named user, member, etc. It is displayed by u and identified either by anonymized number or nickname in our datasets.*

Definition 12 (*Location*) *A location is a specific site (e.g., a cafe or a cinema) that can be uniquely identified. In our dataset, a location has two attributes: identifier (name) and geographical location. We use l to represent a location and l^c to denote its corresponding geographical attribute in terms of longitude and latitude coordinates.*

Definition 13 (*Check-in Activity*) *A check-in activity is made of a quadruple $a(u, l, l^c, \tau)$, which means in activity a, user u visits location l on coordinates l^c at time τ.*

Definition 14 (*Vicinity*) *Vicinity is a circular region defined by the user's current location p^u and the radius parameter ρ such that $| p^u - l^c | < \rho$. The obtained circular area is used as the recommendation region.*

Definition 15 (Node Embedding) *Given an LBSN as a heterogeneous information network (HIN), represented by $\mathcal{G}$, is a graph $\mathcal{G} < \mathcal{V}, \mathcal{E} >$, we aim to learn the d dimensional latent vector representations that preserve the structural and semantic relations among nodes, where $d \ll |V|$.*

Based on the definitions for the basic LBSN concepts, we define the problems that we focus on in this study as follows.

Problem 3 (Location Recommendation) *Given a check-in activity set $\mathcal{A}$ = {a_1, a_2, ..., a_n}, a querying user u_q in $\mathcal{U}$ within a vicinity, our goal is to find latent representations of v in $\mathcal{V}$ and generate a set of top-k locations { $l_1, ..., l_i, ..., l_k$} as recommendations so that the recommendation set consisting of previously not visited locations u_q will be visited next time with the highest accuracy.*

In this work, we devise a user subgraph-based location recommender, and we hypothesize that utilizing user subgraphs to generate random walk sequences improves the

Table 3.2: Definitions Of Main Notations in SgWalk

Notation	Explanation
$\mathcal{G}$	Graph of location based social network
$\mathcal{A}$	Set of check-in activities
$\mathcal{U}$	Set of all users
$\mathcal{L}$	Set of all locations
$\mathcal{V}$	Set of all vertices as union of users and locations
$\mathcal{E}$	Set of edges representing connections between $v \in \mathcal{V}$
$\mathcal{R}_{u_i}$	Expert score of a user i
$\mathcal{R}_{l_j}$	Popularity score of a location j
$\mathcal{A}_{u_i}$	Check-in activities of user u_i
$\mathcal{A}_{l_j}$	Check-in activities of location l_j
$\mathcal{W}$	Set of walk sequences
r	Number of walks per node
s	Length of walk sequence
w	Window size for the skip-gram process
d	Embedding vector dimension
$@k$	Recommendation count from top

recommendation accuracy. The main notations used in this paper are given in Table 3.2.

3.4.2 User Subgraph Construction

The user subgraph construction in SgWalk follows the same steps in the TLoRW technique as described in section 3.3.2.

3.4.3 Random Walk with SgWalk

In our technique, random walks are performed over the user subgraphs to sample from an input heterogeneous information network. Formally, for each user node u

in $\mathcal{U}$, we initiate a random walk sequence that starts from u and finishes when the walk length is reached. Likewise, in existing graph embedding approaches based on random walks [24, 25, 26, 46], we generate a set of node sequences with length s by performing a specified number of random walks r on the subgraphs of each node u in $\mathcal{U}$. Algorithm 4 illustrates our random walk process to generate a set of walks.

Algorithm 4 Random walk generation algorithm

```
1:  procedure GENERATERANDOMWALKS
2:      walkLength ← maximum number of nodes in a random walk sequence
3:      walkCount ← count of random walk sequences per user subgraph
4:      walks ← ∅
5:      for all user in 𝒰 do
6:          userLocations ← FindClusters(user)
7:          for all crLocation in userLocations do
8:              G < V, E > ← GenerateUserSubgraph(user, crLocation)
9:              nextLocation ← null
10:             while i < walkCount do
11:                 walkSequence ← ∅
12:                 while len(walkSequence) < walkLength do
13:                     nextUser ← SelectNextUser(nextLocation)
14:                     walkSequence.append(nextUser)
15:                     nextLocation ← SelectNextLocation(nextUser)
16:                     walkSequence.append(nextLocation)
17:                 walks.append(walkSequence)
18:                 i ← i + 1
19: return walks                ▷ returns the set of generated random walk sequences
```

After the subgraph of a given user is obtained according to Algorithm 2, the yielded user graph is provided to Algorithm 4 for the generation of the walk sequences. In this algorithm, *walkLength* and *walkCount* variables denote the maximum walk length and the numbers of sequences per subgraph, respectively. *FindClusters* method applies clustering algorithm DBSCAN [74] to the check-in activities $\mathcal{A}_u$ of user u with the radius parameter ρ. DBSCAN needs two parameters: the radius of the neighborhood and the minimum number of points in a neighborhood. Therefore, we prefer

this algorithm because DBSCAN does not need the number of cluster parameters and generates clusters according to neighborhood and density concepts. Random walks are generated for each cluster (subgraph) of the user. At each move, *nextLocation* and *nextUser* variables store the last visited location and the user nodes, respectively, by the random walk algorithm. In each iteration, *nextLocation* is updated, and the visited node id is appended to the walk sequence. When the random walk length is reached, the algorithm returns the generated random walk sets. Since our LBSN graph has unweighted and undirected edges from user to location, the movement probabilities are equal for all the nodes in all random walk iterations. Therefore, a uniformly sampled node from the neighbors of the current node is selected as the next node.

3.4.4 Node Embedding Learning with Skip-Gram

Since the frequency distribution of vertices in random walks of social networks and words in a language both follow a power law [75], in our method, we use a technique similar to word2vec [62] to generate the low-dimensional vector-space (d-dimensional) representation of a node in the graph. We adopt a skip-gram model with hierarchical softmax to generate the node embeddings. Skip-gram is a model that maximizes the co-occurrence probability among the nodes that appear within a window with size w in the set of walk sequences, and hierarchical softmax is utilized to speed up the training phase. Formally, for a pair of nodes, v_i and v_j, appearing in the window in the set of walks $\mathcal{W}$, the co-occurrence probability is defined as given in Equation 3.8.

$$Pr((v_i, v_j) \in \mathcal{W}) = \Phi(\vec{v_i} \cdot \vec{v_j}) \tag{3.8}$$

In equation $\Phi(\cdot)$, is the sigmoid function $\Phi(x) = 1/(1 + exp(-x))$, and $\vec{v_i}$ and $\vec{v_j}$ refer to the embeddings (vectors) of v_i and v_j, respectively. In addition, the skip-gram model employs negative edge sampling techniques to improve embedding accuracy by a randomly sampled edge between negative nodes $\tilde{v_k}$ and v_i that does not appear

in the set of walks. The negative edge probability is defined as given in Equation 3.9.

$$Pr((v_i, \tilde{v_k}) \notin \mathcal{W}) = 1 - Pr((v_i, \tilde{v_k}) \notin \mathcal{W}) = \Phi(-\vec{v_i} \cdot \vec{\tilde{v_k}}) \tag{3.9}$$

Here, negative samples (nodes) are uniformly drawn from node distributions in the walks. In summary, for the pair of nodes (v_i, v_j), the skip-gram model maximizes the objective function as given in Equation 5.

$$O = log(\Phi(\vec{v_i} \cdot \vec{v_j})) + \sum_{k=1,(v_i,\tilde{v_k}) \notin \mathcal{W}}^{N} log(\Phi(-\vec{\tilde{v_k}} \cdot \vec{v_i})) \tag{3.10}$$

In the equation, *N* is the number of negative edge samples. Parallel asynchronous stochastic gradient descent (ASGD) is utilized to learn the node embeddings (vector representations) efficiently by iterating over all node pairs appearing within a context window of size w in each walk sequence. The dimension size, d, of embedding vectors and the window size, w, (i.e., the context size) for model training using the skip-gram model are given in the experiments section (see Section 4.4.3).

3.4.5 Location Recommendation with SgWalk

Once the graph embedding of the nodes in an LBSN is performed, the vectorial representation of locations and users is learned in a shared space. For a given user $u_i \in \mathcal{U}$, we calculate the prediction score for each of the unvisited location $l_j \in \mathcal{L}$ in the vicinity according to Equation 3.11, and then we recommend the top-n of the ranked list of locations having the highest scores.

$$Prediction\_Score(u_i, l_j) = \vec{u^T}_i \cdot \vec{l_j} \tag{3.11}$$

In the equation, $\vec{u_i}$ and $\vec{l_j}$ refer to the embeddings (vectors) of u_i and l_j, respectively.

3.5 Complexity Analysis

In the proposed approach, 3 algorithms are considered for complexity analysis. First, in the trustworthiness prediction algorithm, objectivity score calculation is performed for the users $u \in \mathcal{U}$ and the locations $l \in \mathcal{L}$ per user, so its complexity is O($|\mathcal{L}| \times |\mathcal{U}|$). However, consistency is computed for the users $u \in \mathcal{U}$ over the obtained objectivity values that correspond to the complexity of O($|\mathcal{U}|$). Therefore, the overall algorithm complexity can be concluded as O($|\mathcal{L}| \times |\mathcal{U}|$) since O($|\mathcal{U}|$) is asymptotically smaller than O($|\mathcal{L}| \times |\mathcal{U}|$).

Second, in the user subgraph construction, we employ R-tree indices for the spatial database queries. The R-tree algorithm complexity for the average case is O(*log*n), and the worst case is O(N). Therefore, in the subgraph construction algorithm, locations in the vicinity can be filtered with a complexity of O($|\mathcal{L}|$) in the worst case. Similarly, experts and trusted users in the vicinity can be selected with a complexity of O($|\mathcal{U}|$). Therefore, we can calculate the complexity of vertex selection by O($|\mathcal{U}|$) + O($|\mathcal{L}|$), which is equal to O($|\mathcal{V}|$). The complexity of adding the edges among $u \in \mathcal{U}$ is C($|\mathcal{U}|$, 2), which is O($|\mathcal{U}|^2$), and between $l \in \mathcal{L}$ and $u \in \mathcal{U}$ is $|\mathcal{L}| \times |\mathcal{U}|$ in the worst case, where C is the combination function. Consequently, the overall complexity of subgraph construction can be determined as O($|\mathcal{V}|^2$), as O($|\mathcal{V}|$) + O($|\mathcal{U}|^2$) + O($|\mathcal{L}| \times |\mathcal{U}|$) is asymptotically smaller than O($|\mathcal{V}|^2$).

Last, in random walks, complexity relies on the iteration count. The required iteration counts are determined by the graph size, which depends on the edge count. In the worst case, a graph may have C($|\mathcal{V}|$, 2) edges at maximum, where C denotes the combination function. Let *m* be the expected movements for an edge on average. Since not all edges are evenly visited in movements, we specify *m* as the mean of movement counts for each edge to limit the iteration count.

Hence, random walk iteration complexity can be derived as $\mathrm{C}(|\mathcal{V}|, 2) \times m$, and it asymptotically equals O($|\mathcal{V}|^2$). In addition, vertices are sorted next until the random walk iterations are finished. The sorting complexity of vertices in a graph is O($|\mathcal{V}|$ log $|\mathcal{V}|$). As the sorting complexity is asymptotically smaller than random walk complexity, the overall complexity of random walk-based recommendation generation is

deduced as O($|\mathcal{V}|^2$).

It is crucial to note that the user subgraph construction and location recommendation algorithm is executed on the subgraph covering the current context of the user. Therefore, the $|\mathcal{V}|$ value for the user subgraph is expected to have considerably smaller values compared to the entire graph.

CHAPTER 4

EXPERIMENTAL ANALYSIS

In this chapter, we describe the datasets and methods used for the experiments and present the details of the experimental results.

4.1 Datasets

We employ 4 real-life datasets in our experiments. These are Brightkite [41], Foursquare [76], Gowalla [41] and Wee Places [77] datasets. All of these datasets contain user check-in data for the locations and friends of users. Check-in data contains anonymized user id and location id, longitude and latitude of the location and time of visit attributes in all datasets. In addition to these attributes, Foursquare dataset contains category of the location and Wee Places dataset contains city name and category of the location. We used the common attributes for evaluation in all datasets. Ten lines of check-in data from Brightkite dataset is given in Table 4.1

We use the subsets of the datasets that are filtered for New York City in the experiments. The statistics of our datasets are given in Table 4.2. As shown in Table 4.2, all these datasets have different average check-in counts per user. In contrast, the average numbers of friends per user are similar to each other. The numbers of friends per user in the Brightkite and Gowalla datasets are closer, but Foursquare has a slightly higher number than Brightkite and Gowalla. Furthermore, Wee Places has the highest number of friends per user among all datasets.

Table 4.1: Sample check-in data from Brightkite dataset

userid	**check-in date**	**latitude**	**longitude**	**locationid**
14736	2009-03-13T12:12:30Z	41.978563	-87.901799	9384ff46b84611dd8c13003048c10834
14738	2009-10-12T03:22:19Z	47.043218	-122.846168	ee43ba97c492be062eb23ff3d3840e70
14738	2009-10-12T02:41:51Z	46.422401	-122.891093	a18297ccc38c899b30ae6afd63518af2
14738	2009-06-06T06:08:53Z	47.299900	-122.254000	af9fe2e0aed211dda29d003048c10834
14738	2009-06-06T04:05:08Z	47.273074	-122.228913	d73df63a524611de8249003048c10834
14738	2009-05-01T02:44:02Z	47.466815	-122.342836	8bad6d9835f111dea2fd003048c10834
14738	2008-12-17T07:26:01Z	47.329026	-122.222428	91cabc60cc0311dd8600003048c10834
14738	2008-12-06T18:50:39Z	47.063599	-121.580536	6131134ec3be11dd8bff003048c10834
14738	2008-11-23T05:14:29Z	47.299900	-122.254000	af9fe2e0aed211dda29d003048c10834
14738	2008-11-13T07:52:19Z	47.302724	-122.225676	71bde1eca30211ddac69003048c10834

Table 4.2: Dataset Statistics

Dataset	Brightkite	Foursquare	Gowalla	Wee Places
No. of Users	6,144	11,154	10,142	4,812
No. of Locations	41,870	103,822	57,756	28,598
No. of Friendships	27,138	59,078	45,618	31,915
Check-ins per User	40.28	45.96	26.68	150.07
No. of Check-ins	247,464	512,645	270,571	722,119
Friends per User	4.42	5.30	4.50	6.63

4.2 Evaluation Metrics

We utilized three widely used metrics to evaluate the performance of SgWalk and baseline methods: precision@k, recall@k and f-measure@k [49, 26, 37, 23], where $@k$ notation denotes the recommendation count top k. Precison@k is calculated by checking whether locations in the ground truth are ranked in the recommendation list. It is calculated as given in Equation 4.1.

$$precision@k = \frac{\text{number of true locations in recommendation@k}}{\text{number of recommendations@k}} \tag{4.1}$$

Recall@k measures the ratio of truly recommended locations to all visited locations in the ground truth, which is calculated as given in Equation 4.2.

$$recall@k = \frac{\text{number of true locations in recommendation@k}}{\text{number of locations in ground truth}} \tag{4.2}$$

Precison@k and Recall@k are inversely affected in evaluations. In other words, when Precison@k is high, Recall@k is low or vice versa. To normalize these metrics, the f-measure is used, which is calculated as given in Equation 4.3.

$$f\text{-}Measure@k = \frac{2 \times Precision@k \times Recall@k}{Precision@k + Recall@k} \tag{4.3}$$

In the experiments, for all of the metrics, the performance is analyzed under k values of 3, 5, 8, 10, 15, 20.

4.3 Evaluation of TLoRW

In this section, we present the details of the experimental results of the TLoRW technique and describe the methods used for the experiments.

4.3.1 Evaluation methodology and parameter settings

Since the challenging problem and the proposed solution involve spatial context, we need to acquire locations for the test users. To simulate the current locations of the user and the vicinity of the current locations, we cluster the check-in data. As the clustering algorithm, we use DBSCAN [74]. We prefer this algorithm since it does not need the number of cluster parameters and generates the cluster according to density. Additionally, since DBSCAN is based on the concepts of neighborhood and density, it is possible to set the current locations for users with other locations in the vicinity. To fulfill this, DBSCAN needs two parameters: the radius of the neighborhood and the minimum number of points in a neighborhood. In the experiments, we set the minimum number of points to 3 and the neighborhood radius to 2,000 m, which is also consistent with the optimal neighborhood size in [78]. When TLoRW is used as

a live recommender system, the current location of a target user is directly used as a starting node. Therefore, it does not need DBSCAN in real use.

After the check-in data of a user are clustered by the DBSCAN algorithm, the center of each cluster is considered the current location of the user for the region of recommendation. The clustered user data are sorted by check-in date from old to recent. The sorted check-in data for each cluster are partitioned into training and test datasets such that the old data constitute the training dataset since we aim to generate location recommendations to visit next.

To set the restart probability (α value), during validation experiments, we observed that the α parameter had different best values according to the datasets. However, using high values of α for different datasets may result in model overfitting. Hence, we set a unique α value of 0.05 for all the datasets[49].

The experiments are conducted under 5-fold cross validation (such that the number of check-ins in the training dataset is four times the number of check-ins in a test dataset). We set the random walk iteration count to 1,000.

4.3.2 Investigation on Effect of Friendships

In this section, we investigate the impact of friendship on recommendation accuracy. For this study, we first analyze the number of friendship connections per user. As stated in [75], the frequency distribution of words in a language fits to a power law, and we can see the same pattern in the distribution of user count according to the number of friends. Distributions for the Brightkite, Foursquare, Gowalla, and Wee Places datasets are given in Figure 4.1. We used these obtained distributions to create user groups for testing.

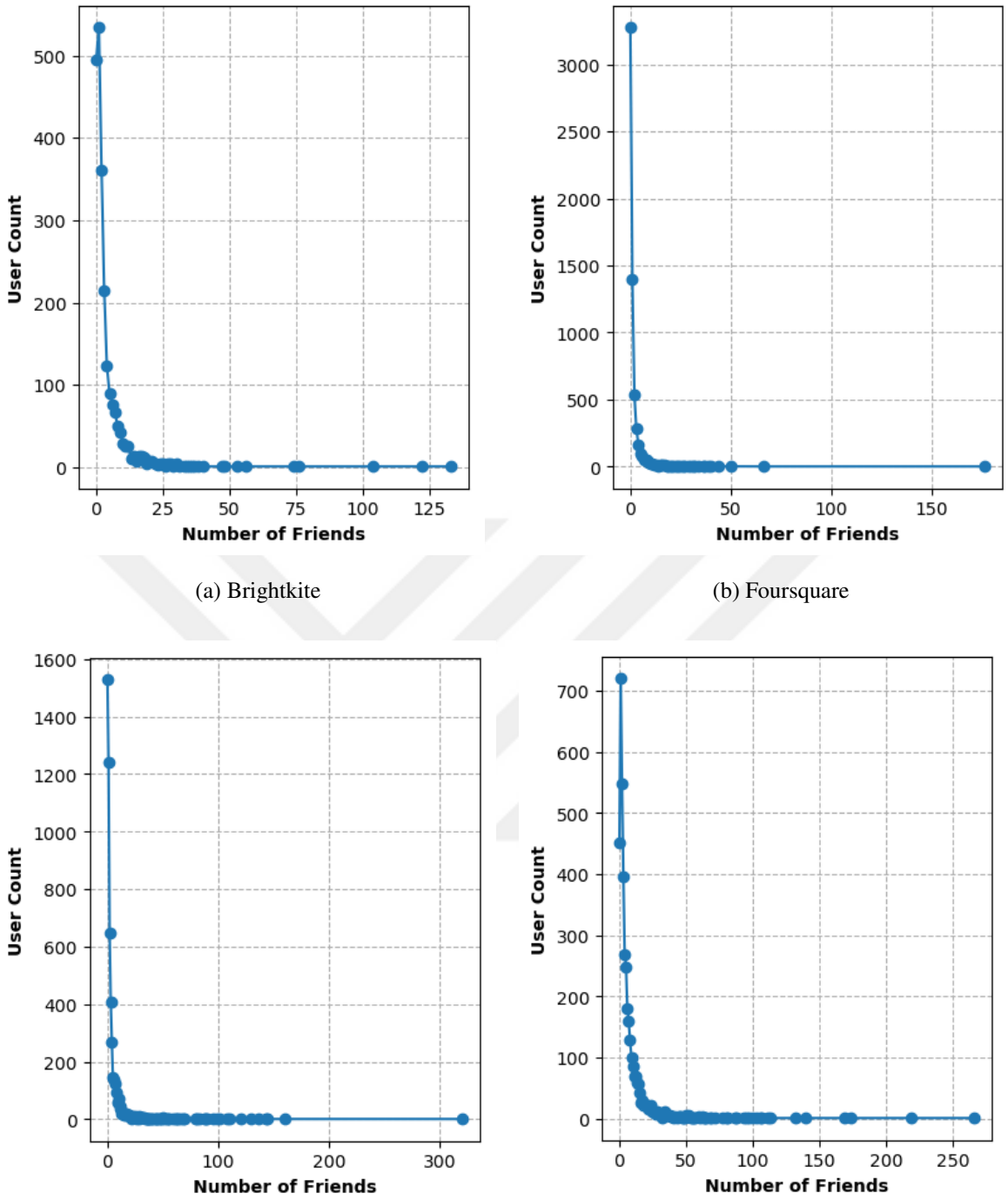
The size of user groups differs in literature, and there is no common methodology or systematic to do it. For example, while Tang et al. [79], investigating the influence of the friends on the output, users having 2, 5, and 10 friends were compared with each other. In [80], 'Friend Groups' were studied and in the evaluation, groups sizes are considered as follows, G1: [1-1], G2: [2-5], G3: [6-10], G4: [11-15], G5:[16-20]. Therefore, groups are constructed according to the best fit for their circumstances.

In our case, we attempted to keep the user counts in each group closer to each other for the same dataset to be able to compare them. Groups should also be compatible across datasets. To satisfy these constraints, we slightly modified the setup in [80] to fit power law and created six user groups as follows: (i) users having no friend at all ([0–0]), (ii) users having only 1 friend ([1–1]), (iii) users having 2 or 3 friends ([2–3]), (iv) users having 4, 5 or 6 friends ([4–6]), (v) users having 7, 8, 9 or 10 friends ([7–10]), and (vi) users having 11 or more friends ([11–350]). User counts per created group are given in Table 4.3. As shown in Table 4.3, while most people are in the first group for Foursquare and Gowalla with 3,276 and 1,528, respectively, in the third group for Brightkite and Wee Places 575 and 942, respectively. The fewer users are moved to the end of the table as the fifth group for the Brightkite and Gowalla with 188 and 344, respectively, the sixth group for Foursquare with 150, and the first group for Wee Places with 452.

Table 4.3: User counts for having a number of friends

	[0-0]	[1-1]	[2-3]	[4-6]	[7-10]	[11-350]
Brightkite	495	534	575	288	188	196
Foursquare	3,276	1,397	816	334	159	150
Gowalla	1,528	1,242	1,053	554	344	375
Wee Places	452	720	942	697	473	699

We performed the tests using these six groups independently for different recommendation counts in all datasets according to the setting explained in the evaluation methodology. Evaluation results are displayed in terms of the f-measure metric per group and per recommendation count in Figure 4.2. In accordance with the number of friendships, performance values are minimal when the users have no friends. However, performance values are noticeably increased by the increasing number of friends in all recommendation counts for all datasets. According to Figure 4.2, an increase in the friends of the user has a more significant effect for the Gowalla and Wee Places dataset compared to a monotonic increase in the Brightkite and Foursquare datasets. When we check the impact on the recommendation count, fewer recommendations (R3, R5) have remarkable improvements with respect to the higher number of recommendations (R15, R20)



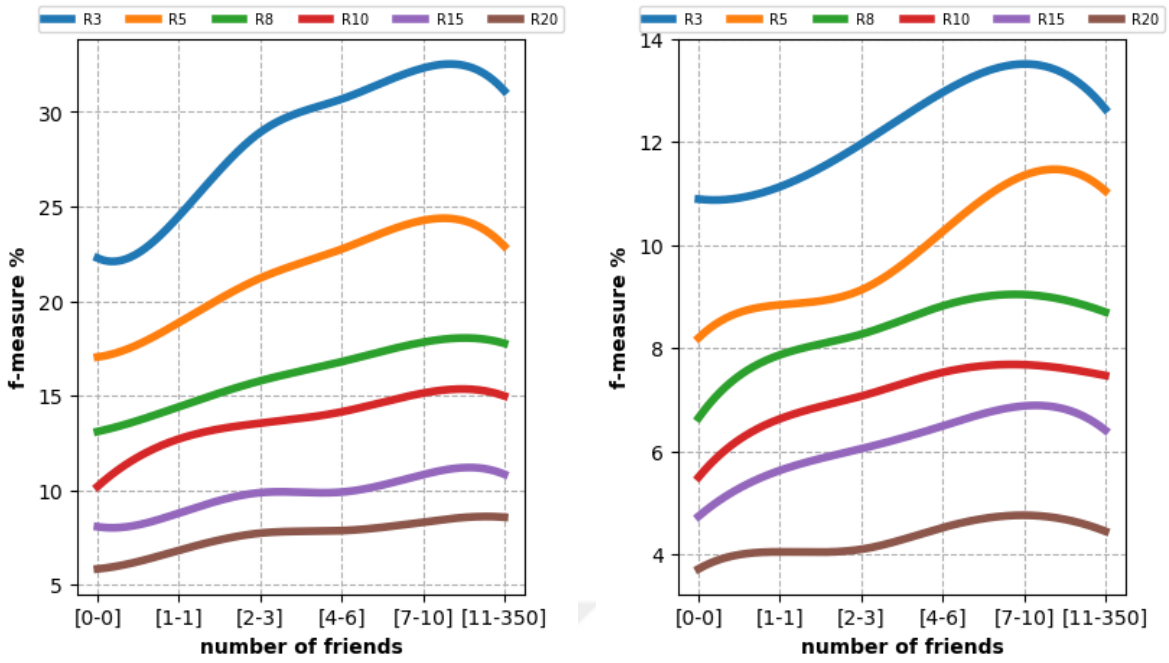
(a) Brightkite

(b) Foursquare

(c) Gowalla

(d) Wee Places

Figure 4.1: Distribution of user count according to number of friends *Brightkite, Foursquare, Gowalla and Wee Places* datasets.



(a) Brightkite

(b) Foursquare

(c) Gowalla

(d) Wee Places

Figure 4.2: F-measure values for varying numbers of friends with *Brightkite, Foursquare, Gowalla and Wee Places* datasets.

In all datasets, performance value increases with the increase in the number of friends user nodes and gains the maximum values at a certain point; after that, users having 11 and more friends suffer losing personal context and decrease in performance except for Wee Places dataset. The same pattern for performance change can be observed on different recommendation counts R3, R5, R8, R10, R15 and R20. According to the results, maximum performance is achieved with [4-6] friends in Gowalla dataset, [7–10] friends in Brightkite and Foursquare datasets, and [11-350] friends in Wee Places dataset.

4.3.3 Analysis on the Effect of Information Fusion

As the first set of experiments, we compared the user subgraphs covering different types of nodes to assess the impact of information fusion:

- ***Friend*** is the simplest subgraph that is constructed with three types of nodes: user, friend and location nodes and the edges between them. It is displayed as $\mathcal{G} < \mathcal{V}, \mathcal{E} >_{friend}$ where $\mathcal{V}_{friend} \subset (\mathcal{U} \cup \mathcal{F} \cup \mathcal{L})$ and $\mathcal{E}_{friend} = \{e | e \in \mathcal{E}, e.start \in \mathcal{V}_{friend} \wedge e.end \in \mathcal{V}_{friend}\}$.
  The accuracy results of $\mathcal{G} < \mathcal{V}, \mathcal{E} >_{friend}$ are used as a base for performance comparison.

- ***Expert*** is the derived subgraph that is constructed by including two additional latent nodes, which are experts and popular location nodes, to $\mathcal{G} < \mathcal{V}, \mathcal{E} >_{friend}$. It is displayed as $\mathcal{G} < \mathcal{V}, \mathcal{E} >_{expert}$ where $\mathcal{V}_{expert} \subset (\mathcal{V}_{friend} \cup \mathcal{EU} \cup \mathcal{PL})$ and $\mathcal{E}_{expert} = \{e | e \in \mathcal{E}, e.start \in \mathcal{V}_{expert} \wedge e.end \in \mathcal{V}_{expert}\}$.
  Subgraph $\mathcal{G} < \mathcal{V}, \mathcal{E} >_{expert}$ is designed to assess the effect of adding expert nodes.

- ***Trust*** is the derived subgraph that is constructed by including two additional latent nodes, which are trusted user and popular location nodes, to $\mathcal{G} < \mathcal{V}, \mathcal{E} >_{friend}$. It is displayed as $\mathcal{G} < \mathcal{V}, \mathcal{E} >_{trust}$ where $\mathcal{V}_{trust} \subset (\mathcal{V}_{friend} \cup \mathcal{TU} \cup \mathcal{PL})$ and $\mathcal{E}_{trust} = \{e | e \in \mathcal{E}, e.start \in \mathcal{V}_{trust} \wedge e.end \in \mathcal{V}_{trust}\}$.
  Subgraph $\mathcal{G} < \mathcal{V}, \mathcal{E} >_{trust}$ is designed to assess the effect of adding trusted user nodes.

- ***TLoRW*** is the proposed subgraph that is constructed by including three additional latent nodes, which are expert, trusted user and popular location nodes, to the $\mathcal{G} < \mathcal{V}, \mathcal{E} >_{friend}$. It is displayed as $\mathcal{G} < \mathcal{V}, \mathcal{E} >_{TLoRW}$ where
  $\mathcal{V}_{TLoRW} \subset (\mathcal{V}_{friend} \cup \mathcal{TU} \cup \mathcal{EU} \cup \mathcal{PL})$ and
  $\mathcal{E}_{TLoRW} = \{e | e \in \mathcal{E}, e.start \in \mathcal{V}_{TLoRW} \wedge e.end \in \mathcal{V}_{TLoRW}\}$.
  We aim to evaluate the effect of fusing two types of latent nodes, expert and trusted user, with subgraph $\mathcal{G} < \mathcal{V}, \mathcal{E} >_{TLoRW}$.

In particular, we aim to determine which subgraph performs best to be able to continue our evaluation with state-of-the-art techniques. The results of the subgraph evaluation experiments conducted on the Brightkite, Foursquare, Gowalla and Wee Places datasets are given in Table 4.4. The results clearly indicate that the types of nodes affect the recommendation performance in terms of precision, recall and f-measure metrics.

Friendship relations exist on all datasets, so we set $\mathcal{G} < \mathcal{V}, \mathcal{E} >_{friend}$ as the baseline subgraph. Then, we investigate the impacts of 2 latent node types: expert and trusted users. As expected, a subgraph having only friend nodes and its check-ins ($\mathcal{G} < \mathcal{V}, \mathcal{E} >_{friend}$) has the lowest recommendation accuracy in the experiments among all subgraphs. It can be concluded that the friends of a user cannot cover a sufficient number of locations for the user's current context. Then, we experimented with the performance of subgraphs with latent nodes. Experimental results of subgraphs including friend and expert nodes ($\mathcal{G} < \mathcal{V}, \mathcal{E} >_{expert}$) and subgraphs including friend and trusted user nodes ($\mathcal{G} < \mathcal{V}, \mathcal{E} >_{trust}$) indicate that both subgraphs acquire better metric values than subgraph $\mathcal{G} < \mathcal{V}, \mathcal{E} >_{friend}$. In addition, subgraph ($\mathcal{G} < \mathcal{V}, \mathcal{E} >_{expert}$) performs better than subgraph $\mathcal{G} < \mathcal{V}, \mathcal{E} >_{trust}$ in all datasets. It is reasonable since if a location is visited by an expert, then it has a higher possibility of being visited by other users. Trusted users may also recommend essential places, so their contribution is not as high as that of experts. As the last option, we consider the subgraph containing all node types together (subgraph $\mathcal{G} < \mathcal{V}, \mathcal{E} >_{TLoRW}$), and then we obtain the best performance in all datasets. Experts and trusted users have different check-in behaviors, and including these latent nodes improves the performance in terms of popularity and trustworthiness. These results also indicate that combining disjoint types (friend, expert, trusted user) improves the accuracy of the

recommended locations.

In all four datasets, we observe a similar performance pattern. It is an important observation revealing that the performance of subgraph $\mathcal{G} < \mathcal{V}, \mathcal{E} >_{TLoRW}$ is not dataset-specific. Although all datasets' statistics differ in terms of check-ins per user and friends per user values, subgraph $\mathcal{G} < \mathcal{V}, \mathcal{E} >_{TLoRW}$ attains the best performance in all of the test cases. Furthermore, the experimental results are more distinctive for a small number of recommendations, but all subgraphs yield similar results by the increase in the number of recommendations.

4.3.4 Parameter Tuning Experiments

In this set of experiments, we further elaborate on the effect of the number of experts and trusted users on the accuracy of the proposed TLoRW algorithm. The configuration parameter values of this experiment are the same as those of the information fusion experiments except for the parameter for the inclusion of experts and trusted users. In this analysis, to facilitate the comparison, we use recall@5 as the metric, since it gives the most distinctive values in Table 4.6.

In this experiment, expert and trusted user inclusion is increased from 0 to 10, and the results of each configuration are acquired separately. The results of this experiment are depicted in Figure 4.3. For better visualization of performance changes, a *heat map* graph is preferred, and for the intermediate values, experimental results are interpolated.

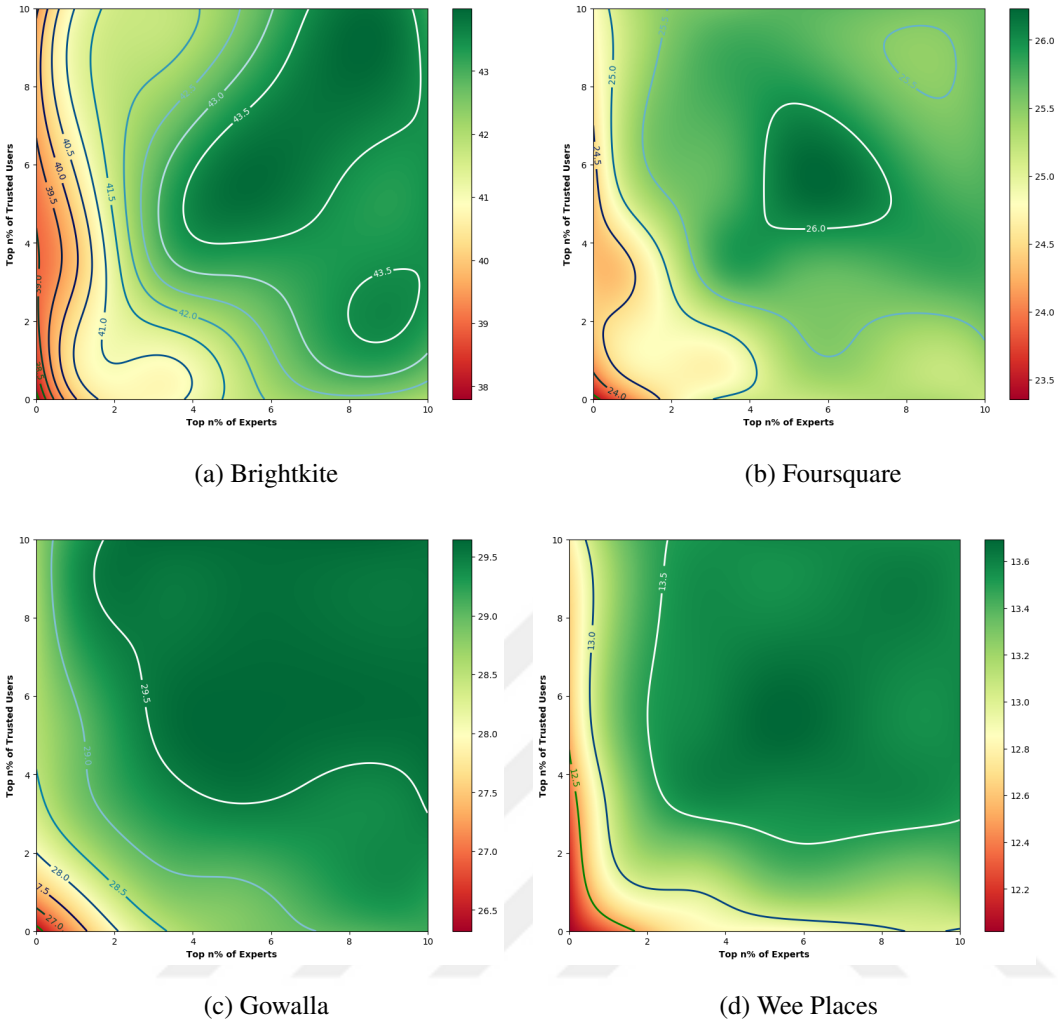
In accordance with information fusion, performance values are minimal when there are no expert and trusted user nodes added to the user subgraph. However, the performance values increase noticeably immediately after latent node inclusion in all datasets. According to Figure 4.3(a) and 4.3(b), an increase in the expert node percent has a greater effect than a trusted user in the Brightkite and Foursquare datasets. However, for the Gowalla and Wee Places datasets, given in Figure 4.3(c) and 4.3(d), respectively, adding latent nodes has an almost symmetric effect.

In all datasets, the performance value increases with the increase in *n* for both expert and trusted user nodes and gains the maximum values at a certain point, after which

Table 4.4: Recommendation performance under varying k values for the effect of information fusion

		Friend			Expert			Trust			TLoRW		
	@k	precision	recall	f-measure	precision	recall	f-measure	precision	recall	f-measure	precision	recall	f-measure
Brightkite	3	**22.52**	30.08	**25.76**	**26.11**	34.88	**29.87**	**23.67**	31.62	**27.07**	**29.15**	38.94	**33.34**
	5	18.51	37.79	24.85	20.73	42.33	27.83	19.40	39.61	26.04	21.49	43.87	28.85
	8	14.60	45.38	22.09	15.99	49.70	24.19	15.44	48.01	23.37	16.74	49.92	25.07
	10	12.92	49.41	20.49	13.69	52.34	21.70	13.30	50.85	21.09	14.48	52.53	22.70
	15	9.96	56.04	16.92	10.22	57.50	17.36	10.06	56.58	17.08	10.95	58.96	18.47
	20	8.05	**59.78**	14.19	8.08	**59.98**	14.24	8.07	**59.94**	14.23	8.17	**60.66**	14.40
Gowalla	3	**16.20**	22.29	**18.76**	**17.04**	23.44	**19.73**	**16.83**	23.16	**19.49**	**17.40**	23.94	**20.15**
	5	12.45	26.32	16.90	13.79	29.16	18.73	13.58	28.70	18.43	14.04	29.68	19.06
	8	9.26	29.75	14.12	10.76	34.57	16.41	10.86	34.91	16.57	11.06	35.54	16.87
	10	7.91	31.20	12.62	9.41	37.11	15.01	9.62	37.93	15.34	9.64	38.04	15.39
	15	5.89	34.01	10.04	7.11	41.04	12.12	7.50	43.29	12.78	7.78	44.04	13.23
	20	4.70	**35.77**	8.31	5.71	**43.41**	10.09	6.11	**46.49**	10.80	6.36	**47.56**	11.21
Foursquare	3	**13.91**	18.75	**15.97**	**14.67**	19.77	**16.84**	**14.41**	19.42	**16.54**	**16.54**	22.29	**18.99**
	5	11.49	23.36	15.41	12.41	25.23	16.64	12.19	24.76	16.33	12.85	26.12	17.23
	8	9.67	29.66	14.59	10.20	31.27	15.38	10.04	30.77	15.13	10.39	31.33	15.61
	10	8.76	33.00	13.85	9.12	34.34	14.41	8.97	33.80	14.18	9.59	34.85	15.04
	15	7.19	37.82	12.08	7.65	39.65	12.83	7.59	39.27	12.71	7.98	40.91	13.36
	20	5.99	**41.04**	10.46	6.82	**42.79**	11.77	6.82	**42.78**	11.77	7.21	**44.26**	12.40
Wee Places	3	**7.94**	9.49	**8.64**	**8.34**	11.15	**9.54**	**8.07**	10.34	**9.06**	**8.62**	11.49	**9.85**
	5	7.09	10.89	8.59	7.87	12.09	9.53	7.38	11.58	9.02	8.05	12.68	9.84
	8	6.51	13.09	8.69	6.71	13.50	8.96	6.61	13.28	8.82	7.12	14.33	9.52
	10	6.06	14.52	8.55	6.21	14.91	8.77	6.13	14.90	8.68	6.50	15.09	9.09
	15	5.10	15.90	7.73	5.46	16.66	8.23	5.23	15.93	7.87	5.66	17.59	8.57
	20	4.37	**18.88**	7.09	4.87	**19.10**	7.76	4.70	**18.42**	7.49	5.08	**19.92**	8.09

it changes little if n is increased further. To better represent the locality and trustworthiness, we aim to keep the n value as small as possible; therefore, we use the top 5% of the experts and the top 5% of trusted users as the maximum number of experts and trusted users, respectively, in the following experiments.



(a) Brightkite

(b) Foursquare

(c) Gowalla

(d) Wee Places

Figure 4.3: Performance comparison of top n% expert vs. trusted user under recall@5 for *Brightkite, Foursquare, Gowalla and Wee Places* datasets.

4.3.5 Comparative Accuracy Performance Analysis

We analyze the accuracy performance of TLoRW under a varying number of recommendations against the state-of-the-art methods from the literature that were developed as trust-based recommendation techniques. The details of the compared methods are listed as follows:

- **CLR [50]** clusters similar users and locations by applying pattern similarities between different objects to obtain location recommendations. In this approach, the probability of a particular user visiting a particular location is calculated by

using the user similarity scores. We calculate the similarity of a pair of users and locations in terms of location or user vectors, respectively, by employing cosine similarity.

- **CDL [27]** is deep representation learning for content information and collaborative filtering for the rating values, considering two-way interactions between the two. In this work, the stacked denoising autoencoder (SDAE) is used as the deep learning model. We provide the $\mathcal{U} \times \mathcal{L}$ matrix as training data and *geographical coordinates* of locations as feature data to obtain the location recommendations from the CDL approach.

- **TECF [18]** calculates trust-enhanced user similarity by performing network embedding on the user-user network constructed based on covisiting behavior and then fuses trust and geographic and temporal contexts to generate location recommendations. In the experiments, we set the parameters as δ = 0.2, θ = 0.1 and d = 100.

- **TrustMF [52]** models users with trustor and trustee roles and predicts ratings with these two separate models, and then the results of these two models are incorporated to generate the final predictions. We set the parameters as $\lambda_U = \lambda_V = 0.001$ and λ_T = 1 in the evaluation.

- **TrustSVD [14]** alleviates trust-based matrix factorization by considering explicit and implicit information of user trust and item ratings in the recommendation process. We use parameters λ_t = 1 and λ = 0.5 in the evaluation.

The results of the experiments performed on the Brightkite, Foursquare, Gowalla and Wee Places datasets are given in Figures 4.4, 4.5, 4.6, and 4.7, respectively. According to the results, the performance values of the proposed method are very competitive with several state-of-the-art methods in terms of precision, recall, and f-measure metrics along with all datasets; hence, TLoRW outperforms all comparisons (in terms of the f-measure). TrustSVD has the second-best performance value in the evaluation. However, its performance is worse than the CLR and TECF approaches on the Brightkite dataset. The low performance of these techniques may be caused by the Brightkite dataset having the lowest social data statistics (*friends per user*) among

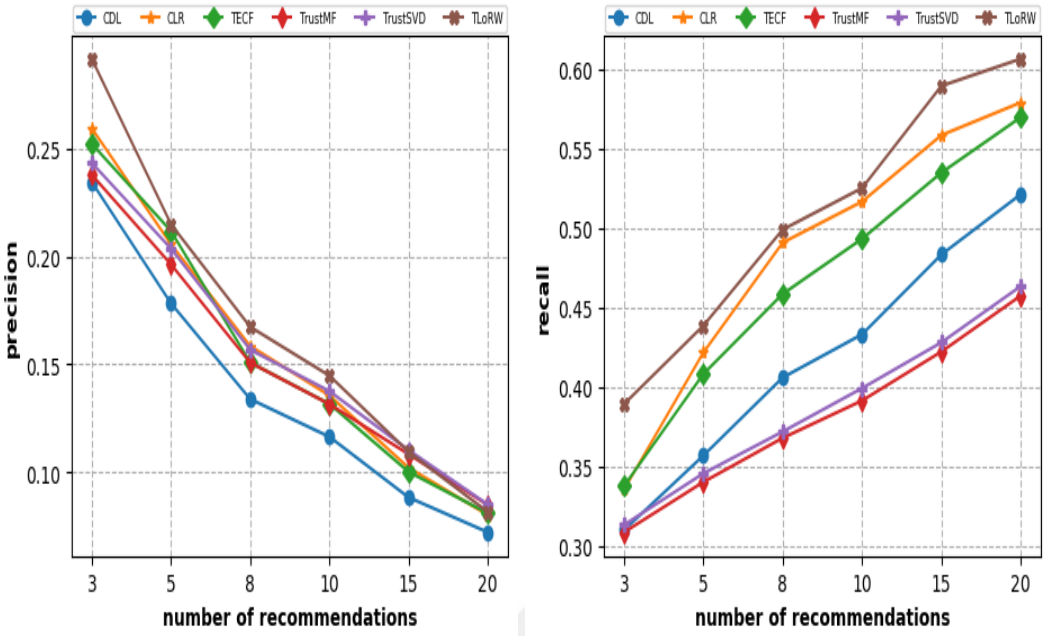
all datasets. TrustMF, the other matrix factorization technique, behaves similarly to TrustSVD. Its performance values are close to TrustSVD, and it performs worse on the Brightkite dataset. TECF has moderate performance on average, but it has the second-best result on the Brightkite dataset. In TECF, trust calculation is performed on the user network embedding, and it does not rely on explicit social network data. For this reason, it has higher performance on the dataset with the lowest social network statistics than the other datasets. The CLR technique cannot perform well since it does not consider the social connections between users and only focuses on user-user and location-location similarities. CLR also achieves its best performance on the Brightkite dataset. In contrast, CDL has the lowest performance among all. CDL heavily depends on item features to perform well. However, we can only feed the longitude and latitudes of the locations as a future in the datasets used for evaluation. For this reason, it may not reach its expected performance during the experiments.

As a summary, accuracy improvement with respect to the compared methods (in terms of f-measure under @5 recommendations) across the datasets are given in Table 4.5. It is observed that TLoRW provides the maximum improvement against CDL with 15% on average. The minimum improvement is over TrustSVD with 5% on average. These results show a strong indication that TLoRW, focusing on spatial context, social context (expert and trusted users as well as friendship) and user preferences, can generate more accurate recommendations.

Table 4.5: TLoRW Improvement Percentages in f-Measure @5

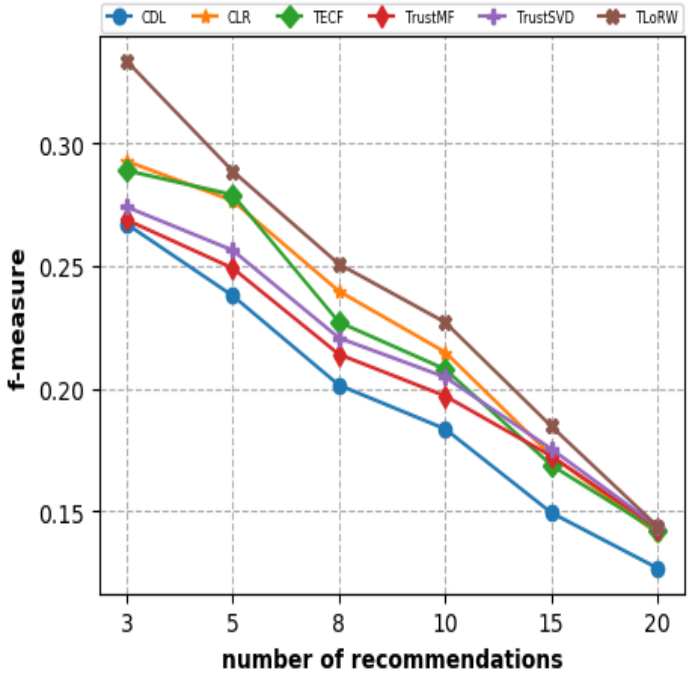
	CDL	TECF	TrustMF	TrustSVD	CLR
Brightkite	21%	3%	16%	13%	4%
Four Square	11%	8%	10%	3%	7%
Gowalla	19%	16%	5%	2%	17%
Wee Places	8%	6%	7%	3%	8%
Average	**15%**	8%	9%	**5%**	9%

The performance of the techniques varies depending on the characteristics of the dataset used (Table 4.2). However, on all four datasets, TLoRW attains the best performance values in all test cases. Hence, we can conclude that the performance of



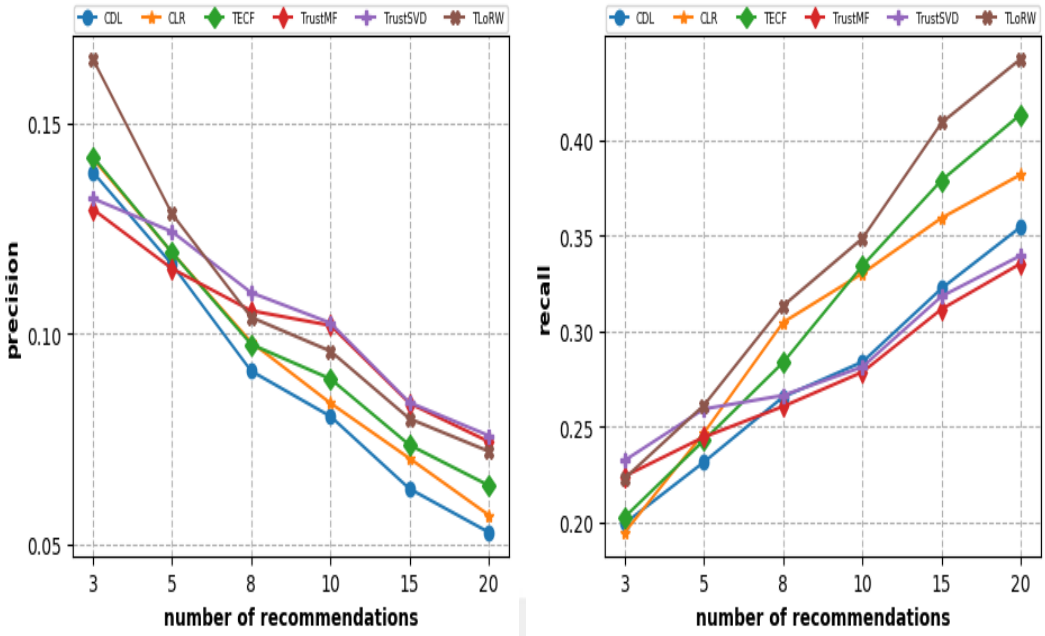
(a) Precision

(b) Recall



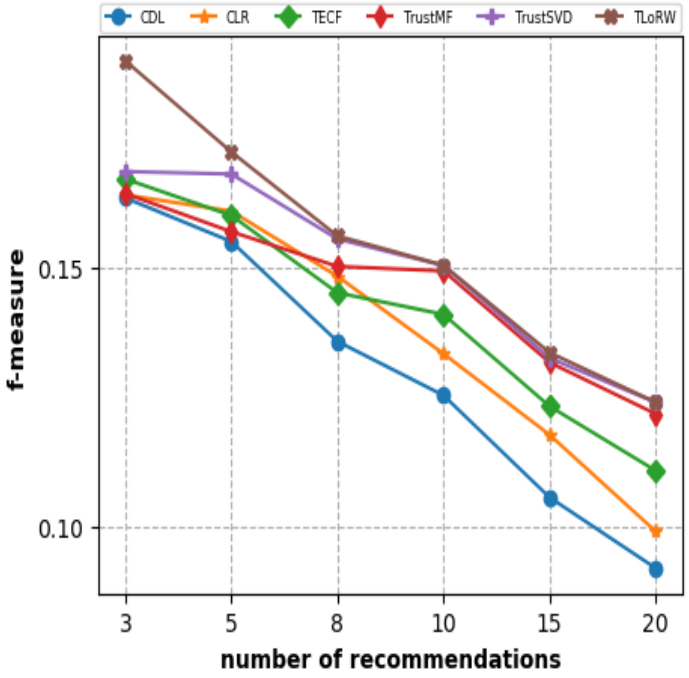
(c) F-measure

Figure 4.4: Precision, recall and f-measure values of the comparison algorithms for varying numbers of recommendations with the *Brightkite* dataset.



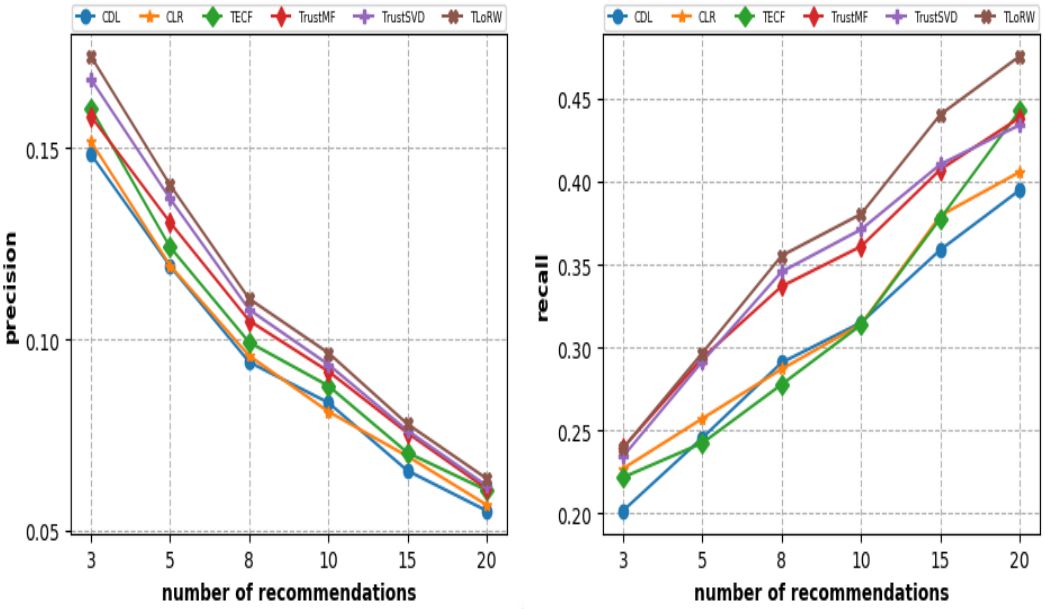
(a) Precision

(b) Recall



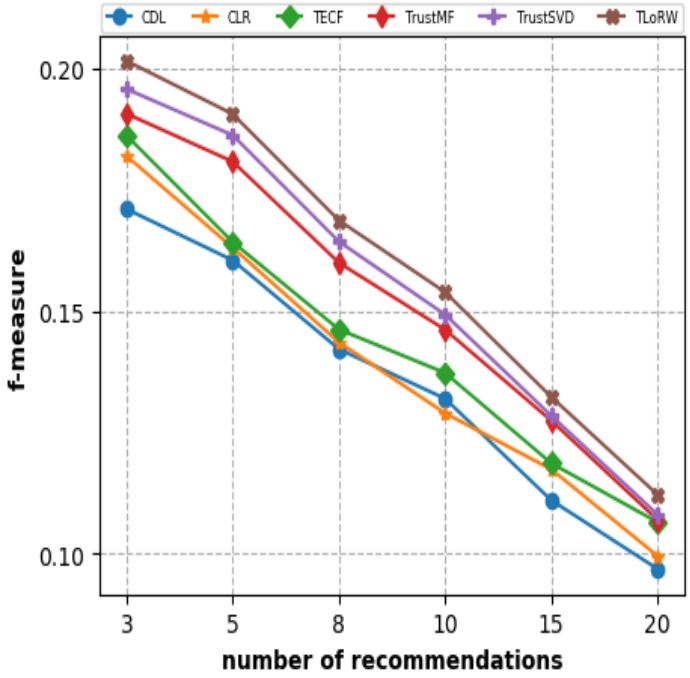
(c) F-measure

Figure 4.5: Precision, recall and f-measure values of the comparison algorithms for varying numbers of recommendations with the *Foursquare* dataset.



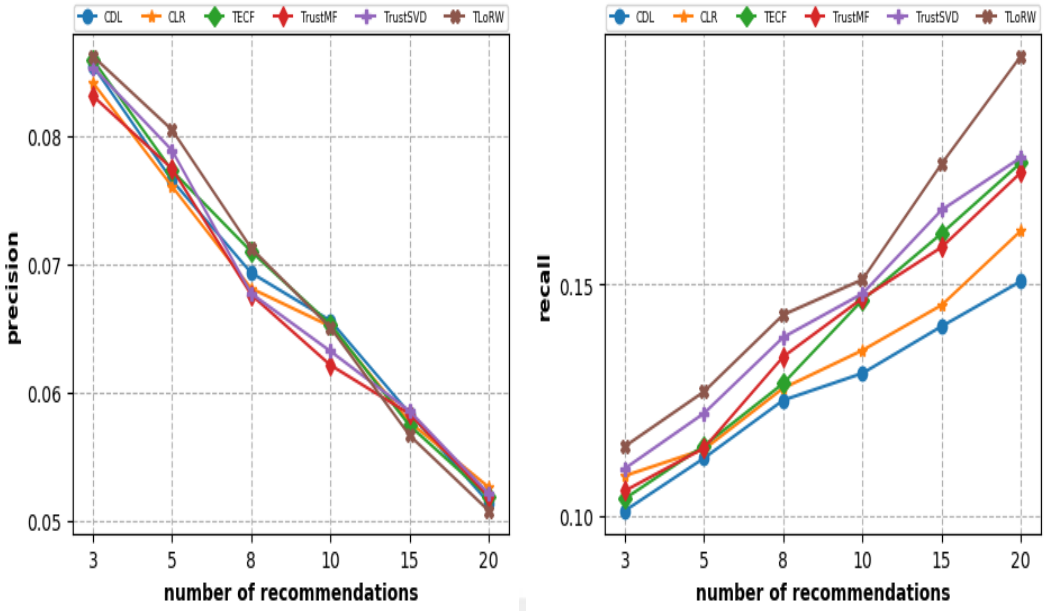
(a) Precision

(b) Recall



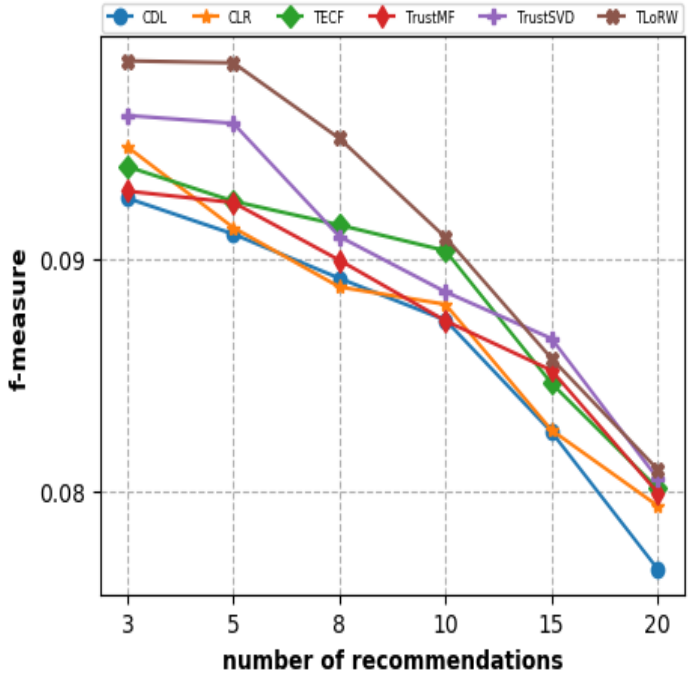
(c) F-measure

Figure 4.6: Precision, recall and f-measure values of the comparison algorithms for varying numbers of recommendations for the *Gowalla* dataset.



(a) Precision

(b) Recall



(c) F-measure

Figure 4.7: Precision, recall and f-measure values of the comparison algorithms for varying numbers of recommendations with the *Wee Places* dataset.

TLoRW is not dataset dependent. When we analyze the performance of the state-of-the-art methods on datasets, we see that TECT, TrustSVD, and TrustMF perform better on the Foursquare and Gowalla datasets, but CLR and CDL perform better on the Brightkite dataset. This may have resulted from the fact that the Foursquare and Gowalla datasets have relatively higher social relation statistics than the Brightkite dataset. Performance values are very close to each other and even overlap on the Wee Places dataset, which has the highest *check-ins per eser* value. The results also show that the *check-ins per user* value affects the recommendation accuracy. The recommendation accuracy improves while the number of *check-ins per user* is increasing. Moreover, when the number of recommendations is low, the experimental results are more distinctive, but with the increase in the number of recommendations, all techniques produce similar results.

4.4 Evaluation of SgWalk

In this section, we present the experiments conducted for empirical evaluation of our approach, SgWalk. We first describe the datasets used in the experiments and evaluation metrics. Then, our experimental setup and parameter settings are presented. In the experiments, we first present the evaluation results on the effects of two key parameters, vector dimension and window size, on the quality of the learned embeddings. Then, we compare our proposed technique against the state-of-art methods.

4.4.1 Evaluation methodology and parameter settings

SgWalk is a subgraph-based approach, and subgraphs are determined according to the check-ins clustered with respect to the spatial information. The size of the created clusters needs to be relatively small compared with the whole graph for SgWalk in order to capture the contextual relationships and to execute efficiently. Therefore, as the clustering algorithm, we use DBSCAN [74]. We prefer this algorithm since it does not require the number of clusters in advance, and it determines clusters based on the concepts of neighborhood and density. To fulfill this, DBSCAN needs two parameters: the radius of the neighborhood and the minimum number of points in

a neighborhood. In the experiments, we set the minimum number of points as 3 and the neighborhood radius as 2,000 m, which is also consistent with the optimal neighborhood size in [78].

We want to highlight the inclusion of DBSCAN in our work as it is used for different purposes in the proposed approaches. In TLoRW, DBSCAN is used to determine the starting coordinates for the cases to generate recommendations.Hence it is only part of the analysis process. On the contrary, in SgWalk clustering is crucial in subgraph construction step. Here, we use DBSCAN as the clustering algorithm.

Once the clustering is performed, the center of each cluster is considered the current location of the user to define the spatial context of the user and the vicinity for recommendation. The clustered user data are sorted by check-in date from old to recent. The sorted check-in data for each cluster are partitioned into training and test datasets such that the old data constitute the training dataset since we aim to find the next location recommendations. Data are partitioned such that 80% of the data per user is used for training and 20% is used for testing. The experiments are conducted under 5-fold cross validation. We set the walk count r to 10 per subgraph and the length of walk sequence s to 40 in random walk generation processes.

4.4.2 Analysis of the Effect of Adding Different Node Types

As the first analysis, we compare the recommendation accuracy under subgraphs A, B, C and D illustrated in Figures 3.2a, 3.2b, 3.2c, and 3.2d, respectively, to assess the impact of adding a new type of node. The results of the experiments performed on the Brightkite, Foursquare, Gowalla, and Wee Places datasets are given in Table 4.6. The results clearly indicate that the types of nodes affect the recommendation performance in terms of precision, recall and f-measure metrics. Friendship relations exist on four datasets, so we set them as the baseline subgraph. Then, we investigate the impacts of 2 latent node types: expert and trusted users. As expected, the subgraph having only friend nodes and its check-ins (subgraph A) has the lowest recommendation accuracy in the experiments among all subgraphs. It can be concluded that the friends of a user cannot cover a sufficient number of locations for the user's current context.

Table 4.6: Recommendation performance under varying k values for the effect of different node types

		Subgraph A			Subgraph B			Subgraph C			Subgraph D		
	@k	precision	recall	f-measure	precision	recall	f-measure	precision	recall	f-measure	precision	recall	f-measure
Brightkite	3	30.11	19.42	23.61	30.68	19.79	24.06	30.89	19.92	24.22	31.15	20.09	24.42
	5	21.01	22.57	21.76	21.75	23.37	22.53	21.72	23.34	22.50	22.22	23.88	23.02
	8	14.57	25.05	18.42	15.25	26.22	19.29	14.96	25.72	18.92	15.54	26.73	19.65
	10	12.04	25.89	16.44	12.57	27.03	17.16	12.43	26.73	16.97	12.90	27.73	17.61
	15	8.32	26.83	12.70	8.68	28.01	13.25	8.71	28.07	13.29	9.02	29.07	13.77
	20	6.39	27.46	10.37	6.61	28.44	10.73	6.62	28.47	10.74	6.92	29.75	11.23
Foursquare	3	6.53	3.18	4.28	9.24	4.49	6.05	10.51	5.11	6.88	12.24	5.96	8.01
	5	4.90	3.97	4.39	6.77	5.49	6.06	7.75	6.28	6.94	9.28	7.52	8.31
	8	3.52	4.56	3.97	4.81	6.25	5.43	5.61	7.27	6.33	6.68	8.66	7.54
	10	2.98	4.83	3.68	4.07	6.61	5.03	4.71	7.62	5.81	5.61	9.11	6.94
	15	2.12	5.16	3.01	2.90	7.05	4.11	3.31	8.04	4.68	3.94	9.61	5.59
	20	1.64	5.32	2.51	2.22	7.21	3.39	2.52	8.16	3.85	3.02	9.81	4.62
Gowalla	3	6.53	3.18	4.28	9.24	4.49	6.05	10.51	5.11	6.88	12.24	5.96	8.01
	5	4.90	3.97	4.39	6.77	5.49	6.06	7.75	6.28	6.94	9.28	7.52	8.31
	8	3.52	4.56	3.97	4.81	6.25	5.43	5.61	7.27	6.33	6.68	8.66	7.54
	10	2.98	4.83	3.68	4.07	6.61	5.03	4.71	7.62	5.81	5.61	9.11	6.94
	15	2.12	5.16	3.01	2.90	7.05	4.11	3.31	8.04	4.68	3.94	9.61	5.59
	20	1.64	5.32	2.51	2.22	7.21	3.39	2.52	8.16	3.85	3.02	9.81	4.62
Wee Places	3	10.24	6.34	7.83	11.68	7.24	8.94	11.84	7.34	9.06	12.08	7.48	9.24
	5	7.19	7.42	7.30	8.20	8.47	8.33	8.38	8.65	8.51	8.53	8.80	8.66
	8	4.95	8.18	6.17	5.60	9.24	6.97	5.79	9.56	7.21	5.86	9.69	7.30
	10	4.11	8.49	5.54	4.65	9.59	6.26	4.81	9.94	6.48	4.85	10.01	6.53
	15	2.88	8.93	4.35	3.24	10.03	4.90	3.33	10.32	5.04	3.38	10.47	5.11
	20	2.23	9.19	3.59	2.48	10.25	3.99	2.55	10.54	4.11	2.60	10.74	4.19

Experimental results for subgraph B and subgraph C indicate that both subgraphs acquire higher performance values than subgraph A. In addition, subgraph B performs better than subgraph C on all datasets. It is reasonable since if a location is visited by an expert, then it has a higher possibility of being visited by the other users. Additionally, trusted users may recommend essential places, so their contribution is not as high as that of experts. As the last option, we consider the subgraph containing all node types together (subgraph D), and then we obtain the best performance in four datasets. Experts and trusted users have different check-in behaviors, and including

these latent nodes improves the performance from popularity and trustworthiness aspects. These results also indicate that combining disjoint types (friend, expert, trusted user) improves the accuracy of the recommended locations.

In all datasets, we observe a similar performance pattern. It is an important observation revealing that the performance of SgWalk is not dataset-specific. Although all datasets' statistics differ in terms of check-ins per user and friends per user values, subgraph D, which contains friend, trusted user and expert together, attains the best performance in all of the test cases. Furthermore, the experimental results are more distinctive for a small number of recommendations, but all subgraphs yield similar results by increasing the recommendation number. Subgraph D has the best f-measure value with a recommendation count of 3 for Brightkite and Wee Places and 5 for Foursquare and Gowalla, and the performance of the top 5 in Brightkite and Wee Places is closer to the top 3 than the top 8. Therefore, we choose the top 5 as the recommendation count in the following experiments.

4.4.3 Parameter Tuning Experiments

In this section, we present the results of tuning the parameters of our embedding model described in Section 3.4.4. In particular, we examine the impact of window size w and the embedding vector dimension d when we train the model using the skip-gram language model. The candidate values of these two variables are w = 1, 2, 3, 4, 5, 6, 7, 8 and d = 100, 120, 150, 200, 250.

Best values for the w and d parameters differs in the literature and obtained by tuning process. For example, w parameter value is set to 5 in [67] , 10 in [46, 24] or 20 in [81] and d parameter value is set to 100 in [30], 128 in [24, 46] or 200 in [32, 81] as best values for the maximum performance of the technique. Here, we examine the different values of the window size w and the dimension size d, i.e., (w, d) pairs that reach the best values per cost in terms of the f-measure. Table 4.7 shows the SgWalk's f-measure values for the top@5 predictions for different values of w and d.

It is noticeable that as the dimension d increases until approximately 200, the performance increases as well. Increasing the dimension of the vector has an inverse

Table 4.7: Recommendation performance (in f-measure) under varying window size w and dimension size d values

	brightkite@5					foursquare@5				
w/d	100	120	150	**200**	250	100	120	150	**200**	250
1	2.98	3.01	3.17	3.18	3.11	0.22	0.24	0.26	0.27	0.31
2	27.48	27.51	27.54	**27.61**	27.48	11.3	11.77	11.77	**12.19**	12.05
3	20.72	21.3	21.76	21.85	21.79	8.03	7.99	7.9	7.95	7.91
4	23.4	22.66	22.92	23.21	23.34	9.2	9.19	9.03	8.86	9.14
5	18.78	18.65	18.88	18.94	18.68	6.48	6.57	6.29	6.3	6.19
6	19.91	20.11	20.04	20.08	19.98	6.99	7.08	7.06	7.09	6.92
7	17.65	17.81	17.58	17.23	17.52	5.77	5.75	5.82	5.52	5.72
8	18.62	18.56	18.26	18.39	18.07	6.36	6.06	6.4	6.29	6.07

	gowalla@5					weeplaces@5				
w/d	100	120	150	**200**	250	100	120	150	**200**	250
1	0.49	0.49	0.49	0.51	0.56	0.52	0.54	0.55	0.63	0.63
2	10.44	10.48	10.62	**10.68**	10.65	6.01	6.47	6.96	**7.53**	7.48
3	8.48	8.56	8.60	8.87	8.81	5.33	5.34	5.35	5.38	5.37
4	8.94	8.81	8.99	8.88	8.82	5.29	5.42	5.50	5.20	5.12
5	7.13	7.41	7.47	7.57	7.51	4.36	4.21	4.07	3.52	3.17
6	7.63	7.66	7.39	7.51	7.67	3.81	3.76	3.55	3.14	2.72
7	6.78	6.69	6.77	6.70	6.84	3.31	3.05	2.77	2.33	2.11
8	6.52	6.66	6.63	6.68	6.68	2.80	2.70	2.50	2.05	1.94

effect after the convergence point. Therefore, we set the dimension size to 200 for all datasets.

We also observe that the window size w has an interesting effect on the f-measure value. It converges alternatingly according to whether w is odd or even (Figure 4.8, Figure 4.9, Figure 4.10 and Figure 4.11). This is due to the nature of the generated walk sequences, which contain mostly user and location nodes alternatingly as user-user nodes occurrence frequency is too smaller than user-location nodes. The f-measure has the highest value at w=2 and drops and increases gradually as long as the window size w increases. Having lower w values also reduces the computational cost significantly. In Tables 4.7, we highlight the parameter values that achieve the best performance and are used for the rest of the evaluation.

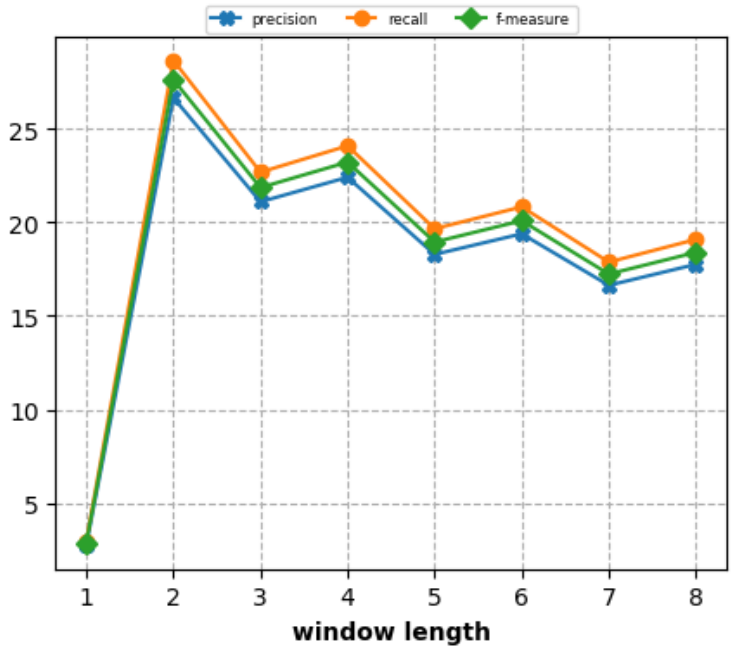


Figure 4.8: Effect of changing window size for dimension = 200 for the Brightkite dataset

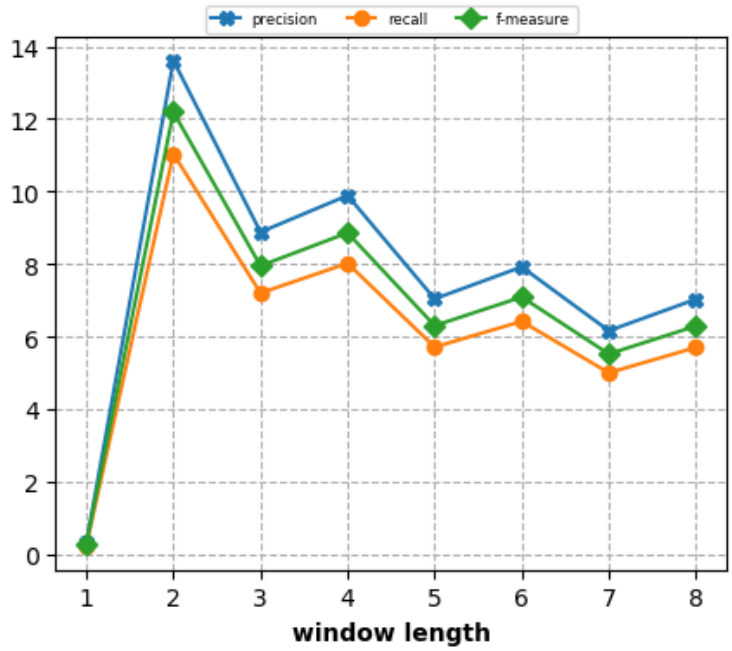


Figure 4.9: Effect of changing window size for dimension = 200 for the Foursquare dataset

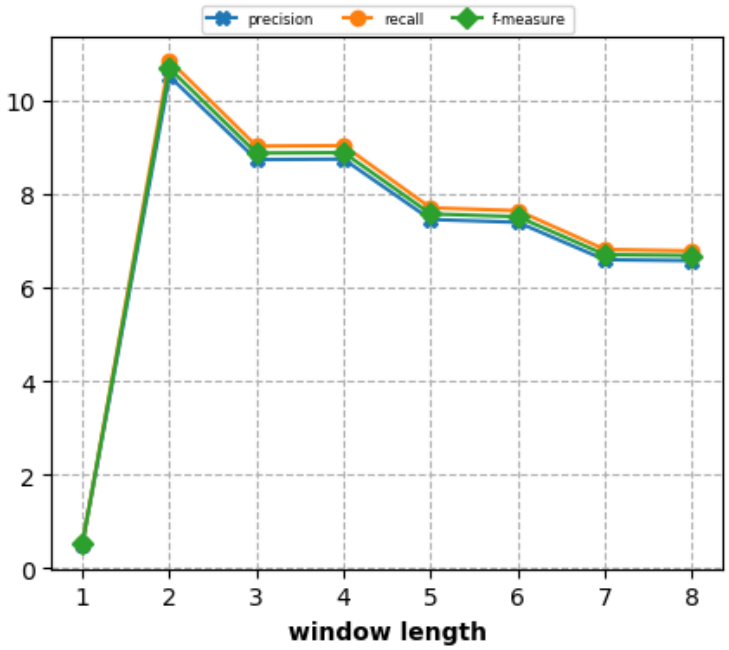


Figure 4.10: Effect of changing window size for dimension = 200 for the Gowalla dataset

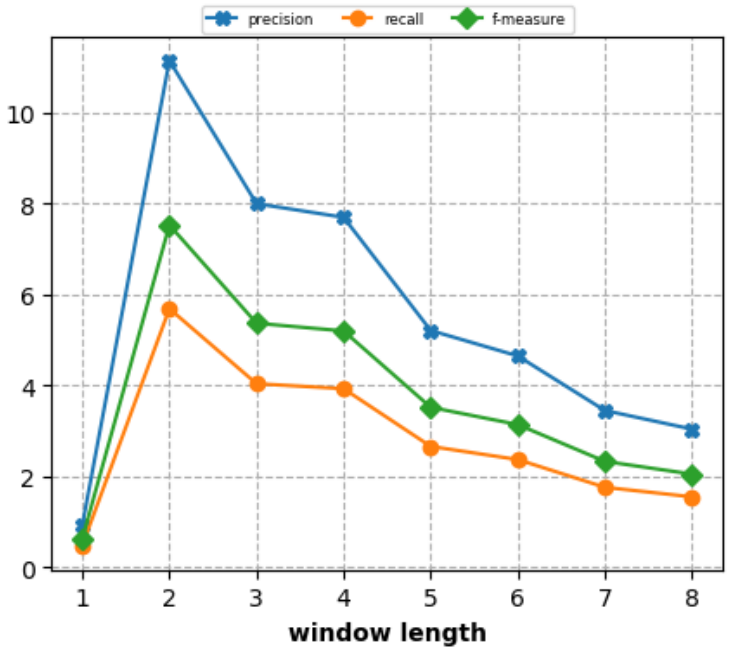


Figure 4.11: Effect of changing window size for dimension = 200 for Wee Places dataset

```
set SUBFILENAME=%CUR_YYYY%%CUR_MM%%CUR_DD%-%CUR_HH%%CUR_NN%%CUR_SS%

set dataset=gw
set mode=7

set dimension=200

set dataset=wp

set window=2
start cmd /c C:\Python27\python.exe __main__.py --window-size %window%
 --mode %mode% --dataset %dataset% --representation-size %dimension%
^> out_%dataset%_m%mode%_w%window%_d%dimension%_%SUBFILENAME%.txt

set window=3
start cmd /c C:\Python27\python.exe __main__.py --window-size %window%
 --mode %mode% --dataset %dataset% --representation-size %dimension%
^> out_%dataset%_m%mode%_w%window%_d%dimension%_%SUBFILENAME%.txt

set window=4
start cmd /c C:\Python27\python.exe __main__.py --window-size %window%
 --mode %mode% --dataset %dataset% --representation-size %dimension%
^> out_%dataset%_m%mode%_w%window%_d%dimension%_%SUBFILENAME%.txt

set window=5
start cmd /c C:\Python27\python.exe __main__.py --window-size %window%
 --mode %mode% --dataset %dataset% --representation-size %dimension%
^> out_%dataset%_m%mode%_w%window%_d%dimension%_%SUBFILENAME%.txt

set window=6
start cmd /c C:\Python27\python.exe __main__.py --window-size %window%
 --mode %mode% --dataset %dataset% --representation-size %dimension%
^> out_%dataset%_m%mode%_w%window%_d%dimension%_%SUBFILENAME%.txt

set window=7
start cmd /c C:\Python27\python.exe __main__.py --window-size %window%
 --mode %mode% --dataset %dataset% --representation-size %dimension%
^> out_%dataset%_m%mode%_w%window%_d%dimension%_%SUBFILENAME%.txt

set window=8
start cmd /c C:\Python27\python.exe __main__.py --window-size %window%
 --mode %mode% --dataset %dataset% --representation-size %dimension%
^> out_%dataset%_m%mode%_w%window%_d%dimension%_%SUBFILENAME%.txt
```

Figure 4.12: Batch script used for test execution

4.4.3.1 Parameter Tuning Process

Test execution for parameter tuning is handled manually. The proposed algorithm implementation is executed in a parameterized way, as shown in Figure 4.12, so that we can run parallel executions without affecting each other. Execution results of precision and recall values are recorded into the log files once the run finishes success-

fully, as shown in 4.13. After all executions are finished, we collect all the results in a spreadsheet and pick the maximum performance values with the help of spreadsheet functions.

After the experiment runs are completed, we examine the accuracy results and check if the maximum value is obtained by the boundary values of the parameters. If so, we conduct new tests by increasing or decreasing the value of the parameter according to the boundary condition (being either min or max value in the previous experiment run). We continue until the parameter value that provides the maximum accuracy falls within the minimum and the maximum values (i.e. not a boundary value). For example, for vector dimension size tuning, we started with dimensions 100, 120, and 150, but the maximum value was obtained at the boundary of the dimension size values. For this reason, we conducted new tests with dimension 200. Nevertheless, the maximum value was still obtained at the boundary. And then, we executed new tests with dimension 250. Finally, the dimension size setting providing the maximum accuracy fell inside the minimum and maximum boundaries. Therefore we concluded that the optimal dimension value is 200 for the SgWalk method.

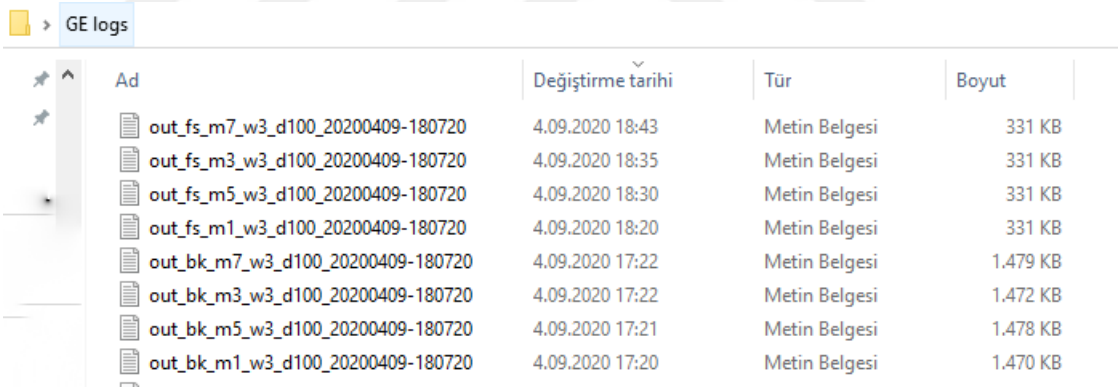


Figure 4.13: Test execution logs

4.4.4 Comparative Accuracy Performance Analysis

We analyze the accuracy performance of SgWalk under a varying number of recommendations against the state-of-the-art methods from the literature that were developed as graph embedding techniques. The details of the compared methods are listed as follows:

- **DeepWalk [24]** learns node embeddings by first performing classical random walks on an input graph and then feeds the generated random walks to a skip-gram model. DeepWalk was originally designed for homogeneous graphs, so it is applied to a heterogeneous graph by ignoring the heterogeneity and treating all nodes and edges of the graph as being of the same type. We set the number of walks per node r = 10, length of walk sequence s=40, window size w = 10 and embedding vector dimension d = 100 for skip-gram

- **PTE [29]** is a semisupervised model for learning text embeddings using both labeled and unlabeled data. In these experiments, we used PTE in an unsupervised way. Specifically, for a heterogeneous graph, we created bipartite subgraphs as follows: (U)ser-(L)ocation, (U)ser-(F)riend, (U)ser-(T)rusted, (U)ser-(E)xpert, and then fed these graphs to PTE to output the node embeddings.

- **Metapath2vec [25]** generates random walks depending on a specific metapath and feeds the generated sequences to a skip-gram model. In these experiments, we used four different metapaths: "U-L-U" for representing different users visiting the same location, "U-F-L-U" for representing locations visited by friends of users, "U-T-L-U" for representing locations visited by trusted users, and "U-E-L-U" for representing locations visited by expert users. For random walk generation and the skip-gram process, we used the same values in the DeepWalk case for parameters r, s, w, and d.

- **HIN2Vec [26]** forms metapath guided random walks from the combined set of metapaths shorter than a specified length and jointly learns both node embeddings and metapath embeddings. In these experiments, we set the maximum metapath length to 3 and provided eight different edge types: "U-L", "L-U", "U-F", "F-L", "U-T", "T-L", "U-E" and "E-L". For random walk generation and the skip-gram process, we used the same values in the DeepWalk case for parameters r, s, w, and d.

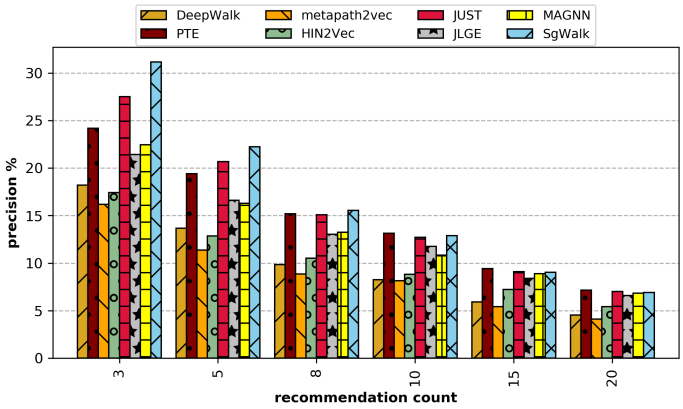
- **JUST [46]** is a heterogeneous graph embedding technique using random walks with the jump and stay strategies to learn node embeddings more efficiently. In these experiments, we set the stay or jump parameter across domains α = 0.5 and provided five domains as follows: (U)ser, (L)ocation, (F)riend, (T)rusted

user, and (E)xpert user with the allowed transitions between domains. For random walk generation and the skip-gram process, we used the same values in the DeepWalk case for parameters r, s, w, and d.

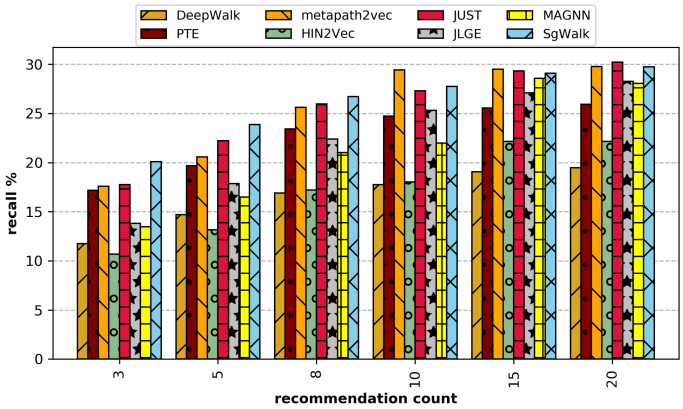
- **JLGE [32]** is a spatial-temporal graph-based model that recommends location with learnt embeddings. It extends the LINE method on LBSN via multiple bi-partite graphs according to (U)ser, (L)ocation and (T)ime. In the experiments, we created six bi-partite graphs (U-U, U-L, U-T, L-U, L-L, L-T) as proposed in original work, for one month periods. We used the parameter values as reported in the original work such that d is set as 200, s is 100, ρ is 0.025, and negative edge count is 5.

- **MAGNN [67]** is the metapath aggregated graph neural network for heterogeneous graph embedding. MAGNN first applies type-specific linear transformations for node content transformation via intrametapath aggregation and utilizes intermetapath aggregation to generate node embeddings. In these experiments, we used eight different metapaths, "U-L-U", "U-F-L-U", "U-T-L-U", "U-E-L-U" (same as in metapath2vec experiments), and their location counterparts were "L-U-L", "L-U-F-L", "L-U-T-L", and "L-U-E-L". We set the best values for the GNN parameters (dropout rate, learning rate, weight decay, etc.) suggested in the paper and used the same value in previous cases for embedding the dimension parameter d.

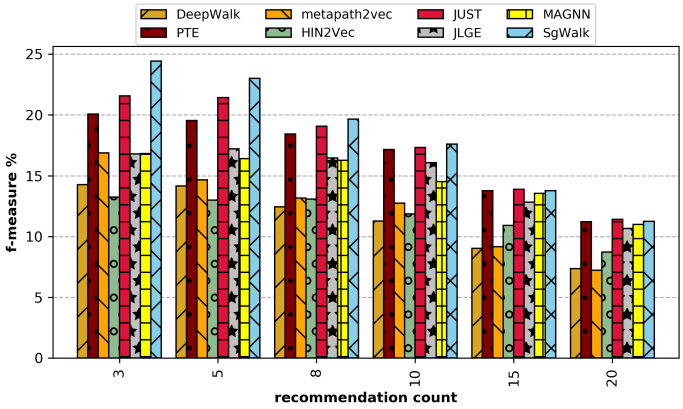
The results of the experiments performed on the Brightkite, Foursquare, Gowalla and Wee Places datasets are given in Figure 4.14, Figure 4.15, Figure 4.16 and Figure 4.17, respectively. The results clearly indicate that SgWalk continuously performs better than the compared methods in terms of precision, recall, and f-measure metrics. We believe that the proposed SgWalk considers the distinctive structural characteristics and semantic information by employing friends of users, trusted users, and expert users in the vicinity, which effectively guarantees the embedding accuracy of LBSN datasets via heterogeneous network embedding. JLGE has the second-best accuracy value in the evaluation. However, JUST provides higher recommendation accuracy than JLGE and PTE on the Brightkite dataset and MAGNN has better results under high number of recommendation on Foursquare and Gowalla. JLGE, which is not



(a) Precision

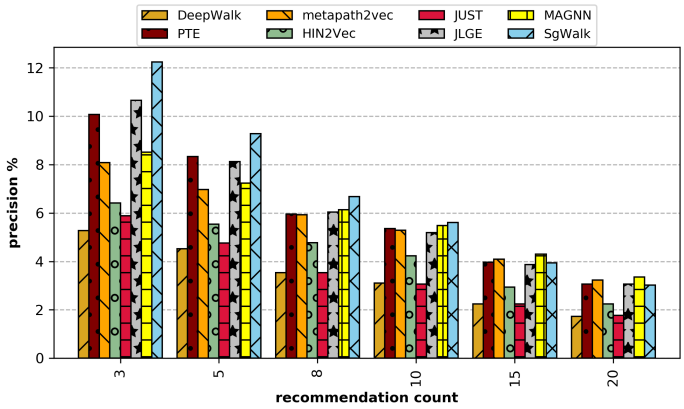


(b) Recall

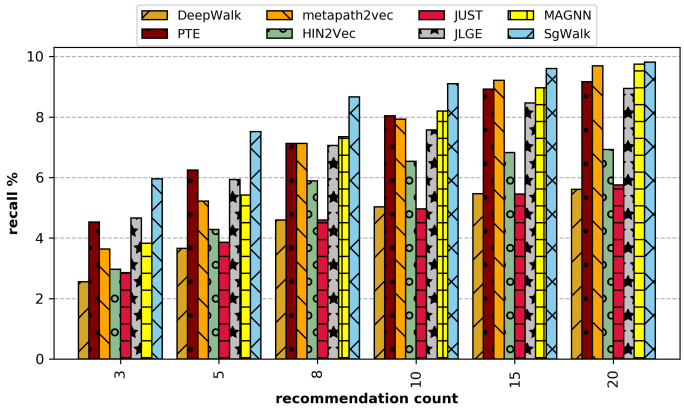


(c) F-measure

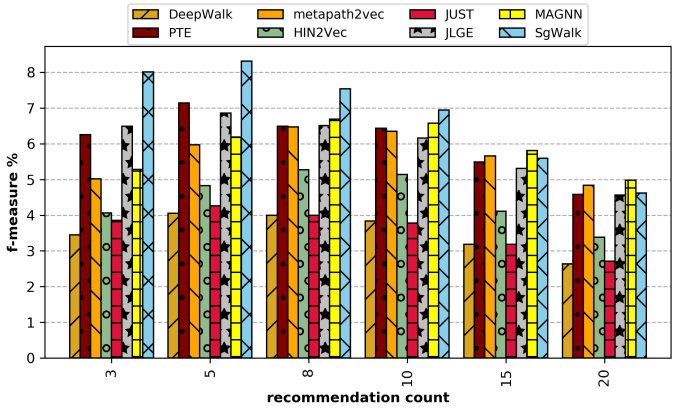
Figure 4.14: Precision, recall and f-measure values of the comparison algorithms for varying numbers of recommendations with the *Brightkite* dataset.



(a) Precision

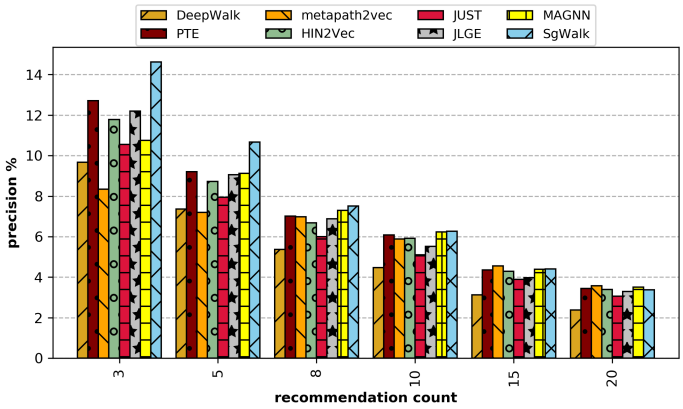


(b) Recall

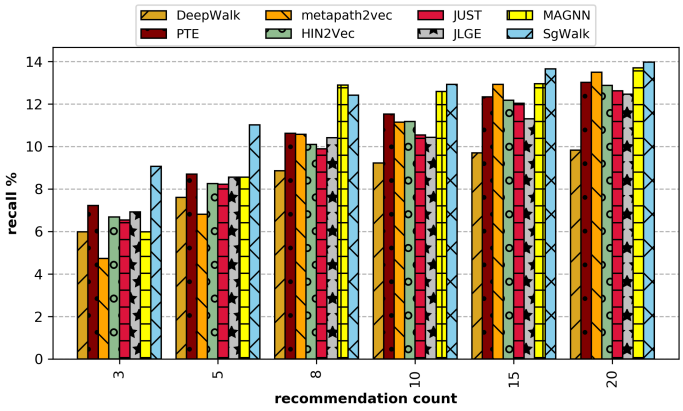


(c) F-measure

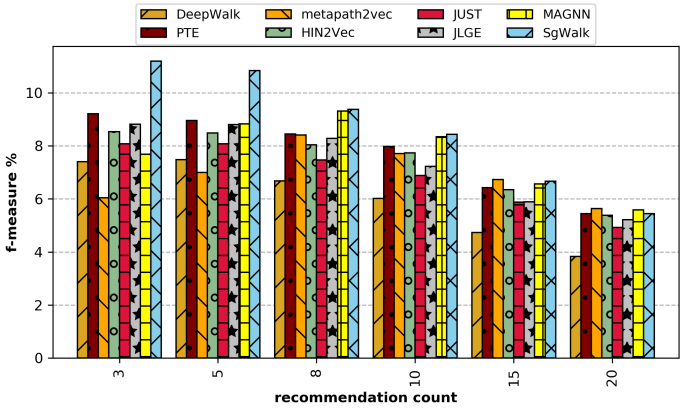
Figure 4.15: Precision, recall and f-measure values of the comparison algorithms for varying numbers of recommendations with the *Foursquare* dataset.



(a) Precision

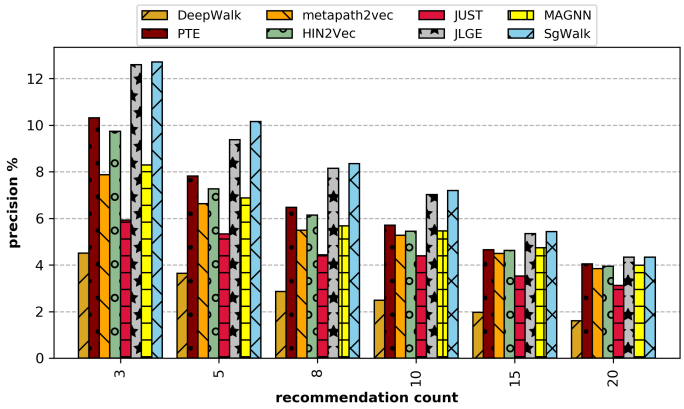


(b) Recall

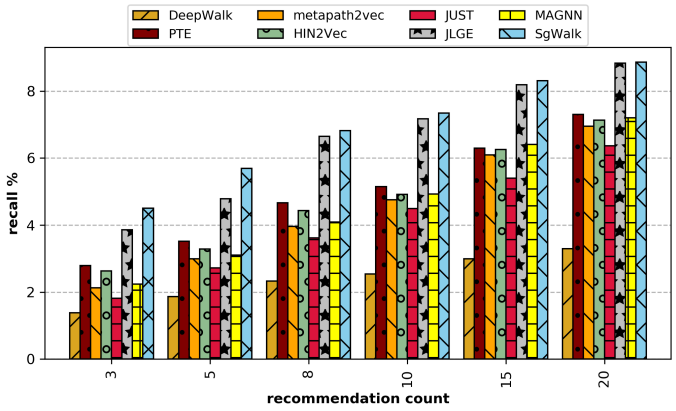


(c) F-measure

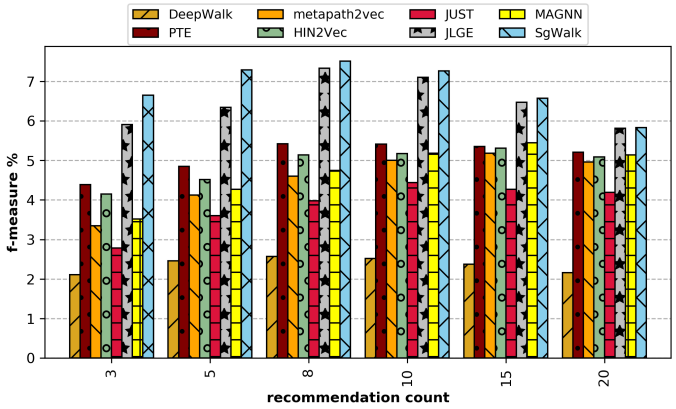
Figure 4.16: Precision, recall and f-measure values of the comparison algorithms for varying numbers of recommendations with the *Gowalla* dataset.



(a) Precision



(b) Recall



(c) F-measure

Figure 4.17: Precision, recall and f-measure values of the comparison algorithms for varying numbers of recommendations with the *Wee Places* dataset.

a random walk based approach and designed for location recommendation, on average, performs better than other random walk based approaches JUST, metapath2vec, HIN2Vec, and DeepWalk. Among the meta-path based approaches MAGNN, Metapath2vec and HIN2Vec, MAGNN is the best with the help of deep learning techniques and metapath2vec performs better than HIN2Vec under fewer number of recommendations. On the other hand, HIN2Vec first catches and then beats metapath2vec as the recommendation count increases. DeepWalk has the lowest accuracy since it is originally designed for homogeneous graphs.

As a summary, accuracy improvement with respect to the compared methods (in terms of f-measure under @5 recommendations) across the datasets are given in Table 4.8. It is observed that SgWalk provides the maximum improvement against DeepWalk with 102% on average. The minimum improvement is over JLGE with 23% on average. These results show a strong indication that SgWalk, focusing on spatial context and generating node embeddings by using subgraphs with respect to spatial context, can capture the contextual relationships more effectively.

Table 4.8: SgWalk Improvement Percentages in f-Measure @5

	DeepWalk	PTE	Metapath2Vec	HIN2Vec	JUST	JLGE	MAGNN
Brightkite	63%	18%	57%	77%	7%	34%	40%
Four Square	105%	16%	39%	72%	95%	21%	34%
Gowalla	45%	21%	55%	28%	34%	23%	23%
Wee Places	197%	50%	77%	61%	102%	15%	71%
Average	**102%**	26%	57%	60%	60%	**23%**	42%

The performance of the techniques varies depending on the characteristics of the dataset used (given in Table 4.2). However, on both datasets, SgWalk attains the best performance values in all test cases. Hence, we can conclude that the performance of SgWalk is not dataset dependent. When we analyze the performance of the state-of-the-art methods on datasets, we see that DeepWalk, HIN2VEC and MAGNN perform better on the Foursquare dataset, PTE and metapath2vec perform better on the Gowalla dataset, JUST and JLGE perform better on the Brightkite and Wee Places datasets, respectively. This shows the characteristics of the datasets has an effect on

the methods. The results also show that *Check-ins per User* value affects the recommendation accuracy. The recommendation accuracy increases as the number of *Check-ins per User* increases. Moreover, when the number of recommendations is low, accuracy results are more distinctive, but with the increase in the number of recommendations, all techniques produce similar results.

4.4.5 Validity Threats and Limitations

SgWalk is a subgraph-based approach, and subgraphs are determined according to the check-ins clustered with respect to the spatial information. The size of the created clusters needs to be relatively small compared with the whole graph for SgWalk in order to capture the contextual relationships and to execute efficiently. Therefore, the selected clustering algorithm should generate the clusters from the dataset that allows small-sized user subgraph construction. We used the density-based algorithm DBSCAN to create clusters, and SgWalk has the best result with the four datasets used in experiments, with minimum 7% improvement over all cases under @5 recommendations. On the other hand, if the method generates large clusters failing to fulfill small-sized subgraphs construction, possibly due to the density distribution of the dataset, SgWalk may not capture contextual relationships and perform as expected.

Since clustering is a crucial step in SgWalk, it constitutes a vulnerable point whose quality can effect the outcome. In the proposed setting, since the challenged problem is location recommendation for LBSN, the clustering is built on spatial feature to extract the spatial context. For different settings and different contexts, other features of the social network can be user to decompose the full graph into small-sized subgraphs.

4.5 Statistical Significance Test for Evaluation Results

Statistical significance testing is widely used in the literature to assess the difference in results of the proposed approaches against the random selection of input. According to recent surveys on information retrieval, the t-test is the most commonly used method for statistical significance in these studies [82, 83, 84]. The t-test measures

the difference in performance values between two samples. A p-value quantifies the probability of the null hypothesis being true as a more extreme value. P-values greater than a threshold suggest that our observation was unlikely to have been due to chance. So, we are not rejecting the null hypothesis of evaluation results. By checking the p-value if it's below our threshold, we will have evidence against the null hypothesis of the evaluation result. Two types of t-tests are dependent and independent according to the values that are obtained from the same or different environments. In this work, we evaluated all approaches with the same dataset, so we assessed the statistical significance of the results with a dependent t-test.

The significance test is applied as follows:

- The f-measure metric is used as the value for investigating the difference of significance in results
- Our null hypothesis is that there is no difference between our proposed methods and the other compared approaches.
- A significance level is computed by taking the value of the f-measure with respect to different recommendation counts in our experiments.
- The significance test is applied to each dataset separately.
- T-test score (t-value) and p value are used for significance determination.
- When the significance level is low, the null hypothesis is rejected and proven that the proposed approach achieves statistically significant improvements.

We used a Python script to calculate the t-value and p value with the help of the *ttest_rel* method from the *scipy.stats* package.

4.5.1 Significance of TLoRW Results

We assessed the significance of TLoRW under the null hypothesis, as there is no difference in our proposed TLoRW method and the other compared approaches CDL, TECF, TrustMF, TrustSVD and CLR. We used the significance level α = .05 and

found the critical value = 2.571 to determine the significance of the TLoRW results. The t-test assessment results of TLoRW are given in Table 4.9 and significant values are marked with * in the table. TLoRW results are significantly different from all compared approaches in Brightkite and Gowalla datasets and from CDL, TECF and CLR approaches in all datasets. According to t-test scores, TLoRW results are statistically significant in 17 of 20 cases covering 85% of the comparison results. This proves that the null hypothesis is incorrect and that TLoRW has significant improvements.

Table 4.9: t-test assessment results of TLoRW

	Brightkite		Foursquare		Gowalla		Wee Places	
	t-value	p value	t-value	p value	t-value	p value	t-value	p value
CDL	6.440*	0.001*	11.476*	0.000*	10.108*	0.000*	7.552*	0.001*
TECF	3.211*	0.024*	6.581*	0.001*	5.675*	0.002*	3.004*	0.030*
TrustMF	3.346*	0.020*	2.161	0.083	7.914*	0.001*	3.736*	0.013*
TrustSVD	3.030*	0.029*	1.332	0.240	16.391*	0.000*	2.512	0.054
CLR	2.844*	0.036*	5.842*	0.002*	8.392*	0.000*	4.671*	0.005*

4.5.2 Significance of SgWalk Results

We assessed the significance of SgWalk the null hypothesis, as there was no difference in our proposed method SgWalk and the other compared approaches DeepWalk, PTE, JUST, metapath2vec, HIN2vec, JLGE, MAGNN. We used the significance level α = .05 and found the critical value = 2.571 to determine the significance of the SgWalk results. The t-test assessment results of SgWalk is given in Table 4.10 and significant values are marked with * in the table. SgWalk results are significantly different from DeepWalk and HIN2Vec approaches in all datasets. According to t-test scores, SgWalk results are statistically significant in 21 of 28 test cases covering 75% of the comparison results. This proves that the null hypothesis is incorrect and that SgWalk has significant improvements.

Table 4.10: t-test Assessment Results of SgWalk

	Brightkite		Gowalla		Four Square		Wee Places	
	t-value	p-value	t-value	p-value	t-value	p-value	t-value	p-value
DeepWalk	7.041*	0.001*	7.992*	0.000*	7.769*	0.001*	23.021*	0.000*
PTE	2.074	0.093	2.807*	0.038*	2.647*	0.046*	6.150*	0.002*
JUST	1.728	0.145	8.724*	0.000*	4.19*	0.009*	8.270*	0.000*
metapath2Vec	8.334*	0.000*	2.100	0.090	1.912	0.114	5.674*	0.002*
HIN2Vec	4.443*	0.007*	5.242*	0.003*	2.817*	0.037*	6.071*	0.002*
JLGE	2.799*	0.038*	3.488*	0.017*	3.965*	0.011*	2.269	0.073
MAGNN	2.781*	0.039*	1.776	0.136	1.541	0.184	5.083*	0.004*

CHAPTER 5

CONCLUSIONS

The data collected from LBSNs contain rich information that constitutes a basis for building location recommendation systems. In this work, we aim to develop a recommendation system from LBSNs according to the user's current context to suggest locations. We represent the LBSN data by an undirected and unweighted graph model, which includes nodes from different contexts and relationships of these nodes. Based on the LBSN graph model, we introduce two novel context-aware location recommendation techniques, *TLoRW* and *SgWalk*, to generate accurate recommendation lists.

First, *TLoRW* is a trust-aware location recommendation technique and utilizes the trustworthiness values of users. There are two significant contributions: the trust metric for users in LBSNs and a trust-aware algorithm that improves the accuracy of location recommendations. For trust prediction, most of the approaches in the literature find/fill missing trust data depending on the transitivity of the trust relation. In our method, the trustworthiness of users is computed from two trust features, namely, objectivity and consistency, regardless of the trustor or trustee data. Furthermore, in trust-availed datasets, trust data are provided in binary format, i.e., yes or no, due to privacy concerns, but in our method, numerical values can be calculated.

TLoRW experiments also reveal that recommendation accuracy is improved by utilizing trust and expert notions together. In the proposed recommendation system, we construct an undirected graph to model an LBSN that consists of users and locations. The graph is enriched with expert users, trusted users, and their popular locations in addition to regular users and locations. By applying a random walk on the generated user-specific graph, we rank the locations based on visit counts.

We evaluate our recommendation technique, TLoRW, by comparing it with the state-of-the-art methods on well-known datasets filtered for New York City. The experiments show that our proposed algorithm, TLoRW, brings minimum 5% improvement for location recommendation accuracy in terms of f-measure@5. The analysis on the results shows that the improvement by TLoRW is statistically significant (α = .05) for 85% of the test cases. The results indicate that the trust score is useful in location recommendation when combined with other notions of friends and experts in LBSNs. For the rest of the compared techniques, their performances change with the dataset accordingly, but they perform better when the average check-ins per user are higher.

The proposed trust method in TLoRW is also quite suitable for the methods, which depends on explicit trust data as trust-aware recommendation systems [13, 16, 19, 21, 22]. With our work's contribution, these methods can operate on the dataset when there are no explicit trust data in the dataset. For the cold start cases (when a user has (i) very few friends or no friends at all, (ii) very few check-ins or no check-in at all), integrating the experts in the vicinity, as well as trusted users, can improve recommendation accuracy.

The second proposed algorithm, *SgWalk* includes a user subgraph-based graph embedding technique for location recommendation, which utilizes the user subgraphs instead of bipartite subgraphs proposed in the literature, and does not depend on metapaths for random walk generation. The constructed user subgraph considers a wide range of information sources, including personal, social, spatial, and trustworthiness contexts, to generate a better recommendation list. The SgWalk technique follows four steps: (1) user subgraph construction, (2) random walk sequence generation, (3) learning graph embeddings, and (4) location recommendation list creation to explore the proximity between users and locations to provide recommendations.

Few research efforts exist in the literature that employ subgraphs in the embedding process for recommending locations to users. In these works, graphs are decomposed along with relationships or metapaths. SgWalk introduces a new subgraph creation process that decomposes the graphs according to the geographical attributes of the nodes, such that location nodes in the user's vicinity and user nodes (friend, expert, trusted) visit these locations. With this idea, we can model user preferences too.

Furthermore, the count of the nodes in the user subgraph is much fewer than the number of nodes in the bipartite graphs, which allows improved computation time.

We evaluate the SgWalk technique by comparing it with the state-of-the-art methods on well-known datasets filtered for New York City. The experiments show that our proposed algorithm, SgWalk, provides the highest accuracy against compared methods for majority of test cases, and the results indicate that our method brings minimum 23% improvement for location recommendation accuracy in terms of f-measure@5. The analysis on the results reveal that the obtained improvement by SgWalk is statistically significant (α = .05) for 75% of the test cases. The experimental results show a strong indication that SgWalk, focusing on spatial context and generating node embeddings by using subgraphs with respect to spatial context, can capture the contextual relationships more effectively, and hence it provides more accurate personalized and contextual location recommendations.

As s future work, other user features can be considered, such as user activity, competence, honesty, satisfaction, and centrality, to formulate trust scores depending on the domain. For example, for a movie dataset, such subgraphs can be constructed by using genre (with sub-genres) feature, if a variety of genres are provided within the dataset. Moreover, new trust modeling or uncertainty-aware trust relationships can be studied to improve recommendation accuracy. Additionally, user-based subgraph generation can be applied in different domains, such as movie and author databases, to recommend movies to users and authors for topics. Additionally, the effects of temporal context nodes in the user subgraph can be studied within the scope of LBSNs.

REFERENCES

[1] H. Gao, J. Tang, X. Hu, and H. Liu, "Modeling temporal effects of human mobile behavior on location-based social networks," in *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management*, CIKM '13, (New York, NY, USA), p. 1673–1678, Association for Computing Machinery, 2013.

[2] W. Sherchan, S. Nepal, and C. Paris, "A survey of trust in social networks," *ACM Comput. Surv.*, vol. 45, Aug. 2013.

[3] J.-H. Cho, K. Chan, and S. Adali, "A survey on trust modeling," *ACM Comput. Surv.*, vol. 48, Oct. 2015.

[4] D. Gambetta, "Can we trust trust?," in *Trust: Making and Breaking Cooperative Relations* (D. Gambetta, ed.), pp. 213–237, Blackwell, 1988.

[5] B. Lahno, "Olli lagerspetz: Trust. the tacit demand," *Ethical Theory and Moral Practice*, vol. 2, no. 4, pp. 433–435, 1999.

[6] J. B. Rotter, "Interpersonal trust, trustworthiness, and gullibility," in *Interpersonal trust, trustworthiness, and gullibility*, vol. 35(1), p. 1–7, 1980.

[7] J.-H. Cho, A. Swami, and I.-R. Chen, "A survey on trust management for mobile ad hoc networks," *IEEE Communications Surveys Tutorials*, vol. 13, no. 4, pp. 562–583, 2011.

[8] S. Adali, "Modeling trust context in networks," in *SpringerBriefs in Computer Science*, 2013.

[9] A. Zarghami, S. Fazeli, N. Dokoohaki, and M. Matskin, "Social trust-aware recommendation system: A t-index approach," in *2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*, vol. 3, pp. 85–90, 2009.

[10] H. Oh and S. Kim, "Identifying and exploiting trustable users with robust features in online rating systems," *TIIS*, vol. 11, no. 4, pp. 2171–2195, 2017.

[11] M. Eisend, "Source credibility dimensions in marketing communication-a generalized solution," *Journal of Empirical Generalisations in Marketing*, vol. 10, pp. 1–33, 01 2006.

[12] S. Rieh and D. Danielson, "Credibility: A multidisciplinary framework," *Annual Review of Information Science and Technology*, vol. 41, no. 1, pp. 307–364, 2007.

[13] C. Ju, J. Wang, and C. Xu, "A novel application recommendation method combining social relationship and trust relationship for future internet of things," *Multimedia Tools and Applications*, vol. 78, pp. 1–14, 08 2018.

[14] G. Guo, J. Zhang, and N. Yorke-Smith, "Trustsvd: Collaborative filtering with both the explicit and implicit influence of user trust and of item ratings," *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pp. 123–129, 01 2015.

[15] I. Ali, J. Hong, and S. Kim, "Exploiting implicit and explicit signed trust relationships for effective recommendations," *Proceedings of the Symposium on Applied Computing - SAC 17*, 2017.

[16] S. Ahmadian, M. Meghdadi, and M. Afsharchi, "Incorporating reliable virtual ratings into social recommendation systems," *Applied Intelligence*, vol. 48, pp. 4448–4469, Nov 2018.

[17] T. Shanmuigapriya and S. Swamynathan, "Reliability score inference and recommendation using fuzzy-based technique for social media applications," *Soft Computing*, vol. 22, pp. 8289–8300, Dec 2018.

[18] W. W, C. J, W. J, C. J, L. J, and G. Z, "Trust-enhanced collaborative filtering for personalized point of interests recommendation," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 9, pp. 6124–6132, 2020.

[19] Y. Jiang, H. Chen, and B. Yang, "Deep social collaborative filtering by trust," in *Proceedings of 2018 International Conference on Big Data Technologies*,

ICBDT '18, (New York, NY, USA), p. 52–56, Association for Computing Machinery, 2018.

[20] X. Lin, M. Zhang, Y. Liu, and S. Ma, "Enhancing personalized recommendation by implicit preference communities modeling," *ACM Trans. Inf. Syst.*, vol. 37, Nov. 2019.

[21] Y. Pan, F. He, H. Yu, and H. Li, "Learning adaptive trust strength with user roles of truster and trustee for trust-aware recommender systems," *Applied Intelligence*, pp. 314–327, 07 2019.

[22] G. Beigi, J. Tang, and H. Liu, "Social science–guided feature engineering: A novel approach to signed link analysis," *ACM Trans. Intell. Syst. Technol.*, vol. 11, Jan. 2020.

[23] P. Cui, X. Wang, J. Pei, and W. Zhu, "A survey on network embedding," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 5, pp. 833–852, 2019.

[24] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, (New York, NY, USA), p. 701–710, Association for Computing Machinery, 2014.

[25] Y. Dong, N. V. Chawla, and A. Swami, "Metapath2vec: Scalable representation learning for heterogeneous networks," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, (New York, NY, USA), p. 135–144, Association for Computing Machinery, 2017.

[26] T.-y. Fu, W.-C. Lee, and Z. Lei, "Hin2vec: Explore meta-paths in heterogeneous information networks for representation learning," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17, (New York, NY, USA), p. 1797–1806, Association for Computing Machinery, 2017.

[27] H. Wang, N. Wang, and D.-Y. Yeung, "Collaborative deep learning for recommender systems," in *Collaborative Deep Learning for Recommender Systems*,

KDD '15, (New York, NY, USA), p. 1235–1244, Association for Computing Machinery, 2015.

[28] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, (Republic and Canton of Geneva, CHE), p. 1067–1077, International World Wide Web Conferences Steering Committee, 2015.

[29] J. Tang, M. Qu, and Q. Mei, "PTE: predictive text embedding through large-scale heterogeneous text networks," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015*, pp. 1165–1174, ACM, 2015.

[30] M. Xie, H. Yin, H. Wang, F. Xu, W. Chen, and S. Wang, "Learning graph-based poi embedding for location-based recommendation," in *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, (New York, NY, USA), p. 15–24, Association for Computing Machinery, 2016.

[31] C. Shi, B. Hu, W. X. Zhao, and P. S. Yu, "Heterogeneous information network embedding for recommendation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 2, pp. 357–370, 2019.

[32] G. Christoforidis, P. Kefalas, A. Papadopoulos, and Y. Manolopoulos, "Recommendation of points-of-interest using graph embeddings," in *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*, pp. 31–40, 2018.

[33] H. Yin, W. Wang, H. Wang, L. Chen, and X. Zhou, "Spatial-aware hierarchical collaborative deep learning for poi recommendation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, pp. 2537–2551, Nov 2017.

[34] H. Wang, F. Zhang, M. Hou, X. Xie, M. Guo, and Q. Liu, "Shine: Signed heterogeneous information network embedding for sentiment link prediction," in *Proceedings of the Eleventh ACM International Conference on Web Search*

and Data Mining, WSDM '18, (New York, NY, USA), p. 592–600, Association for Computing Machinery, 2018.

[35] Y. Lu, C. Shi, L. Hu, and Z. Liu, "Relation structure-aware heterogeneous information network embedding," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 4456–4463, 07 2019.

[36] B. Li, D. Pi, Y. Lin, I. A. Khan, and L. Cui, "Multi-source information fusion based heterogeneous network embedding," *Information Sciences*, vol. 534, pp. 53 – 71, 2020.

[37] S. N. Mighan, M. Kahani, and F. Pourgholamali, "Poi recommendation based on heterogeneous graph embedding," in *2019 9th International Conference on Computer and Knowledge Engineering (ICCKE)*, pp. 188–193, 2019.

[38] W. Zhao, H. Ma, Z. Li, X. Ao, and N. Li, "Improving social and behavior recommendations via network embedding," *Information Sciences*, vol. 516, pp. 125 – 141, 2020.

[39] T. Lyu, F. Sun, P. Jiang, W. Ou, and Y. Zhang, "Compositional network embedding for link prediction," in *Proceedings of the 13th ACM Conference on Recommender Systems*, RecSys '19, (New York, NY, USA), p. 388–392, Association for Computing Machinery, 2019.

[40] D. Lian, C. Zhao, X. Xie, G. Sun, E. Chen, and Y. Rui, "Geomf: Joint geographical modeling and matrix factorization for point-of-interest recommendation," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, (New York, NY, USA), p. 831–840, Association for Computing Machinery, 2014.

[41] E. Cho, S. 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. ACM, San Diego, pp 1082–1090*, 2011.

[42] V. Zheng, Y. Zheng, X. Xie, and Q. Yang, "Collaborative location and activity recommendations with gps history data," in *Proceedings of the 19th Inter-*

national Conference on World Wide Web, WWW '10, (New York, NY, USA), p. 1029–1038, Association for Computing Machinery, 2010.

[43] Y. Zheng, X. Xie, and W. Ma, "Geolife: a collaborative social networking service among user. location and trajectory," *IEEE Data Eng Bull 33(2):32–40*, 2010.

[44] C. Cheng, H. Yang, I. King, and M. R. Lyu, "A unified point-of-interest recommendation framework in location-based social networks," *ACM Trans. Intell. Syst. Technol.*, vol. 8, Sept. 2016.

[45] X. Li, D. Han, J. He, L. Liao, and M. Wang, "Next and next new poi recommendation via latent behavior pattern inference," *ACM Trans. Inf. Syst.*, vol. 37, Sept. 2019.

[46] R. Hussein, D. Yang, and P. Cudré-Mauroux, "Are meta-paths necessary? revisiting heterogeneous graph embeddings," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18, (New York, NY, USA), p. 437–446, Association for Computing Machinery, 2018.

[47] S. Zhang, L. Yao, A. Sun, and Y. Tay, "Deep learning based recommender system: A survey and new perspectives," *ACM Comput. Surv.*, vol. 52, Feb. 2019.

[48] Y. Pan, F. He, and H. Yu, "A correlative denoising autoencoder to model social influence for top-n recommender system," *Frontiers of Computer Science*, vol. 14, p. 143301, Dec 2019.

[49] H. Bagci and P. Karagoz, "Context-aware location recommendation by using a random walk-based approach," *Knowledge and Information Systems*, 2015.

[50] K. Leung, D. Lee, and W. 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. ACM, Beijing*, pp. 305–314, 01 2011.

[51] W. Wang, H. Yin, X. Du, Q. V. H. Nguyen, and X. Zhou, "Tpm: A temporal personalized model for spatial item recommendation," *ACM Trans. Intell. Syst. Technol.*, vol. 9, Nov. 2018.

[52] B. Yang, Y. Lei, J. Liu, and W. Li, "Social collaborative filtering by trust," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 8, pp. 1633–1647, 2017.

[53] F. Wang, J. She, Y. Ohyama, W. Jiang, G. Min, G. Wang, and M. Wu, "Maximizing positive influence in competitive social networks: A trust-based solution," *Information Sciences*, vol. 546, pp. 559 – 572, 2020.

[54] R. Mayer, J. Davis, and F. Schoorman, "An integrative model of organizational trust," *Academy of management review, vol. 20, no. 3, pp. 709–734*, 1995.

[55] G. Guo, J. Zhang, D. Thalmann, and N. Yorke-Smith, "Etaf: An extended trust antecedents framework for trust prediction," *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, 2014.

[56] Z. Gong, H. Wang, W. Guo, Z. Gong, and G. Wei, "Measuring trust in social networks based on linear uncertainty theory," *Information Sciences*, vol. 508, pp. 154 – 172, 2020.

[57] W. Zhao, H. Ma, Z. Li, X. Ao, and N. Li, "Improving social and behavior recommendations via network embedding," *Information Sciences*, vol. 516, pp. 125 – 141, 2020.

[58] H. Cai, V. W. Zheng, and K. C.-C. Chang, "A comprehensive survey of graph embedding: Problems, techniques, and applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 9, pp. 1616–1637, 2018.

[59] H. Polat and W. Du, "Svd-based collaborative filtering with privacy," in *Proceedings of the 2005 ACM symposium on Applied computing*, pp. 791–795, 2005.

[60] O. N. Osmanli and İ. H. Toroslu, "Using tag similarity in svd-based recommendation systems," in *2011 5th International Conference on Application of Information and Communication Technologies (AICT)*, pp. 1–4, IEEE, 2011.

[61] A. Ahmed, N. Shervashidze, S. Narayanamurthy, V. Josifovski, and A. J. Smola, "Distributed large-scale natural graph factorization," in *Proceedings of the 22nd International Conference on World Wide Web*, WWW '13, (New York, NY, USA), p. 37–48, Association for Computing Machinery, 2013.

[62] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, (Red Hook, NY, USA), p. 3111–3119, Curran Associates Inc., 2013.

[63] A. Grover and J. Leskovec, "Node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, (New York, NY, USA), p. 855–864, Association for Computing Machinery, 2016.

[64] S. Cao, W. Lu, and Q. Xu, "Grarep: Learning graph representations with global structural information," in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, (New York, NY, USA), p. 891–900, Association for Computing Machinery, 2015.

[65] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, (Red Hook, NY, USA), p. 1025–1035, Curran Associates Inc., 2017.

[66] C. Zhang, D. Song, C. Huang, A. Swami, and N. V. Chawla, "Heterogeneous graph neural network," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '19, (New York, NY, USA), p. 793–803, Association for Computing Machinery, 2019.

[67] X. Fu, J. Zhang, Z. Meng, and I. King, "MAGNN: metapath aggregated graph neural network for heterogeneous graph embedding," in *WWW '20: The Web Conference 2020, Taipei, Taiwan, April 20-24, 2020* (Y. Huang, I. King, T. Liu, and M. van Steen, eds.), pp. 2331–2341, ACM / IW3C2, 2020.

[68] S. Singh and S. Bawa, "A privacy, trust and policy based authorization framework for services in distributed environments," *International Journal of Computer Science*, vol. 2, 01 2007.

[69] B. Solhaug, D. Elgesem, and K. Stolen, "Why trust is not proportional to risk,"

in *The Second International Conference on Availability, Reliability and Security (ARES'07)*, pp. 11–18, 2007.

[70] S. P. Marsh, "Formalising trust as a computational concept," tech. rep., Ph.D. Dissertation. University of Stirling, 1994.

[71] S. Chakrabarti, B. Dom, P. Raghavan, S. Rajagopalan, D. Gibson, and J. Kleinberg, "Automatic resource compilation by analyzing hyperlink structure and associated text," *Computer Networks and ISDN Systems*, vol. 30, no. 1, pp. 65 – 74, 1998. Proceedings of the Seventh International World Wide Web Conference.

[72] J. Bao, Y. Zheng, and M. Mokbel, "Location-based and preference-aware recommendation using sparse geo-social networking data," in *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '12, (New York, NY, USA), pp. 199–208, ACM, 2012.

[73] A. Noulas, S. Scellato, N. Lathia, and C. Mascolo, "A random walk around the city: new venue recommendation in location-based social networks. in: Privacy, security, risk and trust (passat)," *international conference on and 2012 international conference on social computing (SocialCom). IEEE, Amsterdam, pp 144–153*, 2012.

[74] M. Ester, H. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, pp. 226–231, AAAI Press, 1996.

[75] C. Petersen, J. G. Simonsen, and C. Lioma, "Power law distributions in information retrieval," *ACM Trans. Inf. Syst.*, vol. 34, Feb. 2016.

[76] D. Yang, B. Qu, J. Yang, and P. Cudre-Mauroux, "Revisiting user mobility and social relationships in lbsns: A hypergraph embedding approach," in *The World Wide Web Conference*, WWW '19, (New York, NY, USA), pp. 2147–2157, ACM, 2019.

[77] Y. Liu, W. Wei, A. Sun, and C. Miao, "Exploiting geographical neighborhood characteristics for location recommendation," in *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, CIKM '14, (New York, NY, USA), pp. 739–748, ACM, 2014.

[78] M. Mehaffy, S. Porta, Y. Rofè, and N. Salingaros, "Urban nuclei and the geometry of streets: The 'emergent neighborhoods' model," *URBAN DESIGN International*, vol. 15, pp. 22–46, Mar 2010.

[79] Q. Tang and J. Wang, "Privacy-preserving friendship-based recommender systems," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 5, pp. 784–796, 2018.

[80] Y. Dan-Gur and S. Rafaeli, "'friends group' in recommender systems: Effects of user involvement in the formation of recommending groups," in *ICIS*, 2006.

[81] S. Xu, J. Cao, P. Legg, B. Liu, and S. Li, "Venue2vec: An efficient embedding model for fine-grained user location prediction in geo-social networks," *IEEE Systems Journal*, vol. 14, no. 2, pp. 1740–1751, 2020.

[82] M. D. Smucker, J. Allan, and B. Carterette, "A comparison of statistical significance tests for information retrieval evaluation," in *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, CIKM '07, (New York, NY, USA), p. 623–632, Association for Computing Machinery, 2007.

[83] J. Urbano, M. Marrero, and D. Martín, "A comparison of the optimality of statistical significance tests for information retrieval evaluation," in *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '13, (New York, NY, USA), p. 925–928, Association for Computing Machinery, 2013.

[84] J. Urbano, H. Lima, and A. Hanjalic, "Statistical significance testing in information retrieval: An empirical analysis of type i, type ii and type iii errors," in *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR'19, (New York, NY, USA), p. 505–514, Association for Computing Machinery, 2019.

APPENDIX A

DATA SOURCES FOR CHARTS

Table A.1: Distribution of user count according to number of friends for Figure 4.1

Brightkite		Four Square		Gowalla		Wee Places	
Number of Friends	User Count	Number of Friends	User Count	Number of Friends	User Count	Number of Friends	User Count
0	495	0	3276	0	1528	0	452
1	534	1	1397	1	1242	1	720
2	361	2	535	2	647	2	547
3	214	3	281	3	406	3	395
4	123	4	163	4	269	4	269
5	89	5	92	5	147	5	247
6	76	6	79	6	138	6	181
7	67	7	54	7	122	7	159
8	50	8	50	8	92	8	128
9	43	9	33	9	57	9	100
10	28	10	22	10	73	10	86
11	26	11	24	11	45	11	70
12	25	12	17	12	35	12	70
13	11	13	17	13	21	13	60
14	14	14	7	14	23	14	57
15	7	15	15	15	17	15	42
16	14	16	9	16	16	16	27
17	14	17	11	17	19	17	30
18	12	18	7	18	16	18	21
19	5	19	1	19	12	19	24
20	7	20	3	20	14	20	23
21	8	21	3	21	11	21	23

Table A.1 Continued from previous page

Brightkite		Four Square		Gowalla		Wee Places	
Number of Friends	User Count	Number of Friends	User Count	Number of Friends	User Count	Number of Friends	User Count
22	5	22	1	22	4	22	15
23	3	23	2	23	10	23	18
24	4	24	5	24	5	24	21
25	4	25	2	25	10	25	12
26	2	26	4	26	4	26	10
27	4	27	1	27	7	27	8
28	4	29	2	28	9	28	12
29	2	30	2	29	4	29	7
30	4	31	2	30	6	30	8
31	2	32	3	31	5	31	8
33	1	34	1	32	7	32	2
34	1	36	2	33	6	33	7
35	2	37	2	34	3	34	12
36	2	39	1	35	2	35	8
37	1	40	1	36	1	36	6
38	2	44	1	37	1	37	6
40	1	50	2	38	1	38	6
47	1	66	1	39	3	39	4
48	1	176	1	40	4	40	3
53	1			41	2	41	2
56	1			43	2	42	4
74	1			44	2	43	2
76	1			45	2	44	4
104	1			47	1	45	3
122	1			48	2	46	4
133	1			49	1	47	2
				50	5	48	1
				51	2	49	1
				53	3	50	6
				55	1	51	3
				56	1	52	5
				59	1	54	2
				61	2	55	1

Table A.1 Continued from previous page

Brightkite		Four Square		Gowalla		Wee Places	
Number of Friends	User Count	Number of Friends	User Count	Number of Friends	User Count	Number of Friends	User Count
				63	2	56	1
				64	1	57	1
				66	1	59	3
				68	1	60	1
				69	1	62	4
				80	1	63	2
				81	1	64	2
				83	1	65	2
				88	1	68	1
				89	2	69	1
				90	2	72	1
				95	1	77	1
				99	1	80	1
				101	1	81	1
				108	1	87	1
				110	2	88	1
				121	1	93	1
				130	1	95	1
				136	1	97	1
				143	1	100	1
				144	1	103	1
				160	1	106	1
				320	1	107	1
						111	1
						113	1
						132	1
						140	1
						169	1
						174	1
						219	1
						266	1

Table A.2: f-measure values for varying number of friend groups for Figure 4.2

Dataset	Group	R3	R5	R8	R10	R15	R20
Brightkite	[0-0]	22.27	17.04	13.10	10.20	8.07	5.84
	[1-1]	24.49	18.86	14.41	12.71	8.79	6.82
	[2-3]	28.92	21.21	15.78	13.54	9.87	7.73
	[4-6]	30.67	22.74	16.79	14.14	9.90	7.86
	[7-10]	32.31	24.26	17.82	15.13	10.82	8.31
	[11-350]	31.11	22.90	17.76	14.98	10.84	8.57
Four Square	[0-0]	10.89	8.20	6.65	5.50	4.74	3.72
	[1-1]	11.13	9.84	7.87	6.63	5.63	4.05
	[2-3]	11.96	9.13	8.27	7.07	6.05	4.10
	[4-6]	12.96	10.26	8.82	7.53	6.49	4.52
	[7-10]	13.51	11.35	9.04	7.68	6.88	4.76
	[11-350]	12.64	10.05	8.70	7.47	6.41	4.45
Gowalla	[0-0]	11.43	9.23	6.90	5.93	4.40	3.53
	[1-1]	13.07	10.45	7.77	6.70	5.06	4.08
	[2-3]	16.51	13.03	9.99	8.67	6.59	5.38
	[4-6]	18.43	14.60	11.31	9.90	7.48	6.03
	[7-10]	18.58	14.87	11.62	10.13	7.76	6.38
	[11-350]	17.05	13.51	10.40	9.01	6.98	5.73
Wee Places	[0-0]	7.86	6.08	4.97	4.59	3.98	3.59
	[1-1]	8.75	7.29	6.21	5.73	5.02	4.53
	[2-3]	10.31	9.06	7.73	7.11	6.18	5.60
	[4-6]	11.24	9.70	8.34	7.76	6.73	5.99
	[7-10]	12.76	10.72	9.50	8.73	7.39	6.53
	[11-350]	16.50	13.91	11.66	10.70	8.97	7.75

Table A.3: Performance values of top n% expert vs trusted user for Figure 4.3

		Trustable User from Top					
Dataset	Expert Users from Top	0%	2%	3%	5%	7%	10%
Brightkite	0%	37.79	40.68	40.85	41.63	42.33	42.03
	2%	38.88	41.17	41.68	42.25	43.23	43.32
	3%	38.95	41.30	42.78	43.11	43.30	43.47
	5%	39.08	41.62	43.19	43.87	43.56	43.46
	7%	39.62	41.79	42.54	43.45	43.82	43.44
	10%	39.36	41.61	41.77	42.38	43.58	43.16
Four Square	0%	23.36	24.64	25.06	25.17	25.30	25.17
	2%	24.41	24.87	25.15	25.50	25.50	25.61
	3%	24.32	25.09	25.82	25.75	25.73	25.82
	5%	24.32	25.49	25.76	26.12	26.06	25.77
	7%	24.52	25.67	25.77	26.06	25.82	25.69
	10%	24.75	25.52	25.63	25.71	25.64	25.64
Gowalla	0%	26.32	27.95	28.50	28.80	29.00	29.16
	2%	28.00	28.82	29.11	29.18	29.32	29.40
	3%	28.40	29.14	29.38	29.51	29.44	29.53
	5%	28.55	29.26	29.57	29.64	29.60	29.61
	7%	28.60	29.44	29.54	29.61	29.60	29.57
	10%	28.70	29.55	29.62	29.61	29.63	29.58
Wee Places	0%	12.02	12.58	12.81	12.82	12.92	12.98
	2%	12.19	13.22	13.28	13.44	13.45	13.41
	3%	12.33	13.41	13.57	13.57	13.58	13.53
	5%	12.54	13.50	13.59	13.68	13.63	13.58
	7%	12.62	13.47	13.57	13.62	13.62	13.57
	10%	12.78	13.45	13.54	13.57	13.58	13.57

Table A.4: Performance values of Brightkite dataset for Figure 4.4

Metric	Rec Count	CDL	TECF	TrustMF	TrustSVD	CLR	TLoRW
precision	3	23.44444	25.21111	23.78	24.33	25.93333	29.14815
precision	5	17.82667	21.17333	19.63	20.36	20.56	21.48889
precision	8	13.39167	15.08611	15.08	15.68	15.85	16.73611
precision	10	11.64667	13.16889	13.16	13.76	13.56	14.47778
precision	15	8.84	10.02222	10.84	11.03	10.21333	10.94815
precision	20	7.226667	8.107778	8.49	8.51	8.026667	8.166667
recall	3	31.01117	33.78837	30.88	31.34	33.56261	38.94112
recall	5	35.70093	40.83321	34.04	34.58	42.18291	43.87477
recall	8	40.61157	45.86978	36.81	37.21	49.09932	49.92055
recall	10	43.32837	49.33645	39.16	39.94	51.67857	52.52931
recall	15	48.34711	53.5	42.24	42.82	55.85805	58.95833
recall	20	52.14045	56.9835	45.76	46.36	57.91246	60.65842
f-measure	3	26.7021	28.87627	26.86888	27.39365	29.25881	33.34039
f-measure	5	23.77946	27.88658	24.90051	25.62973	27.64553	28.84845
f-measure	8	20.14163	22.70483	21.39506	22.06288	23.96405	25.06804
f-measure	10	18.35855	20.78882	19.69976	20.46832	21.48304	22.69932
f-measure	15	14.94702	16.88193	17.25251	17.54149	17.26911	18.46709
f-measure	20	12.69395	14.19574	14.32267	14.3803	14.09919	14.39525

Table A.5: Perfomance values of Four Square dataset for Figure 4.5

Metric	Rec Count	CDL	TECF	TrustMF	TrustSVD	CLR	TLoRW
precision	3	13.84788	14.21332	12.965	13.22	14.16985	16.5392
precision	5	11.6504	11.93308	11.55	12.43	11.9336	12.85277
precision	8	9.115286	9.749522	10.55	10.98	9.800291	10.39245
precision	10	8.03889	8.929063	10.2	10.26	8.352986	9.589866
precision	15	6.31375	7.354685	8.345325	8.366	7.036191	7.979924
precision	20	5.281641	6.394742	7.435984	7.58	5.689918	7.206979
recall	3	19.92941	20.26094	22.40355	23.23	19.46209	22.29094
recall	5	23.14163	24.28	24.45	25.92	24.67	26.12007
recall	8	26.55077	28.36	26.05464	26.63	30.45881	31.3279
recall	10	28.3826	33.42	27.85	28.12	33.02223	34.85273
recall	15	32.23497	37.9	31.15	31.82	35.9234	40.9131
recall	20	35.45712	41.35	33.54	33.96	38.19799	44.25595
f-measure	3	16.34116	16.70668	16.42488	16.85051	16.39958	18.98908
f-measure	5	15.49834	16.00169	15.68875	16.80238	16.08596	17.22817
f-measure	8	13.57133	14.51062	15.01867	15.54892	14.8292	15.60742
f-measure	10	12.52912	14.09284	14.93141	15.03446	13.33331	15.04111
f-measure	15	10.55929	12.31884	13.16393	13.2487	11.76752	13.35501
f-measure	20	9.193789	11.07651	12.17313	12.39368	9.904479	12.3954

Table A.6: Perfomance values of Gowalla dataset for Figure 4.6

Metric	Rec Count	CDL	TECF	TrustMF	TrustSVD	CLR	TLoRW
precision	3	14.83981	16.03981	15.83323	16.80398	15.17804	17.40249
precision	5	11.91343	12.41343	13.05762	13.67938	11.9325	14.03669
precision	8	9.403889	9.920389	10.48634	10.77574	9.569516	11.05787
precision	10	8.34336	8.78336	9.161543	9.340924	8.110785	9.644529
precision	15	6.563463	7.03463	7.549821	7.610858	6.940572	7.782311
precision	20	5.518342	6.051834	6.09112	6.170652	5.67058	6.357053
recall	3	20.17086	22.18	23.95545	23.43968	22.70831	23.93852
recall	5	24.56284	24.25	29.37029	29.16288	25.72572	29.67638
recall	8	29.09762	27.78	33.69121	34.5716	28.70431	35.53715
recall	10	31.50488	31.37	36.09272	37.11392	31.4027	38.04201
recall	15	35.89216	37.78	40.72884	41.039	37.95437	44.04197
recall	20	39.48709	44.28	43.84947	43.41048	40.57644	47.55565
f-measure	3	17.09946	18.61668	19.06533	19.57476	18.19482	20.15383
f-measure	5	16.04482	16.42103	18.07801	18.62321	16.30306	19.05874
f-measure	8	14.21403	14.61992	15.99443	16.43028	14.35374	16.86727
f-measure	10	13.19288	13.72408	14.61365	14.92539	12.89183	15.38788
f-measure	15	11.09756	11.86078	12.73836	12.84041	11.73518	13.22732
f-measure	20	9.68342	10.64834	10.6964	10.80536	9.950563	11.21494

Table A.7: Perfomance values of Wee Places dataset for Figure 4.7

Metric	Rec Count	CDL	TECF	TrustMF	TrustSVD	CLR	TLoRW
precision	3	8.547156	8.593953	8.310664	8.533503	8.414844	8.620467
precision	5	7.652833	7.728532	7.743823	7.886553	7.605611	8.046115
precision	8	6.93129	7.097506	6.76053	6.77079	6.810264	7.124168
precision	10	6.559325	6.530575	6.212545	6.320996	6.515223	6.504916
precision	15	5.836634	5.741031	5.832295	5.854642	5.769388	5.664447
precision	20	5.139521	5.183682	5.186971	5.20987	5.260548	5.077704
recall	3	10.10451	10.36486	10.53375	11.01532	10.85912	11.49425
recall	5	11.24039	11.50989	11.45792	12.20905	11.42945	12.67551
recall	8	12.49079	12.8529	13.42846	13.84971	12.74987	14.33472
recall	10	13.07183	14.65202	14.69015	14.79098	13.56752	15.0942
recall	15	14.08544	16.09044	15.79333	16.59662	14.54308	17.5949
recall	20	15.05779	17.61588	17.40425	17.7271	16.14729	19.91789
f-measure	3	9.260814	9.396698	9.291077	9.616872	9.481994	9.852072
f-measure	5	9.105998	9.247594	9.241671	9.582923	9.133455	9.843688
f-measure	8	8.915346	9.145031	8.993368	9.095171	8.878261	9.518009
f-measure	10	8.735337	9.034409	8.732197	8.856937	8.803122	9.091715
f-measure	15	8.253314	8.46262	8.518723	8.655841	8.261396	8.569921
f-measure	20	7.66338	8.010252	7.992073	8.053016	7.935747	8.092398

Table A.8: Effect of changing window values for Figure 4.8, Figure 4.9, Figure 4.10 and Figure 4.11

Dataset	Precision	Recall	f-measure	Window
Brightkite	2.78	2.98	2.88	1
	26.65	28.64	27.61	2
	21.09	22.67	21.85	3
	22.40	24.08	23.21	4
	18.28	19.65	18.94	5
	19.38	20.82	20.08	6
	16.63	17.87	17.23	7
	17.75	19.08	18.39	8
Four Square	0.30	0.24	0.27	1
	13.61	11.04	12.19	2
	8.88	7.20	7.95	3
	9.89	8.02	8.86	4
	7.04	5.71	6.30	5
	7.92	6.42	7.09	6
	6.16	5.00	5.52	7
	7.03	5.70	6.29	8
Gowalla	0.50	0.51	0.51	1
	10.51	10.85	10.68	2
	8.73	9.02	8.87	3
	8.74	9.03	8.88	4
	7.45	7.70	7.57	5
	7.39	7.64	7.51	6
	6.59	6.81	6.70	7
	6.57	6.78	6.68	8
Wee Places	0.93	0.48	0.63	1
	11.15	5.69	7.53	2
	8.00	4.04	5.37	3
	7.70	3.93	5.20	4
	5.21	2.66	3.52	5
	4.65	2.37	3.14	6
	3.45	1.76	2.33	7
	3.04	1.55	2.05	8

Table A.9: Perfomance values of Brightkite dataset for Figure 4.14

Metric	Rec Count	DeepWalk	PTE	metapath2Vec	HIN2Vec	JUST	JLGE	MAGNN	SgWalk
precision	3	18.20	24.18	16.20	17.41	27.51	21.42	22.44	31.15
	5	13.67	19.42	11.37	12.86	20.69	16.60	16.29	22.22
	8	9.83	15.18	8.85	10.53	15.09	13.03	13.26	15.54
	10	8.25	13.13	8.13	8.83	12.70	11.76	10.85	12.90
	15	5.91	9.43	5.43	7.23	9.10	8.40	8.89	9.02
	20	4.53	7.15	4.11	5.42	7.03	6.58	6.84	6.92
recall	3	11.74	17.17	17.59	10.67	17.74	13.82	13.46	20.09
	5	14.69	19.69	20.58	13.14	22.23	17.84	16.50	23.88
	8	16.90	23.42	25.62	17.22	25.96	22.40	21.02	26.73
	10	17.74	24.72	29.42	18.04	27.30	25.29	21.98	27.73
	15	19.05	25.54	29.49	22.16	29.34	27.10	28.56	29.07
	20	19.48	25.93	29.78	22.16	30.21	28.27	28.07	29.75
f-measure	3	14.27	20.08	16.87	13.23	21.57	16.80	16.83	24.42
	5	14.16	19.55	14.65	13.00	21.43	17.20	16.40	23.02
	8	12.43	18.42	13.16	13.07	19.08	16.47	16.26	19.65
	10	11.26	17.15	12.74	11.86	17.33	16.06	14.53	17.61
	15	9.02	13.78	9.17	10.90	13.89	12.83	13.55	13.77
	20	7.35	11.21	7.22	8.71	11.41	10.67	11.00	11.23

Table A.10: Perfomance values of Four Square dataset for Figure 4.15

Metric	Rec Count	DeepWalk	PTE	metapath2Vec	HIN2Vec	JUST	JLGE	MAGNN	SgWalk
precision	3	5.27	10.07	8.09	6.41	5.88	10.65	8.52	12.24
	5	4.52	8.33	6.97	5.54	4.76	8.13	7.23	9.28
	8	3.54	5.95	5.93	4.77	3.54	6.04	6.13	6.68
	10	3.10	5.36	5.29	4.23	3.06	5.19	5.48	5.61
	15	2.24	3.97	4.09	2.94	2.24	3.87	4.30	3.94
	20	1.73	3.06	3.23	2.24	1.77	3.06	3.35	3.02
recall	3	2.56	4.53	3.64	2.97	2.86	4.66	3.83	5.96
	5	3.66	6.25	5.22	4.28	3.86	5.93	5.42	7.52
	8	4.59	7.13	7.12	5.89	4.60	7.06	7.35	8.66
	10	5.03	8.04	7.93	6.54	4.96	7.57	8.20	9.10
	15	5.46	8.92	9.21	6.82	5.45	8.46	8.96	9.60
	20	5.61	9.17	9.69	6.92	5.75	8.94	9.75	9.81
f-measure	3	3.45	6.25	5.02	4.06	3.85	6.49	5.28	8.01
	5	4.05	7.14	5.97	4.83	4.26	6.86	6.19	8.31
	8	4.00	6.49	6.47	5.27	4.00	6.51	6.69	7.54
	10	3.84	6.43	6.35	5.14	3.78	6.16	6.57	6.94
	15	3.18	5.49	5.66	4.11	3.18	5.31	5.81	5.59
	20	2.64	4.58	4.84	3.38	2.71	4.56	4.98	4.62

Table A.11: Perfomance values of Gowalla dataset for Figure 4.16

Metric	Rec Count	DeepWalk	PTE	metapath2Vec	HIN2Vec	JUST	JLGE	MAGNN	SgWalk
precision	3	9.67	12.71	8.34	11.79	10.56	12.19	10.75	14.62
	5	7.36	9.21	7.20	8.72	7.95	9.05	9.12	10.67
	8	5.37	7.02	6.98	6.68	6.00	6.88	7.29	7.52
	10	4.47	6.09	5.89	5.91	5.10	5.52	6.24	6.26
	15	3.13	4.35	4.55	4.29	3.89	3.98	4.39	4.41
	20	2.38	3.44	3.57	3.40	3.06	3.29	3.50	3.38
recall	3	5.99	7.22	4.74	6.69	6.54	6.92	5.98	9.06
	5	7.60	8.71	6.81	8.25	8.22	8.56	8.56	11.02
	8	8.86	10.62	10.57	10.10	9.90	10.41	12.90	12.42
	10	9.22	11.53	11.15	11.18	10.54	10.44	12.59	12.92
	15	9.70	12.34	12.92	12.18	12.04	11.31	12.95	13.65
	20	9.83	13.02	13.49	12.88	12.63	12.46	13.70	13.97
f-measure	3	7.40	9.21	6.04	8.54	8.08	8.82	7.68	11.19
	5	7.48	8.95	7.00	8.48	8.08	8.80	8.83	10.84
	8	6.68	8.45	8.41	8.04	7.47	8.28	9.31	9.37
	10	6.02	7.97	7.71	7.73	6.88	7.22	8.35	8.43
	15	4.73	6.43	6.73	6.35	5.88	5.89	6.56	6.67
	20	3.83	5.44	5.64	5.38	4.92	5.21	5.58	5.44

Table A.12: Perfomance values of Wee Places dataset for Figure 4.17

Metric	Rec Count	DeepWalk	PTE	metapath2Vec	HIN2Vec	JUST	JLGE	MAGNN	SgWalk
precision	3	4.51	10.32	7.87	9.74	5.92	12.60	8.29	12.71
	5	3.64	7.81	6.63	7.27	5.33	9.38	6.88	10.15
	8	2.86	6.47	5.49	6.14	4.43	8.15	5.68	8.35
	10	2.49	5.71	5.27	5.45	4.40	7.02	5.46	7.19
	15	1.96	4.65	4.50	4.62	3.53	5.35	4.74	5.43
	20	1.61	4.05	3.85	3.95	3.12	4.33	3.99	4.34
recall	3	1.38	2.79	2.13	2.63	1.81	3.86	2.24	4.50
	5	1.86	3.52	2.99	3.28	2.72	4.78	3.10	5.69
	8	2.33	4.66	3.96	4.43	3.62	6.65	4.09	6.82
	10	2.54	5.15	4.75	4.91	4.49	7.17	4.92	7.34
	15	2.99	6.29	6.09	6.25	5.40	8.19	6.41	8.31
	20	3.29	7.30	6.95	7.13	6.36	8.83	7.20	8.86
f-measure	3	2.11	4.39	3.35	4.15	2.78	5.91	3.52	6.65
	5	2.46	4.85	4.12	4.52	3.60	6.34	4.27	7.29
	8	2.57	5.42	4.60	5.14	3.98	7.33	4.75	7.51
	10	2.52	5.41	5.00	5.17	4.44	7.10	5.18	7.26
	15	2.37	5.35	5.18	5.31	4.27	6.47	5.45	6.57
	20	2.16	5.21	4.96	5.09	4.19	5.81	5.14	5.83