LATENT REPRESENTATION MODELS FOR USER SEQUENTIAL MOBILITY AND SOCIAL INFLUENCE PROPAGATION

SHANSHAN FENG

Interdisciplinary Graduate School

2017
LATENT REPRESENTATION MODELS FOR USER SEQUENTIAL MOBILITY AND SOCIAL INFLUENCE PROPAGATION

SHANSHAN FENG

Interdisciplinary Graduate School

A thesis submitted to the Nanyang Technological University in partial fulfilment of the requirement for the degree of Doctor of Philosophy

2017
Abstract

With the increasing popularity of online social media applications, a large amount of data has been generated by users. Based on the user generated data, many research problems have been studied, such as the location-based recommendation and social influence analysis. In this thesis, we investigate the problem of user sequential mobility and the problem of social influence propagation. The main challenge of both problems lies in the difficulty to effectively learn the sequential transition. However, due to the data sparsity, it is hard to model the sequential information by conventional methods, e.g., Markov chain model. To this end, we resort to the latent representation approach, which is to represent items in a low-dimensional latent space, such that the relations between items are captured by their representations. In addition, based on the social influence propagation in social networks, we study the problem of finding a set of influential users.

First, we examine user mobility in location-based social networks (LBSNs), where users share their locations by checking-in points-of-interest (POI). Based on the users’ check-in sequences, we consider two research tasks. The first one is next new POI recommendation problem in which new POIs relating to users’ current location are to be recommended. The second one is to predict users who will visit a given POI in a given future period. The complexity of these problems is to precisely learn POI sequential transition, user preference, and the geographical influence. To solve this difficulty, we develop two latent presentation models. One is personalized ranking metric embedding method (PRME), which embeds POIs in a latent Euclidean space. The other is POI2Vec model, which directly incorporates the geographical influence and further jointly considers the user preference. Experimental studies demonstrate the superiority of our proposed approach over the state-of-the-art algorithms in both next new POI recommendation and future visitor prediction.

Second, we develop a new latent representation model Inf2vec to learn representations of users in a social network, such that the social influence is captured. As a fundamental problem in social influence propagation analysis, learning influence parameters has been investigated. Most of the existing methods are proposed to estimate
the propagation probability for each edge in social networks. However, they cannot
exactly learn the propagation parameters of edges due to data sparsity. Different from
the conventional methods, we introduce a social influence embedding problem, which
is to learn parameters for nodes rather than edges. We develop a new model Inf2vec,
which combines the local social influence context and global user similarity to learn
the representations. We conduct extensive experiments on two real datasets, and the
results indicate that Inf2vec significantly outperforms the baseline algorithms.

Lastly, based on the social influence propagation, we investigate the problem of
influence maximization with novelty decay (IMND). Influence maximization problem
is to find a set of seed nodes in a social network such that their influence spread is
maximized under certain propagation models. A few algorithms have been proposed
for this problem. However, they have not considered the impact of novelty decay
in influence propagation, i.e., repeated exposures will have diminishing influence on
users. We investigate the effect of novelty decay on influence propagation in real-life
datasets and formulate the IMND problem. We further analyze its relevant properties
and propose an efficient influence estimation technique to solve it. Empirical results
on four social networks show the effectiveness and efficiency of our algorithms.
Acknowledgments

I would love to express my sincere gratitude towards my supervisor Prof. Gao Cong. Under his elaborative supervision, I have not only learned the cutting edge techniques, but also got a conscientious research attitude through critical thinking. I also appreciate my co-supervisor Prof. Yeow Meng Chee and mentor Prof. Yu Liu for their kind help.

I would like to acknowledge the Ph.D. grant from Advanced Environmental Biotechnology Centre, Nanyang Environment and Water Research Institute, Interdisciplinary Graduate School, Nanyang Technological University, Singapore.

I would also like to appreciate my group members for their friendly help. They include Xin Cao, Quan Yuan, Lisi Chen, Tuan Anh Nguyen, Kaiyu Feng, Tao Guo, Kaiqi Zhao, Zhida Chen, Yiding Liu, Xiucheng Li, Chaodong Zhang.

Last but not least, my deepest love and gratitude are devoted to all of my family members. Their loves will be endless motivation for me to move forward and help me overcome difficulty in my life.
2.3 Latent Representation Models ........................................ 32
  2.3.1 Metric Embedding .............................................. 33
  2.3.2 Word2vec ...................................................... 33

3 Personalized Ranking Metric Embedding for User Sequential Mobility 36
  3.1 Next New POI Recommendation .................................. 36
    3.1.1 Observations on real-world datasets ......................... 37
    3.1.2 Next new POI problem definition ............................ 38
  3.2 Personalized Ranking Metric Embedding ......................... 39
    3.2.1 Pairwise Ranking Metric Embedding ......................... 40
    3.2.2 Personalized Ranking Metric Embedding ..................... 42
    3.2.3 Incorporating Geographical Influence ....................... 43
  3.3 Parameter Learning ............................................. 44
    3.3.1 Optimization Criterion ..................................... 44
    3.3.2 Learning Algorithm ........................................ 45
  3.4 Experiments ...................................................... 45
    3.4.1 Experimental Setup ........................................ 45
    3.4.2 Performance of Methods .................................... 47
    3.4.3 Effect of Parameters ...................................... 49
  3.5 Conclusion ...................................................... 51

4 POI2Vec Representation Model for User Sequential Mobility 53
  4.1 Future Visitor Prediction Problem .............................. 53
  4.2 POI2Vec Representation Model .................................. 54
    4.2.1 POI2Vec Model for Sequential Transition .................. 55
    4.2.2 Extend POI2Vec Model for User Preference .................. 62
    4.2.3 Predict Future Visitors .................................... 63
  4.3 Experiments ...................................................... 64
    4.3.1 Experimental Setup ........................................ 64
    4.3.2 Latent Representation for Sequential Transition .......... 65
    4.3.3 Future Visitor Prediction .................................. 67
    4.3.4 Effects of Parameters .................................... 69
  4.4 Conclusion ...................................................... 69
List of Figures

1.1 The overview of thesis. ............................................. 2
1.2 Users’ mobility on location based social networks. .............. 3
1.3 Social network and influence propagation sequences. .......... 5
1.4 The research problems studied in this thesis. .................... 15

2.1 The word2vec architectures. .................................... 34

3.1 The average ratio of new POIs with number of days .............. 37
3.2 CDF of time difference between two consecutive check-ins. .... 38
3.3 CDF of geographical distance between two consecutive check-ins. 39
3.4 The result of methods on Foursquare. ............................. 48
3.5 The result of methods on Gowalla. ................................. 49
3.6 Different localized region constraint on two datasets. .......... 49
3.7 The effect of time threshold $\tau$ on Foursquare and Gowalla. .... 50
3.8 Effect of component weight $\alpha$. ................................. 51
3.9 Effect of the number of dimension for PRME. ................. 51

4.1 The POI frequency on Foursquare and Gowalla. ................. 56
4.2 Conventional binary tree for hierarchical softmax .............. 57
4.3 The illustration of bulding binary tree over POIs ............... 59
4.4 The binary tree generated by POI2Vec model .................... 60
4.5 The experimental results for sequential transition on Foursquare. 66
4.6 The experimental results for sequential transition on Gowalla. ... 66
4.7 The experimental results of predicting future visitors on Foursquare. 67
4.8 The experimental results of predicting future visitors on Gowalla. 67
4.9 Effect of region size threshold $\theta$. ............................. 69
4.10 Effect of number of dimensions $D$. ............................. 70

5.1 An example of extracting influence propagation networks. ....... 73
5.2 Social influence learning on social networks. ................. 75
5.3 Distributions of source user frequency on Digg and Flickr. . . . . . . . 77
5.4 Distributions of target user frequency on Digg and Flickr. . . . . . . . 78
5.5 The cumulative distribution function (CDF) of count of active friends before taking actions. . . . . . . . . . . . . . . . . . . . . . . . . . . . . 78
5.6 Effect of number of dimension $K$ on Digg and Flickr. . . . . . . . 93
5.7 Effect of context size $L$ on Digg and Flickr. . . . . . . . . . . . . . . 93
6.1 Novelty Decay on Digg. . . . . . . . . . . . . . . . . . . . . . . . . . . 98
6.2 Novelty Decay on Flickr. . . . . . . . . . . . . . . . . . . . . . . . . . 99
6.3 A social network with influence probabilities and delays on the social links. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 99
6.4 Performance of RGA and RGA-DP based algorithms. . . . . . . . . . 111
6.5 The influence spread of methods. . . . . . . . . . . . . . . . . . . . . 112
6.6 The running time of methods. . . . . . . . . . . . . . . . . . . . . . . 113
6.7 The results of influence spread for different $\gamma$. . . . . . . . . . . 113
List of Tables

3.1 Statistics of two location-based datasets ........................................... 37
4.1 Statistics of Foursquare and Gowalla datasets ..................................... 64
5.1 Statistics of Digg and Flickr for social influence embedding .................. 77
5.2 The performance with different aggregation functions on Digg dataset .... 89
5.3 The performance with different aggregation functions on Flickr dataset .... 89
5.4 The performance for various methods on Digg dataset ........................... 89
5.5 The performance for various methods on Flickr dataset ........................ 90
5.6 The running time of one iteration for various methods on Digg and Flickr dataset .............................................................. 92
6.1 Notation table ..................................................................................... 96
6.2 Statistics of datasets for influence maximization with novelty decay ....... 108
Chapter 1

Introduction

With the increasing popularity of online social medias, a huge amount of data has been generated by users. Based on the user generated data, many research problems have been studied. One important problem is how to discover useful and valuable knowledge from large amounts of data, which is widely known as data mining. Among a variety of data mining tasks on social medias, we are interested in exploring the sequential information, which has attracted extensive research attentions recently. The sequential information is very common in many applications, such as sequential purchase records [1], GPS-trajectories [2], and sentences in natural languages [3]. In this thesis, we mainly focus on two applications: user sequential mobility on location-based social networks and social influence propagation on social networks, which will be introduced in detail later.

From a technical perspective, numerous algorithms have been proposed to model the sequential information. Recently, latent representation models have been widely utilized, and also have been proven effective in various tasks. Latent representation models exploit low dimensional vectors to represent items, and the relationships among items can be captured by these vectors. In this thesis, in order to learn the sequential information, we investigate two latent representation models: metric embedding [4] and word2vec technique [5].

We present a brief outline of this thesis in Figure 1.1. We concentrate our efforts on investigating the sequential information (i.e., $a \rightarrow b \rightarrow c \rightarrow ...$) on social medias, which indicates different sequence transitions on two different application domains: user
Chapter 1. Introduction

Sequential information
\[ a \rightarrow b \rightarrow c \rightarrow \ldots \]
Latent representation models

Applications

User sequential mobility
Social influence propagation

Solutions

Figure 1.1: The overview of thesis.

sequential mobility and social influence propagation. For the solutions, we develop several novel latent representation methods to model the sequential transition in the two application domains.

Next, we first introduce the background and motivation of modeling user sequential mobility and social influence propagation with latent representation methods in Section 1.1. Following that, we present research problems and methodologies in Section 1.2. We summarize the main contributions of our studies in Section 1.3 and state the thesis organization in Section 1.4.

1.1 Background and Motivation

1.1.1 User Mobility

Recent years have witnessed the rapid growth of location-based social networks (LBSNs), such as Gowalla and Foursquare. Users check-in points-of-interest (POIs) and share their locations on the LBSNs. For example, there are more than 10 billion check-ins on Foursquare by September 2016.\(^1\) A POI is one particular place that may be interesting and helpful for some users, e.g., hotel and museum. Figure 1.2(a) shows some POIs in Singapore. Typically, each POI is associated with geographical coordinates: longitude and latitude. Therefore, the large amount of check-in data offers an opportunity for us to understand user mobility behavior.

\(^1\)http://venturebeat.com/2016/09/13/foursquare-users-have-checked-in-over-10-billion-times/
Chapter 1. Introduction

(a) POIs on map

(b) Users’ check-in sequences

Figure 1.2: Users’ mobility on location based social networks.

To investigate the users’ mobility behaviors, many research studies have been conducted. One representative task is POI recommendation, which is to suggest a few POIs for an individual to check-in. It is of great value to help people explore their surrounding locations. The recommendation is far more interesting when the POIs are well in line with users’ personal preference. POI recommendation problem has drawn a great deal of research interest [6–10]. In existing methods, several important factors have been considered to model the user mobility pattern, such as user preference and geographical influence. The user personal interest has been proven effective in general recommendation systems [11]. It also works well for POI recommendation. In contrast to general recommendation systems, POI recommendation problem additionally exploits geographical influence, which acts as a special characteristic of user mobility. Recent studies [6,7,9] demonstrate that users’ check-ins exhibit strong geographical correlation. Intuitively, users incline to visit near locations rather than far away locations. Therefore, the geographical influence is considered in almost all the POI recommendation models. Apart from user preference and geographical influence, other features also have been considered to enhance the POI recommendation
performance, such as temporal information \[9,12,13\], social links \[8,14,15\] and textual information \[16–18\].

Compared to the POI recommendation, the next POI recommendation \[19\] gets much less research attention. Given a user and his/her present location, the next POI recommendation task is to suggest some POIs that this user would like to visit in the next few hours (e.g., 6 hours). The next POI recommendation additionally considers the sequential information of users’ check-ins. The sequential influence is important for POI recommendation because human movement exhibits sequential patterns \[20\]. For instance, a user would visit a cinema after having dinner at a restaurant, and then go home. Intuitively, user’s next location is related to his current location. Figure 1.2(b) shows three users’ check-in sequences. The challenge of next POI recommendation is to learn POI sequential transitions from check-in sequences. The sequential transitions of users’ check-ins are commonly represented by a Markov chain model. However, due to the sparse transition data, it is difficult to exactly estimate the transition probability in Markov chain, especially for unobserved transitions. The problem becomes more challenging if we want to further model user’s personal interest in POI transitions. Therefore, it is necessary to propose effective approaches to capture the personal check-in sequences.

Another interesting problem is user prediction, which is considered from another perspective. Instead of suggesting POIs for users, user prediction task aims to recommend potential users for a POI. The user prediction problem is of great value since it can help the POI holders to find their target customers who will visit the POIs in future. However, most of the existing user mobility studies focus on suggesting POIs for users. The problem of predicting users for a location is rarely studied \[17, 21\]. Nevertheless, previous methods mainly utilize the user’s interest and do not consider the POIs’ sequential information. By considering the POI sequential transition, it is interesting to identify the users who will visit a given POI in next few hours. For example, given a restaurant and the current time, we want to predict potential consumers who would visit this restaurant in the next several hours. Similar to next POI
recommendation problem, the challenge lies in the difficulty to precisely learn POI sequential transition, user preference, and geographical influence.

### 1.1.2 Social Influence Propagation

Recently, online social networks, e.g., Flickr and Weibo, have become more and more popular among people. Social networks are widely used as platforms for spreading of news, messages, and opinions. Generally, a social network can be modeled as a directed or undirected graph $G = (V, E)$, where $V$ is the set of nodes and $E$ is the set of edges. Each node $v$ in graph $G$ represents a user. Figure 1.3(a) presents an example of social network. Each circle denotes a user and each arrow means a directed link. The users are connected by links. A user may watch his/her friends’ online behaviors and further be influenced by friends to perform the same action. Therefore, edge $(u, v)$ indicates that $u$ would affect $v$, in other words, $(u, v)$ is one information diffusion channel from $u$ to $v$. The information would spread in social networks by these links.

![Social network](image1.png)

![Influence propagation sequences](image2.png)

**Figure 1.3:** Social network and influence propagation sequences.

Online social networks play an important role in the diffusion of messages and ideas among users. Information propagation in social networks has obtained extensive
research attention in recent years. To understand the process of influence propagation, a large number of diffusion models have been proposed. Among them, two influence spread models are commonly used: Linear Threshold model [22] and Independent Cascade model [23], which will be introduced in detail in Chapter 2. Based on these influence diffusion models, various applications have been investigated, such as viral marketing [24].

Briefly, viral marketing is a marketing method that exploits the effect of “word-of-mouth”. At first, some influential persons are selected as seed users. Then it utilizes the influence spread of these seed users to promote products to more people [25, 26]. For instance, a camera company that wants to market its new camera may provide incentives to a number of customers by giving them discounts or free samples. These initial customers then post messages on social networks, such as Facebook, where a lot of their friends will see these messages and may adopt this new camera.

One key computational problem of viral marketing is influence maximization [25], which is formulated by Kempe et al. [25]. Influence maximization problem and its extensions have received extensive research interest recently [27–32]. Given a social network, the diffusion parameters (e.g., influence probabilities of edges) and a budget $K$, it finds $K$ users to maximize the expected size of affected users. The task of choosing influential users is modeled as a discrete optimization problem. Meanwhile, this problem is NP-hard [25]. Numerous algorithms have been developed to efficiently solve the influence maximization problem.

In the studies of influence maximization, the diffusion parameters are often given as input. In practice, these parameters are unknown and we need to estimate them. Compared to influence maximization problems, the question of how to compute diffusion parameters has received little research interest.

As a fundamental problem in social influence analysis, influence propagation learning is to infer diffusion parameters for a given social network. In order to infer diffusion parameters, historical social influence propagation information is utilized. Given an action log that contains a set of tuples $(user, action, time)$, we can obtain the influence diffusion data. Each item is corresponding to one diffusion episode, which is a
set of users in chronological order. Based on the real diffusion episode data, we can extract the influence propagation sequences, which records how the social influence spread in a social network. Figure 1.3(b) shows an example of influence propagation. For instance, one propagation sequence is $1 \rightarrow 2 \rightarrow 3$, which indicates that user 1 is activated first (e.g., adopting one item), then user 2 and user 3 are activated in succession. To study the influence propagation, user’s action is assumed to be relative to social influence [33,34]. In this sequence, user 1 may influence user 2, and then user 2 may further influence user 3. The challenge of influence propagation learning is how to effectively model these propagation sequences.

In recent years, a few approaches have been developed to learn propagation parameters. Most of the existing methods directly learn the diffusion probability for each edge [33,35,36]. One common method is maximum likelihood estimation (MLE) model [33]. It directly uses the co-occurrence counting approach to estimate the propagation probabilities. Another widely used method is utilizing Expectation Maximization (EM) technique to learn the diffusion probabilities [34,35,37]. However, due to the sparsity of propagation data, these approaches cannot effectively learn the propagation parameters. Especially, it is impossible to learn the diffusion parameters for unobserved data. For instance, if no social influence has been observed on a link $(u,v)$, it is hard to estimate the propagation probability $P_{uv}$. Therefore, it is necessary to develop effective algorithms to learn the sequential influence propagation in social networks.

1.1.3 Latent Representation Models

In this thesis, we try to answer one question: how to effectively model the check-in sequences and social influence propagation sequences. However, as mentioned early, it is challenging to effectively model the sequential information due to the data sparsity. To this end, we resort to the latent representation techniques to capture the sequential transitions.

Recently, latent representation models have been adopted to a variety of research tasks, such as next product recommendation [1], music playlist prediction [4,38], POI
sequence [19, 39] and network embedding [40, 41]. Instead of directly modeling the sequential transitions, latent representation models reflect the transitions in a low-dimensional latent space, where each item is modeled as a point or a vector. The relations between items are captured by the similarity of their representations.

A variety of models have been proposed to learn the latent representations, such as principal component analysis [42] dimensionality reduction [43] and singular value decomposition [44]. In recent years, two representation models have received extensive research attention, which are able to effectively capture the sequential transitions. One is metric embedding [4, 37, 45], which represents each item with one or multiple points in a latent space. The transition probability of two items is reflected by the Euclidean distance of their representations. For example, Chen et al. [4] propose a metric embedding approach for music playlist prediction. A playlist is a sequence of songs and can be treated as Markov chain sequence. A logistic Markov embedding model is developed, which learns representations of songs in Euclidean space. Inspired by the idea of metric embedding, we develop a new metric embedding method to model user check-in sequences.

The other one is word2vec technique [3, 5], which is a neural network language model for natural language processing. One advantage of word2vec technique is that it is able to capture the semantic relationships among words. Word2vec techniques represent each item with one vector. The inner product of two vectors reflects the sequential transition relationship between two items. Although word2vec techniques are proposed for natural language processing, they have been adopted by various applications, such as user mobility [39] and network embedding [40, 46, 47]. Motivated by the recent progress of word2vec techniques, we develop novel representation models to model the check-in sequences and social influence propagation sequences respectively.

1.2 Research Problems and Methodologies

In this thesis, we mainly develop novel latent representation methods to model users’ check-in sequences and social influence propagation. Moreover, based on the influence
propagation on social networks, we also investigate the task of mining influential users. Overall, we study 4 research problems as follows.

- **Next new POI recommendation**: where to go next?
- **Future visitors prediction**: who will come next?
- **Social influence embedding**: who will be influenced next?
- **Influence maximization with novelty decay**: who are influential users?

Next, we will present the research problems and develop approaches to address them.

### 1.2.1 Next New POI Recommendation

The POI recommendation task is to recommend a few POIs for an individual to visit. However, it does not consider the sequential influence, which means that user’s mobility is influenced by his recent locations. Compared to the conventional POI recommendation problem, the next POI recommendation problem [19] has received much less research attention, in which POIs with respect to users’ current position are to be suggested. We verify users’ sequential behavior in the analysis of two real-world datasets.

Meanwhile, we observe that users often visit new POIs that they have not visited before. In this thesis, we only consider recommend new POIs for users, since we aim at helping individuals better explore their surroundings. Note that, this task is different from the next POI prediction [20], in which both visited and un-visited POIs taken as results. For the next POI prediction task, we find that a high prediction precision can be obtained if we simply utilize frequently visited POIs of a user to predict his next POIs. (A similar finding is reported in [48]). What’s more, its challenging to estimate the likelihood of visiting novel POIs given the historical check-in data. Thus, the next new POI recommendation problem is harder than the next POI prediction problem.
Therefore, we investigate the Next New POI recommendation problem ($N^2$-POI recommendation), which is to suggest new POIs to be visited next given a user’s current location. Since recommendation system usually suggest new items to users, there is no much difference between “next new POI recommendation” and “next POI recommendation”. We just highlight the idea that only new POIs are recommended to distinguish it from “next POI prediction”. In this thesis, “next new POI recommendation” equals to “next POI recommendation”, unless otherwise sated.

The challenge of $N^2$-POI recommendation is to learn transitions of users’ check-ins that are commonly represented by a first-order Markov chain model. Due to the sparse transition data, it is difficult to estimate the transition probability in Markov chain, especially for the unobserved transitions. Factorized Personalized Markov Chain (FPMC)\cite{1} method has been proposed to calculate the item transitions. FPMC exploits matrix factorization technique to factorize the Markov transition matrix. To model the transition, FPMC represents each item with two independent vectors. However, these two vectors are related to the same item and their latent relationship is not exploited. Consequently, this technique is not sufficiently effective to learn the item transitions.

By projecting every POI into one object in a low-dimensional Euclidean latent space, we use the Metric Embedding algorithm to effectively compute the location transition in a Markov chain model. Intuitively, the distance between two objects measures the strength of their sequential relation. We further use a pair-wise ranking Metric Embedding algorithm that ranks potential POIs in a latent space. Subsequently, we propose a Personalized Ranking Metric Embedding model (PRME), which learns the sequential information and individual preference. Since users incline to visit the POIs close to their current positions, geographical influence is important for the recommendation task. We extend the PRME model to accommodate geographical influence in the $N^2$-POI recommendation.
1.2.2 Future Visitor Prediction

Relevant to the next POI recommendation task, future visitor prediction is to predict the users who will visit a given POI in the next few hours. A user’s mobility is highly influenced by his recent visits and his personal interest [19]. However, the users’ check-in data are very sparse, and it is challenging to model the sequential influence of POIs and user preference. This is also the main challenge for predicting future visitors for a given POI.

The progress in neural network language models demonstrates that the word2vec technique [3, 5] is able to effectively capture the sequential semantic relationships among words. Very recently, the word2vec algorithm has been utilized to model users’ sequential check-ins [39]. However, the method in [39] has two limitations. First, it fails to incorporate the geographical influence of POIs, i.e., users usually incline to visit nearby POIs, in the word2vec model. The geographical influence is very important in modeling user movement behaviors as shown in almost all the previous POI recommendation works. Second, it separately models the sequential transition and user preference. Since each check-in behavior is related to both the sequential transition and user preference, it is more reasonable to jointly model these two factors.

To incorporate geographical influence, we develop a POI2Vec latent representation model. In our model, each POI is represented as a vector in a latent low dimensional space, and the inner product of two vectors reflects the relevance between two POIs. To learn the latent vectors, we exploit the hierarchical softmax (HS) technique [49], which is widely used in neural network models by exploiting a binary tree structure. For the hierarchical softmax, the key to achieve good performance is constructing a proper binary tree over items [50]. Instead of using Huffman tree in word2vec, we develop a novel binary tree building method, which is able to incorporate the geographical coordinates of POIs. We hierarchically split the POIs into different regions, and then build a binary tree over the POIs in each region. Since a POI may also influence POIs in adjacent regions, we assign a POI to multiple nearby
regions. In the generated binary tree, one POI may occur more than once, which is able to capture the various relationships with other POIs.

Furthermore, we model the user preference with the proposed POI2Vec model, where each user is also represented with a latent vector. We propose a model to jointly learn the representations of POIs and users by considering both the user preference and sequential transition. To predict potential visitors for a POI, we consider both the users with recent locations and the users whose recent positions are unknown. For users with recent locations, we aggregate the user preference and sequential transition for prediction. For the other users, we only make use of the user preference.

1.2.3 Social Influence Embedding

As a fundamental research problem in social influence analysis, the task of learning diffusion parameters has been investigated in the last years. Given an action log $\mathcal{A}$ and a social graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, it aims to infer the influence probabilities between users. Most of the existing methods are in discrete manners [33, 35, 36]. These methods directly learn diffusion probability for each individual edge. However, due to the sparsity of propagation observations, these methods cannot effectively estimate the diffusion parameters for the $|\mathcal{E}|$ edges.

Different from conventional methods, Bourigault et al. [37] propose an embedded cascade model, which is to learn the embeddings for the $|\mathcal{V}|$ nodes. The propagation probability between two nodes is captured by the Euclidean distance between their embeddings. But this approach has several limitations. First, it does not explicitly utilize the network structure, which is very important for the social influence analysis. Second, the proposed algorithm is not efficient and cannot solve the large-scale network. It employs an Expectation Maximization (EM) technique as [35], which is time consuming. Third, the proposed approach is designed for Independent Cascade model and fails to incorporate other important factors for user’s online behavior, such as user preference.

Inspired by the network embedding [40, 46, 47], we introduce a new research problem: social influence embedding. The problem is to represent each user into a latent
low-dimensional space, such that the propagation relationship between two users is captured by the similarity of their representations. The key of this problem is to effectively model the influence propagation sequences. To this end, we develop a new representation model \textbf{Inf2vec} to learn social influence embedding.

In the Inf2vec model, we first utilize a randomized algorithm to generate social influence context for each user and then employ the word2vec technique \cite{5} to learn the representation of each user. To generate the social influence context, we consider two factors. One is local influence propagation neighborhood. For a given user, we utilize a random-walk process to produce a sequence of users. These users act as local influence propagation context for the given user. The other factor is the similarity of user preference. A user’s behavior not only can be influenced by his friends but also affected by his individual preference. The users’ action can reflect their personalized interest. Intuitively, the users with similar interest are more likely to perform the same action. To incorporate the effect of user similarity, for a given user, we randomly sample several users who perform the same action as the global user similarity context. In this way, the social influence context consists of the local influence propagation context and global user similarity context.

1.2.4 Influence Maximization with Novelty Decay

As an important application in social networks, influence maximization problem \cite{25} has attracted significant attention. It selects $K$ nodes in order to maximize the propagation of ideas, opinions, etc. in social networks.

It has been observed that repeated exposures of an individual to an idea may have diminishing influence on the individual \cite{51, 52}. For example, on Twitter, a user is more likely to retweet a tweet message for the first time that the user reads the message than the subsequent exposures to the message. The chance that a user retweets a message usually diminishes with the number of repeated exposures to the message. We call the phenomenon \textit{novelty decay}. Intuitively, people are less likely to become spreaders of repeated information. To the best of our knowledge, the novelty
decay phenomenon has not been considered by the existing study on the influence maximization problem.

In this thesis, we analyze the effect of novelty decline in social influence propagation based on two real datasets. For a further examination, we develop a fitting function to characterize the effect. By doing this, we are able to integrate the novelty decay factor into an influence propagation model such as independent cascade model, which is commonly used in the literature of influence maximization [30, 31, 53].

Differing from conventional influence propagation models, the new influence function becomes neither monotone nor submodular. This renders inapplicable the greedy algorithm with the CELF optimization [27] that is adopted by nearly all the existing influence maximization algorithms. We then improve the adapted version of U-Greedy algorithm [54] by pruning low-influential nodes in a dynamic way. As the influence propagates differently with the delay time between every pair of nodes, computing influence spread becomes complex. We develop a propagation path based algorithm to estimate the influence spread of seed nodes.

1.2.5 Overall Picture

The framework of this thesis is summarized in Figure 1.4. In this thesis, we examine four research problems in two fields: user sequential mobility (illustrated by orange boxes) and social influence propagation (illustrated by green boxes). The black arrows indicate the features that are utilized in each research task. We can learn that the sequential information are investigated in all the 4 studied research problems. To better model the user sequential mobility, we additionally consider the user preference and geographical influence. In the social influence embedding problem, similarity of user preference is taken into consideration. Furthermore, in order to find influential users, novelty decay factor is studied to solve a novel influence maximization problem.

As shown by the red arrows in Figure 1.4, several latent representation models are proposed to address the research problems. We develop PRME and POI2Vec latent representation models for user sequential mobility and Inf2vec latent representation model for social influence propagation. The PRME is based on metric embedding,
POI2Vec and Inf2vec are developed from word2vec technique. However, since the latent representation model is not feasible for influence maximization with novelty decay, we develop a graph-based algorithm.

![Diagram of research problems studied in this thesis.](image)

**Figure 1.4:** The research problems studied in this thesis.

To better understand the research problems and proposed solutions, we briefly discuss the relationships between them as follows.

(i) Although user sequential mobility and social influence propagation are distinct applications, they have some connections. First, both user mobility and influence propagation are generated by users on social networks. By exploring them, we can understand more about users’ online behaviors. Second, the challenges of these two applications are similar. For user sequential mobility, we need to learn the sequential transition from one POI to another POI for a given user. While in social influence propagation, we aim to model the information flow from one user to another user for a given item. In addition, since individual preference is an important factor in users’ behaviors, we consider user preference in these...
two applications respectively. Lastly, both user sequential mobility and social influence propagation can be effectively solved by latent representation models. Particularly, we develop techniques to address them: POI2Vec and PRME for user sequential mobility and Inf2vec for social influence propagation.

(ii) Note that both PRME and POI2vec are latent representation models for user sequential mobility. To solve the next new POI recommendation problem, we develop the PRME representation model. To address the future visitor prediction problem, we propose a POI2Vec model. Since next new POI recommendation and future visitor prediction are highly relevant, both PRME and POI2Vec can be used to solve the next POI recommendation and future visitor prediction. The difference between PRME and POI2Vec lies in the way of representation. First, PRME embeds each item as a point in the latent Euclidean space, while POI2Vec embeds each item as a vector in the latent space. Second, PRME utilizes the Euclidean distance between two points to reflect the relation of two items, while POI2Vec exploits the inner product of two vectors. Third, they use different approaches to learn representations. PRME employs pair-wise ranking to derive the objective function and makes use of Bayesian Personalized Ranking approach [55]. Meanwhile, as a shallow neural network, POI2Vec exploits softmax function to derive the objective function and utilizes hierarchical softmax to learn the representations. Last but not least, PRME utilizes the geographical influence as a post-filter condition, and POI2Vec explicitly incorporates the geographical influence into the representation model. In general, the performance of POI2Vec is better than PRME.

(iii) In social influence propagation, social influence embedding and influence maximization with novelty decay are two different research problems. One is an unsupervised learning problem and another is a discrete optimization problem. But these two problems are closely connected. First, both of them are related to the social influence propagation on social networks. By exploring these two problems, we can learn more about the social influence propagation, which is a
very complex phenomenon. Second, social influence embedding learns the propagation parameters, and provides an alternative way to obtain the prerequisites for the influence maximization problem, which needs propagation parameters as input. Third, due to the fact that novelty decay effect exists in social influence propagation, the novelty decay studied in IMND can be supplementary for social influence embedding. Since the influence maximization problem is highly related to social influence propagation, we include the problem of influence maximization with novelty decay in this thesis.

(iv) Both POI2Vec and Inf2vec employ the word2vec techniques. However, there are several differences between them. First, the way of generating context is different. To model user sequential transition, POI2Vec directly utilizes the observations, i.e., use the check-ins that occur before current check-in as context. Inf2vec exploits a randomized strategy to generate the context. Second, POI2Vec utilizes the hierarchical softmax to learn parameters, while Inf2vec exploits the negative sampling. For approximating the softmax, computing the complexity of hierarchical softmax is $O(\log|V|)$, where $|V|$ is the size of all the items. The complexity of negative sampling is $O(N)$, where $N$ is the number of sampled negative instances (e.g., 5 or 10). Generally, negative sampling is more efficient than hierarchical softmax when $N < \log(|V|)$. Lastly, they utilize dissimilar ways to combine the additional information. We build a region-based binary tree for POI2Vec. In this way, POI2Vec is able to explicitly incorporate geographical influence in the learning procedure. Meanwhile, Inf2vec incorporates the information (e.g., the similarity of user preference) in the way of generating context.

1.3 Summary of Contributions

In this thesis, we develop various latent representation models for user sequential mobility and social influence propagation. Moreover, we also propose approaches for the IMND problem. The main contributions of this thesis are stated as follows.
• First, we propose a pair-wise Metric Embedding algorithm to model the sequential transition of POIs. As far as we know, this is the first work that exploits the Metric Embedding method for the next POI recommendation problem. To model the personalized sequential information, we propose a novel PRME algorithm, which jointly considers sequential transition and user preference. We further develop a PRME-G algorithm to additionally consider geographical influence for the $N^2$-POI recommendation problem. We conduct comprehensive experiments by comparing our algorithms with baseline techniques over two real-world datasets.

• Second, we develop a novel latent representation model POI2Vec, which incorporates the geographical influence of POIs, to predict the potential visitors for a location in the next few hours. This is the first work to incorporate the geographical influence in the framework of word2vec. We develop a method to jointly learn the latent representation for users and POIs. We aggregate the user preference and POI sequential influence to predict the future visitors for a POI. This model can also be used to predict the subsequent POIs for a user. We conduct extensive empirical studies to evaluate the performance of our method, which is much better than baseline methods for both next POI prediction and future user prediction.

• Third, we introduce a new social influence embedding problem, which is to learn representations of users, such that the social influence is captured. To address this problem, we design a novel representation model Inf2vec, which jointly considers the effect of local influence propagation and global user interest similarity. We conduct extensive experiments, and our experimental results show that our approach significantly outperforms the state-of-the-art algorithms.

• Lastly, we consider the effect of novelty decay in influence propagation on social networks. With the augmented independent cascade propagation model, we formulate the influence maximization problem with novelty decay. To solve
Chapter 1. Introduction

the problem, we employ the R-Greedy algorithm and improve its performance through the dynamic pruning strategy. In addition, we develop an influence propagation path based algorithm to efficiently compute influence spread of seed nodes. Advantages of our algorithms are demonstrated with extensive experiments.

1.4 Thesis Organization

In Chapter 2, we review related work for user mobility, social influence propagation, and latent embedding models. In Chapter 3, we present PRME model for recommending next new POIs. Next, we introduce the POI2Vec model for future visitors prediction in Chapter 4. Chapter 5 presents the Inf2Vec model to learn the latent embedding for social influence. Chapter 6 states the influence maximization with novelty decay in social networks. Finally, in Chapter 7, we conclude the thesis and discuss several directions of future work.
Chapter 2

Literature Review

As introduced in chapter 1, this thesis aims to explore and utilize sequential information for user movement behavior and influence diffusion in online social networks. To learn the sequential transitions, we mainly exploit latent representation models. With the purpose of being consistent with the structure of chapter 1, we will go through the related work in three parts: user mobility, social influence propagation and latent representation models.

In this chapter, we first introduce the research studies related to user mobility on location-based social networks (LBSNs), especially the sequential POI recommendation. Next, in Section 2.2, we review the literature about influence propagation on social networks. Lastly, we discuss some latent embedding models in Section 2.3.

2.1 User Mobility

With the increasing popularity of LBSNs, people would like to share their locations by checking-in POIs. The large amount of check-in data provides opportunities to better understand users’ mobility behaviors. A variety of research studies have been conducted to explore the user’ movement pattern, such as POI recommendation. In this section, we review the existing studies of user mobility.

2.1.1 POI recommendation

With the availability of users’ online mobility information, “POI recommendation” problem has attracted significant research attention, which is to suggest new POIs for
users to visit. Various approaches have been proposed to address the POI recommendation problem. We categorize them based on the information they use.

**User preference.** Similar to other recommendation tasks (e.g., product recommendation), most of the previous POI recommendation solutions are based on collaborative filtering (CF) [11, 56] or matrix factorization (MF) [57, 58] models. The CF and factorization based algorithms mainly exploit user preference to make the recommendation. Collaborative filtering exploits the historical preference data from other users to make recommendations for a given user. The basic idea of the CF approach is that similar users would have similar preference for items. One widely used CF algorithm for POI recommendation is the user-based CF [7,9], which takes advantage of check-ins of similar users for the recommendation purpose. It first computes user similarities between a user and other users. Then it suggests POIs for a user by using a mixture of the other users’ historical locations. In addition, item-based CF also has been utilized for POI recommendation [59,60].

Another widely used POI recommendation technique is factorization model [8,10, 12], which learns the general preference of a user by learning the observed user-item matrix. In the factorization method, both items and users are represented as vectors in a $K$-dimensional space. The original user-item matrix $U \times I$ is decomposed into two matrices: user matrix $U \times K$ and item matrix $I \times K$. Factorization techniques usually can achieve better recommendation accuracy than CF techniques, especially when the check-in data is sparse.

**Geographical influence** One significant characteristic in user mobility behavior is the geographical influence. Intuitively, the close POIs have strong relations than the POIs that are far away. Geographical influence has been extensively exploited to enhance the performance of location recommendation. To model the geographical information, various methods have been developed. One simple way is recommending only near POIs by ignoring POIs that are far away to user historically visited POIs [19]. Power law distribution has been applied to model the geographical influence [7,9]. These methods utilize the power law distribution of spatial distance to
calculate the inclination of a person traveling from one POI to another POI. Geographical influence also has been captured by Gaussian Mixture (GM) distribution [6, 8]. In these models, the willingness of a person checking-in a venue is determined by the multi-center Gaussian distributions. Without assuming any prior distribution, Zhang et al. [61, 62] propose kernel density estimation methods to learn a personalized check-in distribution. Lian et al. [10] propose a factorization method to jointly model the user preference and geographical influence for POI recommendation. In this joint factorization model, users’ latent vectors are augmented with users’ activity areas, while POIs’ latent vectors are augmented with POIs’ influence areas.

Generally, user preference and geographical influence are considered by nearly all the POI recommendation solutions. Apart from these two factors, to enhance the performance of POI recommendation, other features have been studied, which are briefly summarized as follows.

- **Temporal information.** The temporal features are important to model user mobility behavior, since people incline to check-in different POIs at different time. Considering temporal information, time aware POI recommendation is to suggest POIs for user at the specific time (e.g., 7pm.). Several models have been proposed to incorporate temporal information, such as CF model [9], factorization model [12, 13] and graph-based model [63].

- **Social links.** As a widely-used contextual feature in conventional recommendation problems, social influence is employed to enhance performance of POI recommendation [8, 14, 15, 62, 64–66]. The idea of using social influence is that friends are likely to share similar interest in POIs. These methods exploit friends’ preference to make recommendations for a given user.

- **Textual information.** Textual content information has been investigated for the content-aware POI recommendation models [16–18, 21]. Generally, the textual information is considered by probabilistic language models, such as latent Dirichlet allocation model [67] and probabilistic latent semantic analysis model [68].
• **Category.** Location-based social networks usually have predefined categories, such as category hierarchy of Foursquare\(^1\). Each POI is assigned to one or multiple categories. Users may have different preference for different categories. Thus the categories can be exploited to improve the POI recommendation [62, 64, 66, 69, 70].

In this thesis, we investigate the two most important features in POI recommendation: user preference and geographical influence. We do not consider the other four features (temporal information, social links, textual information and category), since they are not highly relevant to the topic of this thesis. In our models, PRME and POI2vec, we consider three factors: user preference, geographical influence and sequential transition. We can easily extend the proposed representation models to incorporate the mentioned four features. This can be left as future work.

### 2.1.2 Next-POI recommendation

However, in POI recommendation tasks, sequential transitions have not been utilized to suggest next moves for users. By considering the check-in sequential transition, we study the next POI recommendation task. Given a user and his current positions, the next POI recommendation is to suggest a few places for this user to visit in next few hours. The difference from conventional POI recommendation is that it additionally exploits sequential transitions between user check-ins to recommend subsequent POIs for users to visit. Conventional POI recommendation task mainly focuses on capturing general taste of a user and does not consider contextual locations. Therefore, the key challenge of next POI recommendation problem is how to effectively capture the sequential influence of user mobility. Next, we will briefly present the review of related research studies based on their techniques of modeling sequential influence.

Compared to the POI recommendation problem, the next POI recommendation problem has received relatively less research attention. Many approaches have been developed for next POI recommendation. Noulas et al. [71] proposes a feature based

\(^1\text{https://developer.foursquare.com/categorytree}\)
method. It first obtains a set of features by analyzing the check-in data, and then use two supervised learning models to predict next places. Meanwhile, most of the studies utilize the property of Markov chain to predict the successive check-ins.

Gambs et al. [72] incorporate the $n$ previous visited locations to predict next location by a mobility Markov chain model. Zhang et al. [73] predict the sequential probability through an additive Markov Chain. However, this method fails to assign the transition probability for the unobserved data because it directly constructs the Markov chain model based on check-in data. Instead of using transition pattern of POIs, [20, 69, 74, 75] exploit the transition pattern of POI categories to predict future check-ins. The category transition is modeled by frequency counting [74, 75], matrix factorization [69] or hidden Markov chain [20]. However, the accuracy of these methods highly depends on the category transition information.

FPMC [1] is the state-of-the-art personalized Markov chain algorithm. To model the Markov chain transition, FPMC associates each item $l$ with two independent vectors: $\vec{F}_l$ to embody the transition to other items, and $\vec{T}_l$ to represent getting transition from other items. The transition from item $l_i$ to another item $l_j$ is embodied as the inner product of the latent vectors $\vec{F}_{l_i} \cdot \vec{T}_{l_j}$. However, FPMC fails to model relations among multiple items. For example, given $l_i \rightarrow l_j$ and $l_j \rightarrow l_k$, transition $l_i \rightarrow l_k$ is expected to have a high probability because both $l_i$ and $l_k$ have close relations with $l_j$. However, FPMC fails to reflect such relationship due to the independent assumption on $\vec{F}_{l_i}$ and $\vec{T}_{l_j}$. This drawback limits its performance.

The work by Cheng et al. [19] aims to recommend POIs for the next hours by merging consecutive check-ins in the previous hours. It directly employs FPMC to model the personalized POI transition. Based on the current POI, their method only considers the POIs in the defined region as candidates. In other words, if the distance between a candidate POI and the current POI is larger than $d$ ($d=40$km in their paper), the candidate POI will be ignored. Lian et al. [48] also adopts FPMC to represent the short- and long-term preference to predict the next check-in. Differently from these FPMC-based studies, we propose new latent embedding models to learn the personalized sequential information.
Recently, other features have been considered for the next POI recommendation problem. For example, [76, 77] consider the temporal influence to predict the successive POIs: [77] proposes a ranking-based method to incorporate the geographical and temporal information, and [76] extends recurrent neural networks to consider the temporal and spatial contexts. Zhang et al. [78] utilize a weighted approximate rank pairwise method to incorporate social relations and temporal information for next-POI recommendation. He et al. [70] incorporate the category transition pattern by inferring latent behaviors. They first learn a behavior pattern distribution and then combine with pattern level preference to suggest next locations for users. Wang et al. [79] adopt the topic model to exhibit sequential patterns using textual information. They model the sequential information and personal interest at topic-region levels. Such studies are complementary to our work.

2.1.3 User Prediction

Most of the existing POI recommendation studies focus on suggesting POIs for users from the users’ perspective. In contrast, very little research has been done to predict visitors for a location. The user prediction is very useful, as the POI holders may want to identify their potential customers and further conduct marketing events to attract customers. There are several research studies investigate the task of user recommendation for a location [17, 21, 80], which is to recommend potential users for a POI. Yuan et al. [17] design a spatial-temporal topic model to learn user mobility patterns. Then they predict the potential users by computing the probability of a user checking-in a POI at the given time. Zhao et al. [21] proposes a joint model to capture the correlations of aspect, sentiment, category and spatial influence. Then they apply the model to suggest locations for a given user. Based on event-based social networks, Pham et al. [80] develop a graph-based approach, which attempts to find the users who will take part in an upcoming event. However, these user prediction methods do not predict future users in the next several hours.
2.2 Social Influence Propagation

Social influence has been widely investigated in social sciences [81]. Due to the lack of data, early studies are limited to very small datasets. The rapid expansion of online social networks, e.g., Facebook and Weibo, brings about a great deal of social influence data. Based on the data, influence propagation analysis in online social networks has received extensive research interest.

2.2.1 Influence Spread Model

Although empirical studies of social influence propagation began to appear very early, formal mathematical spread models were proposed in 1970s [82]. A variety of influence spread models have been introduced. In this thesis, we briefly review two basic influence diffusion models, which are widely used in data mining community.

2.2.1.1 Linear Threshold Model

Threshold models are developed by Granovetter et al. [22]. The most commonly used threshold model is Linear Threshold Model (LT). Many research studies are based on LT model (e.g., [25, 28, 29, 83]).

Consider a directed network $G = (V, E)$, where each edge $(u, v) \in E$ is associated with influence weight $P_{uv}$, which indicates the effect of node $u$ on $v$. $\text{InNei}(v)$ contains all neighbors who can directly influence $v$, which means $\text{InNei}(v) = \{u \in V : (u, v) \in E\}$. In LT model, a node $v$ is affected by each incoming neighbor $u$, and $\sum_{u \in \text{InNei}(v)} p_{uv} \leq 1$. Every node $v$ has an activation threshold $\theta_v$, which is often randomly chosen from the interval $[0,1]$ in order to reflect the lack of prior knowledge about the true threshold of each node.

Given some initial active nodes (denoted as seeds), the diffusion process unfolds in a sequence of steps. At step $t$, the nodes that were activated before $t$ still stay active. A node $v$ is successfully activated if the sum weight of its active incoming neighbors surpasses the threshold $\theta_v$: $\sum_{u \in \text{InNei}(v)} P_{uv} \geq \theta_v$. This process goes on until no more nodes are activated.
2.2.1.2 Independent Cascade Model

Another widely used model is independent cascade (IC) Model [23], in which each newly activated node $v$ can affect its neighbors independently. IC model has obtained extensive research interest recently [25,30,31,35,53,84–88].

Consider a directed graph $G = (V, E)$, where each edge $(u, v) \in E$ is associated with influence weight $P_{uv}$. Let $A_t$ be the users that are affected at step $t$. Starting with an initial set of seeds $A_0$, other users in that graph would be activated by these seeds. Each node $v \in A_t$ tries to activate its inactive friends, and it can only activate its friends at time $t+1$. If one neighbor $w$ is activated, $w$ will switch from inactive to active. Whether or not $v$ succeeds, $v$ loses its ability to affect its inactive friends in following rounds. If no more new activated nodes exist, the whole process ends.

Although LT and IC are different influence spread models, Kempe et al. [25] proves that they are equivalent to each other under some conditions.

2.2.2 Learning Propagation Parameters

Given a diffusion model, learning influence parameters is a fundamental problem in social influence analysis. In many tasks, e.g., influence maximization problem [25], the diffusion parameters (i.e., diffusion probabilities on edges) are often given as input. However, these parameters are unknown in reality and how to infer them is challenging. Numerous approaches have been proposed to learn influence propagation parameters, such as [36, 89–91]. In this thesis, we mainly review the models where diffusion parameters are static over time.

Goyal et al. [33] propose an efficient method to learn the propagation parameters. Given a historical action dataset, which records the actions performed by users, they aim to learn the propagation probabilities of edges. They employ the Maximum Likelihood Estimator to infer the probability $P_{uv}$: $P_{uv} = \frac{A_{uv}}{A_u}$, where $A_{uv}$ is the number of times that $u$ successfully influences $v$ and $A_u$ is the total number of trials.

Another approach is using expectation maximization (EM) algorithm to estimate the diffusion parameters. Saito et al. [35] propose a method to learn propagation parameters.
parameters for the independent cascade model. They formalize the learning task as a likelihood maximization problem, and then utilize the EM technique to infer the probability $P_{uv}$. Barbieri et al. [34] further propose a topic-aware social influence propagation model. By assuming that users may have different influence on different topics, they want to learn the topic-aware propagation probability $P_{u,v}^z$, where $z$ is a topic. Similar to [35], an EM technique is used to learn the parameters. However, the EM-based algorithms are computationally expensive and unfeasible for large dataset.

Recently, Bourigault et al. [37] propose an embedded cascade model to learn diffusion parameters in social networks. In contrast to the conventional learning approaches, it embeds users onto a $K$ dimensional continuous latent space. Each user $u$ has two latent representations: sender representation $z_u \in \mathcal{R}^K$ and receiver representation $w_u \in \mathcal{R}^K$. The transmission probability $P_{uv}$ is defined by a sigmoid function $P_{uv} = \frac{1}{1+e^{-D_{uv}}}$, where $D_{uv}$ is the Euclidean distance between the sender representation $z_u$ and the receiver representation $w_v$ in the latent space.

However, there are several limitations of this embedding model. First, it does not explicitly consider the network structure, which is very important for the social influence analysis. It assumes that if user $v_1$ performs an action before user $v_2$, then there is an edge between $v_1$ and $v_2$. This assumption is not real. Only if there is an existing edge between $v_1$ and $v_2$, then the information would spread from $v_1$ to $v_2$. Thus, we need to explicitly exploit the social graph to learn the influence propagation. Second, it employs the EM algorithm [35] to learn the latent representations, which is time consuming. Third, the proposed approach is designed for Independent Cascade model and fails to incorporate other important factors for user’s online behavior, such as user similarity. Last but not least, it has to convert the learned user embeddings to probabilities by a sigmoid function. This transformation cannot exactly reflect the propagation probability. In our social influence embedding study, we address these drawbacks of embedded cascade model.
2.2.3 Network Embedding

One related work of social influence embedding is network embedding. Generally, network embedding problem is to learn the representation of each node in a latent low-dimensional space, such that the network structure can be preserved. The latent representation of network is useful in various applications, such as node classification [92], link prediction [93] and visualization [94]. Yan et al. [95] proposes a Dimensionality Reduction model. It first constructs an affinity matrix and then extracts the leading eigenvectors of the matrix. But this method is time consuming and inefficient for large-scale networks. Another technique is graph factorization [96]. It obtains the low-dimensional embedding of a large network by matrix factorization. The embedding parameters are learned by sequential stochastic gradient descent approach. A distributed optimization is further developed with the intention of enhancing the learning efficiency.

The state-of-the-art network embedding methods [40,46,47,97] are relevant to the word2vec technique [3, 5], which is developed for learning representations of words. Word2vec is a shallow neural network language model. Empirical study has shown that word2vec is able to effectively capture the semantic relationships among words. Recently, the word2vec technique has been adopted for a variety of applications, e.g., user mobility [39] and product recommendation [98].

Based on the word2vec technique, several solutions have been proposed to solve the network embedding task. Perozzi et al. [40] report that the node frequency of social networks follows a power-law distribution, which is similar to the word frequency in natural language. Therefore, the techniques from language models can be applied to social network analysis. They propose a Deepwalk model, which first generates context with random walks and then update the representations with SkipGram [3]. A hierarchical softmax method [49] is utilized to solve it. Tang et al. [46] design a LINE algorithm, which is able to preserve both the local and global network structure by using the first-order and second-order proximity. Instead of hierarchical softmax, LINE utilizes negative sampling technique to approximate the softmax. Grover et
al. [47] develop a node2vec model for network embedding. To incorporate homophily and structural equivalence, node2vec defines a diverse notion of a node’s network neighborhood. A biased random walk procedure is designed in order to explore flexible neighborhood. Yang et al. [97] develop a semi-supervised learning method for network embedding. They jointly exploit the class label and network structure to learn the embedding.

Different from the shallow models [40, 46, 47], wang et al. [41] adopt a deep architecture [99] to capture the non-linear network structure. They utilize multiple layers of non-linear functions in a deep learning model to reflect the non-linear network structure. Although the deep learning architecture is effective, it is computationally expensive compared to the word2vec technique.

2.2.4 Influence-Maximization

Based on the social influence propagation, mining influential users is of great importance, such as viral marketing [24]. The problem of finding influential users was introduced by Domingos et al. [26, 100], and then the influence maximization problem was formulated by Kempe et al. in 2003 [25]. Influence maximization task is to identify a fixed number of influential individuals in a social network such that the users affected by them is maximized under certain propagation models. As an important application for viral marketing, this problem has attracted a large amount of research interest.

Given a graph $G = (V, E)$ associated with edge probabilities $P_{uv}$, positive integer $K$. Influence maximization problem aims to choose a seed set $S \subseteq V$, such that $\sigma(S)$ is maximized: $S = \arg \max_{S \subseteq V, |S|=K} \sigma(S)$. $\sigma(S)$ is the expected number of nodes affected by $S$. Kempe et al. [25] demonstrate that this problem is NP-hard, which means that it is extremely computationally expensive to exactly solve it.

The influence spread $\sigma(S)$ has two important properties. First, $\sigma(S)$ is monotone, which means that it increases with the size of seed set: $\sigma(S \cup \{s_i\}) > \sigma(S), s_i \in V$. Second, $\sigma(S)$ is monotone. This property indicates the effect of diminishing marginal
return: \( \sigma(S_1 \cup \{s_i\}) > \sigma(S_2 \cup \{s_i\}), s_i \in V, \text{ and } S_1 \subseteq S_2 \). Based on these properties, a natural greedy algorithm obtains \((1 - 1/e) \approx 63\%\) low bound [25].

There are two aspects to address this problem. The first one is how to compute the influence spread \(\sigma(S)\). The second one is how to efficiently select \(K\) users.

The computing of the influence spread \(\sigma(S)\) is \#P-hard [85], which means that no exact and efficient algorithm exists. Since the computation of accurate influence spread is difficult, alternative ways should be considered. Actually, the baseline method for Influence Maximization Problem is MontoCarlo (MC) simulation [25]. MC simulation method often runs many times (e.g., 20000) to evaluate \(\sigma(S)\). However, this simulation based algorithm is extremely time consuming and not suitable for large social networks.

Several heuristic algorithms are proposed to solve this problem. Chen et al. [30] proposes a Degree Discount method. It utilizes the degree as the evaluation metric of node’s influence. At each step, Degree Discount selects the user with the greatest degree and updates the degrees of the selected user’s friends. [85] designs a tree-based method. It computes \(\sigma(S)\) by employing a maximum influence tree structure, which is the combination of the maximum influence paths. For each node \(v\) in graph \(G(V, E)\), it maintains an arborescence structure, which needs a large amount of memory. Wang et al. [101] handle this problem by utilizing community structure of social networks. It divides the whole social network into several communities. Then it uses a dynamic programming technique to choose influential individuals from these communities. In [86], a data-based approach is designed. It introduces a new way to evaluate the expected influence spread.

Propagation path based method is also proposed to calculate the \(\sigma(S)\) in [53,83,87]. Firstly, it finds out all possible paths from seeds to target node \(v\), and it assumes that these paths are independent. Secondly, it utilizes all these paths to compute influence spread. Inspired by the idea of using propagation path, we develop an algorithm to estimate the influence spread with the effect of novelty decay. Differing from existing methods, our algorithm only considers top-\(C\) shortest paths and further incorporates the novelty decay.
Another aspect of influence maximization problem is how to choose $K$ seeds from all the $v \in \mathcal{V}$. Most of the existing research works (for example, [25, 29–31, 53, 85]) exploit the greedy strategy. Firstly, it chooses the node that has the maximal influence score, $S_1 = \arg \max_{v \in \mathcal{V}} (\sigma(v))$. Subsequently, it selects the node with the largest marginal score, $S_2 = \arg \max_{v \in \mathcal{V}} (\sigma(v, S_1) - \sigma(S_1))$. It repeats this procedure until all $K$ nodes are selected. In [27] a CELF framework is proposed, which is much faster than the original Greedy Algorithm. [102] further designs a method to improve CELF. However, CELF is still a greedy algorithm and it exploits submodular property to reduce the computation. Besides, Jiang et al. [84] propose a Simulated Annealing (SA) method and apply several heuristic approaches to accelerate the learning process of SA.

Based on the conventional influence maximization problem, a great deal of relevant research studies have been conducted in the last decade. [31, 53] study the time constraint influence maximization. In these models, the propagation from one node to another node would take different time. Manuel et al. [36, 103] investigate the influence maximization in continuous time propagation network. Apart from temporal information, some other features are considered for the influence maximization, such as location [88, 104], topic [105, 106] and profit [54, 107].

2.3 Latent Representation Models

Latent Representation models are unsupervised learning models, which embed items in a low-dimensional latent space. Representation learning has been applied to many domains [108], such as modeling multi-relational data [109]. Various models have been proposed to learn the latent representations, e.g., principal component analysis [42] dimensionality reduction [43] and singular value decomposition [44]. The relationships between items are captured by their similarities, which can be computed by Euclidean distance, inner product or cosine similarity. In this thesis, we mainly review two latent Representation models that relevant to our research studies, namely metric embedding and word2vec technique.
2.3.1 Metric Embedding

Metric embedding models usually embed the items in a low-dimensional Euclidean space. Metric embedding is mainly used for data visualization and exploratory analysis [43,110]. For example, Hinton and Roweis [110] propose a stochastic neighbor embedding to map the high-dimensional objects into a low-dimensional space while preserving neighborhood identity. The technique has also been used to embed objects into low-dimension space based on their co-occurrence data [111,112] to compute semantic similarity. In addition, metric embedding is applied for recommendation systems [45,113], where items and users are embedded in a common Euclidean space.

Recently metric embedding is adopted in sequential transitions, such as music playlist prediction. To generate the playlists, Chen et al. [4,38] develop a Logistic Markov embedding model (LME). LME maps each song to one or two points in a latent Euclidean space. The transition probability from one song to another is related to the Euclidean distance between the two songs in the latent space. Moore et al. [114] embed songs and tags together for predicting playlist. In the studies [115,116], both users and songs are further jointly embedded into a Euclidean space to reflect the individualized Markov transitions.

Our work (PRME) differs from the aforementioned studies in that we exploit pairwise ranking scheme to learn the parameters, and adapt Metric Embedding for the POI recommendation task by incorporating multiple factors.

2.3.2 Word2vec

Recently, word2vec technique has attracted extensive research attention. Word2vec techniques [3] are developed for the neural network language models. The latent representations learned by word2vec approaches have been shown to capture semantic meanings and are useful in various NLP applications. Given a sequence of words: \( w(1), w(2), ..., w(t - 1), w(t), w(t + 1) ..., \) the context of word \( w(t) \) is \( C(t) = \{w(t - c), ..., w(t-1), w(t+1), ..., w(t+c)\} \), where \( c \) is the size of context window. As illustrated by Figure 2.1, two architectures are designed in word2vec: continuous bag-of-word
model (CBOW) and Skip-gram model. The CBOW predicts the current word \( w(t) \) given its context \( C(t) = \{w(t - 2), w(t - 1), w(t + 1), w(t + 2)\} \). Meanwhile the Skip-gram model tries to predict the contextual words based on current word.

The prediction probability is defined by a softmax function. Since it is very time consuming to directly calculate the softmax function, two efficient solutions are developed [5]. One is hierarchical softmax [49], which additionally exploit a binary tree to approximate the softmax function. The other is negative sampling [117], which estimates the softmax function by randomly generating several negative instances. The parameters (items’ representations) can be inferred by Stochastic Gradient Descent (SGD) method [118].

Inspired by the word2vec technique, many models are proposed to learn representation in natural language. Le et al. [119] develop a paragraph vector model to learn the latent representations for variable number of words, such as sentences or documents. Djuric et al. [120] learn the vector representations for the streaming documents by using a hierarchical framework. To jointly learn the embeddings of words and documents, they combine two components: document stream model and word sequence model. Pennington et al. [121] propose a Glove algorithm to learn the global vectors for word representation. Glove is able to take advantages of both matrix

![Figure 2.1: The word2vec architectures.](image-url)
factorization and word2vec. Matrix factorization method can capture the global statistical information and word2vec do better to capture the local semantic relations. To combine these two methods, a global log-bilinear regression model is developed, which empirically obtains better performance than word2vec technique.

Although word2vec techniques are proposed for natural language processing, they have been adopted by a variety of other applications, such as network embedding [40, 46] and next product recommendation [98]. In particular, [39] utilizes the word2vec technique to model the check-in sequence. However, it fails to incorporate the geographical influence when modeling the sequential POIs, and cannot jointly model the user preference and sequential information. Different from the existing word2vec techniques, we propose a new model POI2Vec, which learns representation vectors of POIs by considering the geographical influence of POIs.
Chapter 3

Personalized Ranking Metric Embedding for User Sequential Mobility

In this chapter, we introduce the personalized ranking metric embedding for modeling the personalized sequential mobility. We first present the observations on real-world datasets and define the next new POI recommendation problem in Section 3.1. Then, in Section 3.2, we introduce the personalized Metric Embedding model. Next, we state the algorithm to learn parameters in Section 3.3. Section 3.4 reports the experimental study on real datasets. Lastly, we conclude this work in Section 3.5.

3.1 Next New POI Recommendation

We first present our observations in the analysis of two real-world datasets. Accordingly, we formulate the \( N^2 \)-POI recommendation problem.

We use two publicly available datasets. The first dataset is the FourSquare check-ins within Singapore [9] while the second one is the Gowalla check-ins dataset within California and Nevada [6]. We use one-year data in both datasets. Each check-in is a tuple in the form of \(< user, POI, time >\). Each POI is associated with the latitude and longitude. We remove users who have check-in fewer than 10 POIs, and POIs which have been visited by fewer than 10 users. The basic statistics of datasets used in this chapter are summarized in Table 3.1.
# Table 3.1: Statistics of two location-based datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>#User</th>
<th>#POI</th>
<th>#Check-in</th>
<th>Time range</th>
</tr>
</thead>
<tbody>
<tr>
<td>FourSquare</td>
<td>1917</td>
<td>2675</td>
<td>155365</td>
<td>08/2010-07/2011</td>
</tr>
<tr>
<td>Gowalla</td>
<td>4996</td>
<td>6871</td>
<td>245157</td>
<td>11/2009-10/2010</td>
</tr>
</tbody>
</table>

## 3.1.1 Observations on real-world datasets

### 3.1.1.1 Observation 1: exploration of new locations

Figure 3.1 shows the average ratio of new POIs over all users on two datasets for every 50 days. For example, the ratio at the 100\textsuperscript{th} day is the proportion of POIs visited at the 100\textsuperscript{th} day that have not been visited in the previous days. The ratio of new POIs is pretty high (most of the ratios above 0.4) on both datasets, which implies that people always like to explore new POIs. This result is consistent with the recent observation in [48].

![Figure 3.1: The average ratio of new POIs with number of days](image)

### 3.1.1.2 Observation 2: temporal influence

Figure 3.2 shows the cumulative distribution function (CDF) of the time difference of two sequential check-ins. Figure 3.2 demonstrates that more than 50% successive check-ins occur in less than 24 hours. Meanwhile, many consecutive check-ins occur in a longer time. For more than 25% consecutive check-ins, their time differences are

![Figure 3.2: CDF of time difference of two sequential check-ins](image)
larger than 48 hours. When time difference of two successive check-ins is large, their correlation is weak.

![Figure 3.2: CDF of time difference between two consecutive check-ins.](image)

3.1.1.3 Observation 3: spatial influence

We compute the geographical distance of two consecutive check-ins and plot the CDF distribution in Figure 3.3, which shows that 70% consecutive check-ins have less than 10km in both datasets. The CDF curve increases fast when distance is small, which implies that most check-ins occur in nearby areas. This result indicates that users’ next movements are influenced by their current locations. The finding is in accordance with the reported result [9, 19]. Consequently, the spatial influence is necessary to be taken into account in the next POI recommendation problem.

3.1.2 Next new POI problem definition

When two check-ins occur in a short time period, Markov chain property exists [19]. This motivates us to consider the POI transition within a short period, which means the next POI is influenced by current POI. Following [73], if the time interval of two consecutive check-ins is smaller than $\tau$, sequential influence shall be considered. Here $\tau$ is the time threshold. In Section 3.4.3, we investigate the impact of $\tau$. Meanwhile, according to observation 1, users incline to visit new POIs for their exploration interests, indicating that the problem of suggesting new POIs for users is meaningful.
Based on the sequential property within a short time period and users’ willingness of new POIs, we formally define the $N^2$-POI recommendation problem below.

**Definition 1 ($N^2$-POI Recommendation Problem )** For a set of users $U$ and a set of POIs $L$, $C$ is the historical check-in data, and $L^u$ is the set of POIs that user $u$ has visited before. Given current POI $l^c$ of user $u$, the $N^2$-POI problem is to recommend a set of POIs $S^{u,l^c} \subset L$ for the user $u$ to visit next and the POIs are new to the user

$$S^{u,l^c} = \{l \in L \setminus L^u\}$$

Note that in the $N^2$-POI recommendation problem we only recommend new POIs to users. In contrast, the next POI prediction task outputs both the visited and unvisited POIs as the results. As reported in [48], if we only use the frequently visited POIs of a user to predict his next POIs, a high prediction precision can be achieved. In addition, it’s challenging to estimate the implicit transition probability of potential new POIs based on the sparse historical data. Thus, the $N^2$-POI recommendation problem is harder than the next POI prediction problem.

### 3.2 Personalized Ranking Metric Embedding

The challenge of Next New POI recommendation is to learn transitions of users check-ins that are commonly represented by a first-order Markov chain model. Due to the
Chapter 3. Personalized Ranking Metric Embedding for User Sequential Mobility

sparse transition data, it is difficult to estimate the transition probability in Markov chain, especially for the unobserved transition. Recently, Metric Embedding has been proved can effectively capture sequential transitions, such as music playlist prediction [4, 38]. In addition, the pair-wise ranking algorithm [1, 55] has been widely exploited to address implicit feedback tasks. The POI recommendation problem can be viewed as one implicit feedback recommendation task [10]. Hence, we propose a novel pair-wise ranking metric embedding technique to learn user mobility behaviors.

Note that both metric embedding and matrix factorization can be exploited to capture the user check-in behaviors. Most of the existing methods utilize matrix factorization technique to learn the user mobility, such as [10, 19, 48], and we are the first one to explore metric embedding to represent POIs in a latent space. Therefore, our work provides alternative method to learn representation of spatial items. What’s more, compared to matrix factorization, metric embedding has several additional advantages [45]. First, the representation of metric embedding can be more intuitively understood and visualized by people. Second, it is efficient to generate recommendation result by utilizing space indexing and searching techniques. Lastly, it can better incorporate new items in the latent space. Since the difference between metric embedding and matrix factorization goes beyond the scope of this thesis, we will not introduce details about this. (For more details, please read [45]).

In this section, we first introduce Metric Embedding algorithm with Pair-wise Ranking to model location transitions in Section 3.2.1. The Personalized Ranking Metric Embedding (PRME) is then presented in Section 3.2.2. Then Section 3.2.3 states PRME-G model, which jointly incorporates sequential transition, user preference, and geographical influence.

3.2.1 Pairwise Ranking Metric Embedding

3.2.1.1 Embedding POIs in latent space

To model the sequential information, we need learn the transition probability in a Markov chain model. However, due to the data sparsity, it is infeasible to estimate the transitions by using standard counting methods. Metric embedding model can
be used to handle the data sparsity and be generalized to the unobserved data. We represent each POI as one point in a latent space. We assume that Euclidean distance between POIs in the latent space reflects the transition probability. The larger the distance, the lower the strength of transitions. With all POIs embedded in a latent space, our model estimates the sensible transition probabilities of POIs. It is also possible to assign meaningful probabilities to those unobserved transitions.

In the Metric Embedding model, each POI \( l \) has a position \( X(l) \) in a \( K \)-dimensional space. Given the observed sequential POI transitions, the goal is to learn the positions of all POIs. We relate the transition probability of a pair, \( l_i \) and \( l_j \), to the Euclidean distance as defined in Eq. 3.1.

\[
\hat{P}(l_j|l_i) = \frac{e^{-||X(l_j) - X(l_i)||^2}}{Z(l_i)} \tag{3.1}
\]

where \( ||X(l_i) - X(l_j)||^2 = \sum_{k=1}^K (X_k(l_i) - X_k(l_j))^2 \), \( K \) is the number of dimensions of the latent space and \( Z(l_i) = \sum_{n=1}^{||l||} e^{-||X(l_n) - X(l_i)||^2} \) is the normalization term. Note that the normalization term \( Z \) embodies the repulsion force. Without this term, Eq. 3.1 may get an useless solution where all objects are mapped to the same point in the latent space.

Compared to matrix factorization in FPMC, Metric Embedding (ME) can better model the sequential transition. It represents each item as a single point in a latent space rather than two independent vectors. This representation is more natural to embody the latent relations of items. For example, given observed transition \( l_i \rightarrow l_j \) and \( l_j \rightarrow l_k, l_i \rightarrow l_k \) is expected to be a potential transition. ME is able to capture this kind of relation. \( ||X(l_i) - X(l_k)||^2 \) would be small, because both \( X(l_i) \) and \( X(l_k) \) will be pulled closely to a same position \( X(l_j) \).

### 3.2.1.2 Ranking based metric embedding

Eq. 3.1 only exploits the observed check-ins to learn the latent position of each POI. Since the observed data is very sparse, we learn the latent position by fitting the rankings for the POI transition. Consequently, we can additionally make use of the unobserved data to learn the parameters. We utilize POI pairs as training data and
optimally estimate the pair-wise ranking. We assume that the observed next POI is more related to current POI than the unobserved POI. For example, if transition \( l_i \rightarrow l_c \) is observed and \( l_i \rightarrow l_j \) is not observed, POI \( l_i \) should be ranked higher than POI \( l_j \). We model it as a ranking > over POIs:

\[
l_i > l_j \iff \hat{P}(l_i | l^c) > \hat{P}(l_j | l^c)
\]  

(3.2)

The goal of POI recommendation is to provide a ranking of all the items, and accordingly recommend the top items. Furthermore, since we are interested in the ranking of POIs, we can simplify the computation by only keeping the Euclidean distance ||\(X(l_i) - X(l_j)\)||^2 in the latent space (abbreviated as \(D_{l_i,l_j}\)). Instead of utilizing the exponential function, we use the Euclidean distance to rank the POIs.

\[
\hat{P}(l_i | l^c) > \hat{P}(l_j | l^c) \Rightarrow e^{-||X(l_i) - X(l^c)||^2} > e^{-||X(l_j) - X(l^c)||^2} \\
\Rightarrow ||X(l_i) - X(l^c)||^2 < ||X(l_j) - X(l^c)||^2 \\
\Rightarrow D_{l_i,l_j} - D_{l_j,l_i} > 0
\]  

(3.3)

### 3.2.2 Personalized Ranking Metric Embedding

Individual preference has been proved to be an important factor for the POI recommendation because each user would prefer some favorite POIs. Given current location \( l^c \) of user \( u \), the recommended \( N^2 \)-POI shall not only be related to \( l^c \), but also capture user’s preference.

We use ME to model the user-item preference. We project each POI and each user into a latent space. The distance between a user and a POI reflects the strength of their relations. Intuitively, if user \( u \) likes POI \( l \), the distance \( ||X(u) - X(l)||^2 \) in latent space should be small. Otherwise, \( ||X(u) - X(l)||^2 \) would be large. By doing this, we model the user-item preference in a latent low-dimension space.

Since ranking based metric embedding only exploits transition data of all users, it does not reflect the user-specific transition. We further develop a Personalized Ranking Metric Embedding (PRME) method, which considers sequential information and individual preference together. We utilize two latent spaces: one is the sequential
transition space and the other is the user preference space. Each POI \( l \) has one latent position \( X^S(l) \) in the sequential transition space, and the Euclidean distance of two POIs \( l_i \) and \( l_j \) is defined as \( \mathcal{D}^S_{l_i,l_j} = ||X^S(l_i) - X^S(l_j)||^2 \). In the user preference space, each user \( u \) has a latent position \( X^P(u) \) and each POI \( l \) has a latent position \( X^P(l) \). \( \mathcal{D}^P_{u,l} = ||X^P(u) - X^P(l)||^2 \) denotes the Euclidean distance of user \( u \) and POI \( l \) in the user preference space.

Given current location \( l^c \) of user \( u \), we model personalized sequential transition for a candidate POI \( l \) by combining the two kinds of metric. As two components contribute differently to the metric score of POI \( l \), we utilize a linear mixture to combine the two metrics.

\[
\mathcal{D}_{u,l^c,l} = \alpha \mathcal{D}^P_{u,l} + (1 - \alpha) \mathcal{D}^S_{l^c,l} \quad (3.4)
\]

where \( \alpha \in [0, 1] \) controls the weight of different kinds of distance.

Based on observation 2 (in Section 3.1.1.2), some successive check-ins have large time difference, which may indicate their irrelevance. We assume that if the time difference between two consecutive check-ins is larger than threshold \( \tau \), only the user preference is considered. We then recompute the distance metric \( \mathcal{D}_{u,l^c,l} \) below.

\[
\mathcal{D}_{u,l^c,l} = \begin{cases} 
\mathcal{D}^P_{u,l} & \text{if } \Delta(l, l^c) > \tau \\
\alpha \mathcal{D}^P_{u,l} + (1 - \alpha) \mathcal{D}^S_{l^c,l} & \text{otherwise}
\end{cases} \quad (3.5)
\]

where \( \Delta(l, l^c) \) is the time difference of two successive check-ins \( l \) and \( l^c \).

### 3.2.3 Incorporating Geographical Influence

As stated in Section 3.1.1.3, given current location, users would like to visit the near POIs rather than the far way POIs. Geographical distance affects users’ visiting behavior. Thus we propose the PRME-G model to incorporate the geographical influence into the PRME model.

We accommodate the geographical influence by using the weight of the geographical distance. For any pair of POIs, we can calculate the geographical distance via their longitude and latitude. Given the geographical distance \( d_{l^c,l} \), we use a weight
function $w_{l,c}$. In this thesis, $w_{l,c} = (1 + d_{l,c})^{0.25}$. The fused distance metric becomes $D_{a,l,c} \cdot w_{l,c}$. When the distance of a POI is large, the fused metric would be large, and thus this POI is less likely to be recommended. The integrated metric with geographical influence is defined as below.

$$D^G_{u,l,c} = \begin{cases} D^P_{u,l,c} & \text{if } \Delta(l,l_c) > \tau \\ w_{l,c} \cdot (\alpha D^P_{u,l,c} + (1 - \alpha) D^S_{l,c}) & \text{otherwise} \end{cases}$$

Note that the PRME-G model is not special for N²-POI recommendation and is general for solving next POI prediction problem. We do not show the improvement for predicting next POI in this thesis.

### 3.3 Parameter Learning

#### 3.3.1 Optimization Criterion

Assuming that users’ check-ins are independent with each other, we can derive the optimization criterion of PRME model. Analogous to Bayesian Personalized Ranking (BPR) approach [55], we estimate the PRME model by using maximum a posterior (MAP):

$$\Theta = \arg \max_{\Theta} \log \prod_{u \in U} \prod_{l_c \in L} \prod_{l_i \in L} \prod_{l_j \in L} P(>_{u,l,c} | \Theta) P(\Theta)$$

where $\Theta = \{X^S(L), X^P(L), X^P(U)\}$ is the set of parameters.

Using logistic function $\sigma(z) = \frac{1}{1 + e^{-z}}$, ranking probability can be further written as

$$P(>_{u,l,c} | \Theta) = P \left( (D_{u,l,c,j} - D_{u,l,c,i}) > 0 | \Theta \right) = \sigma(D_{u,l,c,j} - D_{u,l,c,i})$$

With Gaussian priors on the parameters, we have the final objective function in Eq. 3.9, where $\lambda$ is a parameter controlling the regularization term.

$$\Theta = \arg \max_{\Theta} \log \prod_{u \in U} \prod_{l_c \in L} \prod_{l_i \in L} \sum_{l_j \in L} \log \left( \sigma(D_{u,l,c,j} - D_{u,l,c,i}) \right) - \lambda ||\Theta||^2$$
3.3.2 Learning Algorithm

Directly solving Eq. 3.9 is time consuming. Following the approach of BPR, we independently draw the training tuple and utilize Stochastic Gradient Descent to update the parameters. Based on the historical check-in data, we can obtain a set of observations \( \{ <u, l^c, l_i > \} \), where \( l^c \) is the current location of user \( u \) and \( l_i \) is the next check-in. For each observation, we randomly generate a POI \( l_j \), which is not observed given \( u \) and \( l^c \). Given a training tuple \( <u, l^c, l_i, l_j> \), the update procedure is described as below:

\[
\Theta \leftarrow \Theta + \gamma \frac{\partial}{\partial \Theta} \left( \log \sigma(z) - \lambda \| \Theta \|^2 \right)
\]

where \( z = D_{u,l^c,l_j} - D_{u,l^c,l_i} \), and \( \gamma \) is the learning rate.

The learning algorithm of PRME is summarized in Algorithm 1. First, we initialize all the parameters with a Normal distribution (Line 1). For each observation \( <u, l^c, l_i> \), we randomly generate a POI \( l_j \) that user \( u \) has not visited after \( l^c \). We then calculate the time difference \( \Delta(l^c, l_i) \) to determine whether it is related to \( l^c \). If \( \Delta(l^c, l_i) < \tau \), latent positions in both user preference latent space (Line 6) and sequential transition space (Line 7) are updated. If \( \Delta(l^c, l_i) \geq \tau \), only user preference is considered (Line 9). Note that all the update procedures (Lines 6, 7, 9) are based on Eq. 3.10. For PRME-G model, we utilize \( D_{u,l^c,l}^G \) in Eq. 3.9 as the optimization criterion and use the similar procedure to learn parameters.

The time complexity of the algorithm is \( O(IK|C|) \), where \( I \) is the number of iterations, \( K \) is the number of dimensions and \( |C| \) is the number of observed check-ins in the dataset.

3.4 Experiments

3.4.1 Experimental Setup

3.4.1.1 Experimental setting

In the experiments, we use the two datasets introduced in Section 3.1. For the one-year check-ins data, we use the check-ins in the first 10 months as training set, the
Algorithm 1: PRME

**input**: check-in data $C$, learning rate $\gamma$, regularization $\lambda$, component weight $\alpha$, time threshold $\tau$

**output**: model parameters $\Theta = \{X^S(L), X^P(L), X^P(U)\}$

1. Initialize $\Theta$ with Normal distribution $N(0, 0.01)$;

2. repeat
3. for Each observation $(u, l^c, l_i)$ do
4. Randomly generate an unobserved POI $l_j$;
5. if $\Delta(l^c, l_i) < \tau$ then
6. Update $X^P(u), X^P(l_i), X^P(l_j)$;
7. Update $X^S(l^c), X^S(l_i), X^S(l_j)$;
8. end if
9. if $\Delta(l^c, l_i) \geq \tau$ then
10. Update $X^P(u), X^P(l_i), X^P(l_j)$;
11. until convergence;
12. return $\Theta = \{X^S(L), X^P(L), X^P(U)\}$

11th month as tuning set, and the last month as test set. We exploit two well-known measure metrics [9], i.e., Pre@$N$ and Rec@$N$. Given a user and his current location, we use the next check-in in successive $\tau$ hours as the ground truth. The time window threshold $\tau$ is set at 6 hours following [19]. Based on the tuning set, the number of dimensions is set at $K = 60$, learning rate $\gamma = 0.005$, regularization term $\lambda = 0.03$ and component weight $\alpha = 0.2$.

3.4.1.2 Evaluated methods

In the experiments, we compare PRME and PRME-G with the following baselines:

- Popu: the most popular POIs are recommended as next candidate POIs (user count ranking scheme in [20]).
- UCF: user-based collaborative filtering for POI recommendation [7, 9]. Collaborative filtering is a widely used algorithm to model user preference in POI recommendation problems.
- MF: Matrix Factorization with Bayesian Personalized Ranking (BPR) method [55], which factorizes the user-item preference matrix. Matrix factorization is also widely studied for POI recommendation tasks. The BPR is proposed for solve implicit feedback recommendation problems. Since the user mobility can be
viewed as an implicit feedback problem, we utilize the BPR algorithm to factorize the user-POI matrix.

- **MC**: first order Markov chain model [73], which computes the transition probability by counting method. The transition probability between POIs is directly computed based on the counting method.

- **PME**: the personalized metric embedding [115, 116], which projects users and locations into a common latent space. PME can be utilized to jointly model user preference and sequential transition.

- **FPMC**: the state-of-the-art personalized next POI recommendation algorithm [1, 19].

The Popu is a very naive baseline. UCF and MF are most widely used techniques for conventional POI recommendation problems, and they do not exploit the sequential transition information. MC is a common method to model the markov chain transition, but does not incorporate individual POI preference. By comparing with the performance of MC, we can investigate the effect of user preference in next POI recommendation tasks. PME and FPMC consider the user preference and sequential transition together, which is similar to our proposed PRME. Note that PRME is different from PME in two aspects. First, PRME utilizes the pair-wise ranking scheme, which is more suitable for implicit feedback recommendation tasks. Second, PRME maps the user preference and sequential transition into two independent latent spaces, while PME projects user preference and sequential transition into a common space.

### 3.4.2 Performance of Methods

We compare the precision and recall of various methods in Figure 3.4 and Figure 3.5. The lower precision and recall of Popu method indicates that this naive approach is not feasible for the next POI recommendation. Both UCF and MF perform poorly because they do not make use of sequential information. This result shows the conventional POI recommendation algorithms, which mainly exploit the user preference, are not effective for the $N^2$-POI recommendation. The relatively high performance of
MC method demonstrates that the sequential information plays an significant role in the $N^2$-POI recommendation. The PME performance is not acceptable because the learning of sequential transition and user preference would be interfered with each other in a common latent space. Note that FPMC is state-of-the-art personalized Markov chain algorithm. PRME consistently outperforms FPMC, which illustrates that representing each POI as one point in latent space is more effective than two independent vectors. PRME-G achieves the best performance, which implies that the geographical influence is beneficial for the $N^2$-POI recommendation. All reported improvements over baseline methods are statistically significant with $p$-value $< 0.01$.

We further compare PRME and FPMC with the localized region constraint [19]. Figure 3.6 shows the precision with different region constraints. The local region constraint indicates that we only consider the candidate locations with no more than $d$ km ($d=10, 20$ and $40$) from current location. Note that PRME and FPMC use the same setting. Under different local region constraints, PRME outperforms FPMC, which further verifies the advantage of PRME.
3.4.3 Effect of Parameters

3.4.3.1 Effect of time threshold $\tau$

Figure 3.7 depicts the impact of the time threshold $\tau$ ($\tau = 3$, 6 and 24 hours). PRME and PRME-G outperform the baselines with various $\tau$. In addition, when $\tau$ is large, the performance of sequence based methods (MC, PME, FPMC, PRME and PRME-G) decreases. This is because the location sequential transition becomes weak when time difference is large.
3.4.3.2 Effect of component weight $\alpha$

Figure 3.8 shows the effect of weight $\alpha$. The performance at $\alpha = 0$ (only sequential transition) is much better than $\alpha = 1$ (only user preference). This result implies that sequential influence is more important than user preference in the $N^2$-POI recommendation problem. The best performance is obtained when $\alpha = 0.2$. Hence, we set $\alpha = 0.2$ in our experiments.

3.4.3.3 Effect of number of dimensions $K$

We further investigate the impact of $K$ with FPMC and PRME. Figure 3.9 shows the precision and recall for various $K$. When $K \geq 20$, PRME outperforms FPMC in recommendation quality, which implies the superiority of PRME. The performance of
FPMC and PRME increases with $K$ because high dimensions can better capture the latent metric relations. Empirically, we set $K = 60$ to obtain a satisfying trade-off between recommendation quality and running time.

3.5 Conclusion

We summarize the main contributions of this chapter as follows.

- We develop a pair-wise Metric Embedding algorithm to model the sequential
transition of POIs. To the best of our knowledge, this is the first work that uses the Metric Embedding method for the POI recommendation.

- To model the personalized sequential information, we propose a novel PRME algorithm, which jointly considers sequential transition and user preference. We further develop a PRME-G algorithm to incorporate geographical influence for the $N^2$-POI recommendation problem.

- We conduct comprehensive experiments by comparing our algorithms with the state-of-the-art techniques over two real-world datasets.
Chapter 4

POI2Vec Representation Model for User Sequential Mobility

In this chapter, we introduce a new representation model for user mobility. We first define the future visitor prediction problem in Section 4.1. Then, we introduce the POI-to-vector (POI2Vec) model in Section 4.2. Section 4.3 reports the experimental results on two real-world datasets. Lastly, we conclude this work in Section 4.4.

4.1 Future Visitor Prediction Problem

It is of great value to identify the potential visitors for locations. For instance, it can assist POI holders to find their potential customers. However, most of existing POI recommendation studies focus on suggesting POIs for users. In contrast, very little research has been done to predict visitors for a location. To fill this gap, in this thesis, we investigate a new research problem from the perspective of POI: predicting future visitors. For example, given a restaurant and the current time, we want to predict potential consumers who would visit this restaurant in the next several hours.

We denote the set of users by \( U \), and the set of POIs by \( L \), where each POI \( l \) is associated with its geographical coordinates \( (l^{\text{Lat}}, l^{\text{Lon}}) \). Let \( \mathcal{H} \) be the historical check-in dataset. Each check-in \((u, l, t)\) means that user \( u \) visits location \( l \) at time \( t \). Given a POI, we aim to identify the potential visitors who will visit the POI in the next few hours. The problem is defined as follows.

Definition 2 (Future Visitor Prediction Problem) Consider a set of users \( \mathcal{U} \) and a set of POIs \( \mathcal{L} \). Given a POI \( l \), current time \( t \), and time threshold \( \tau \), the problem
aims to identify a set of users who will visit this POI in the time period $[t, t + \tau]$

$$S^{t, t} = \{u \in U\}$$

A users mobility is strongly affected by his recent visits and his personal interest [19]. However, the users check-in data are very sparse, and it is difficult to model the sequential influence of POIs and user preference. Consequently, the main challenge for predicting future visitors for a given POI is how to effectively learn POI sequential transition and user preference simultaneously, and integrate them for prediction.

The progress in neural network language models demonstrates that the word2vec technique [3, 5] is able to effectively capture the sequential semantic relationships among words. Hence, in this thesis, we investigate the word2vec technique in order to model check-in data.

Very recently, the word2vec algorithm has been utilized to model users’ sequential check-ins [39]. But, it fails to incorporate geographical influence of POIs, i.e., users tend to visit nearby POIs, in the word2vec model. The geographical influence is very important in modeling user movement behaviors as shown in almost all the previous works in POI recommendation. However, there is no straightforward way to consider geographical information within word2vec technique. To this end, we develop a hierarchical softmax approach to incorporate the geographical influence into the process of learning representation, which will be stated in the following sections.

### 4.2 POI2Vec Representation Model

We first present a POI-to-Vector (POI2Vec) representation method to model the POI sequential transition. Then we extend the POI2Vec to jointly model user preference and POI sequential transition. Lastly, we introduce approaches to predict future visitors for a given POI.

The novelty of our method is that it is able to incorporate the geographical influence. Although geographical influence has been widely considered in POI recommendation tasks, there is no existing latent representation model that can directly
capture such geographical factor. Most of the existing methods exploit geographical influence as constraints when predict users’ check-in behaviors. For example, Cheng et al. [19] utilize geographical influence as localized region constraint by only recommending near POIs to users. Unlike the existing methods, we explicitly employ POIs’ geographical coordinates to learn the representation of spatial items, by building a hierarchical region structure. To the best of our knowledge, this is the first work to incorporate the geographical influence in the framework of word2vec.

4.2.1 POI2Vec Model for Sequential Transition

The sequential transition between check-ins is very important for predicting future visitors since a user’s subsequent movement behavior is highly influenced by the previously visited locations. When the time interval of two successive check-ins is small, the next check-in is influenced by the previous one [19]. For example, to predict the potential customers in the next several hours for a cinema at 7 p.m., the users who are at a restaurant are more likely to visit this cinema than the users who are at a sports hall, since people incline to watch movies after dinner. However, due to the sparsity of check-in data, it is challenging to model the sequential transition.

4.2.1.1 Latent Representation Method

The recent progress in neural network language models [3, 119] indicates that latent representation methods can effectively capture the sequential semantic relationships among words.

To study the feasibility of adopting the neural network technique, we compute the distribution of POI frequency for 2 LBSNs datasets, which are used in the experiments. As shown in Figure 4.1, the POI frequency follows power law distribution. Note that the distribution is same to the word frequency distribution in natural language [40]. This connection motivates us to utilize the well-known word2vec technique [5] to model the check-in sequences.

We next introduce the basic word2vec framework to capture the POI sequential transition. Given a user $u$, and his current location $l_u^c$, the context $C(l_u^c)$ is the POIs
that \( u \) have visited before \( l_u^c \) within a predefined time period. We define \( C(l_u^c) = \{l_u^i, 0 < \Delta(l_u^i, l_u^c) < \tau\} \), where \( \Delta(l_u^i, l_u^c) \) is the time interval between visiting \( l_u^i \) and \( l_u^c \).

The goal of POI sequential modeling is to estimate the probability of visiting a POI given its context POIs.

We represent each POI \( l \) with a vector \( w(l) \in \mathbb{R}^D \) in \( D \) dimension latent space. We adopt the Continuous Bag-of-Words (CBOW) architecture [3], which is to predict an item given its contexts. The probability of \( \Pr(l|C(l)) \) is defined with a softmax function:

\[
\Pr(l|C(l)) = \frac{e^{(w(l) \cdot \Phi(C(l)))}}{Z(C(l))} \quad (4.1)
\]

where \( \Phi(C(l)) = \sum_{l_c \in C(l)} w(l_c) \) is the sum of vector of contextual POIs, and \( Z(C(l)) = \sum_{l_i \in \mathcal{L}} e^{(w(l_i) \cdot \Phi(C(l)))} \) is the normalization term.

### 4.2.1.2 Hierarchical Softmax

It is computationally expensive to directly compute Eq. 4.1 since calculating \( Z(C(l)) \) needs to enumerate each item \( l_i \in \mathcal{L} \). To alleviate this issue, we adopt the hierarchical softmax [49], which is widely used to compute softmax. The hierarchical softmax utilizes a binary tree to organize the \( |\mathcal{L}| \) items. In the binary tree, leaf nodes are POIs and the others are inner nodes. The \( |\mathcal{L}| \) leaves of the binary tree correspond to the items, and \( |\mathcal{L}| - 1 \) inner nodes are binary classifiers. The hierarchical softmax model approximates the softmax by estimating the probability of a path from the
root to the leaf node. The path to leaf node \( l \) is defined as a sequence of inner nodes \( path = (b_0, b_1, ..., b_n) \). Then the probability of observing \( l \) along the \( path \) can be estimated by

\[
Pr(l|C(l))^{path} = \prod_{b_i \in path} Pr(b_i|\Phi(C(l))).
\]  

(4.2)

Each inner node \( b \) in the binary tree has a latent vector \( \Psi(b_i) \in \mathcal{R}^D \), which can be viewed as the parameters of a binary classifier. Here \( Pr(b_i|\Phi(C(l))) \) is defined as

\[
Pr(b_i|\Phi(C(l))) = \sigma(\Psi(b_i) \cdot \Phi(C(l))),
\]  

(4.3)

where \( \sigma(x) = 1/(1 + e^{-x}) \) is a sigmoid function.

![Figure 4.2: Conventional binary tree for hierarchical softmax](image)

For example, as illustrated in Figure 4.2, the path of item \( l_2 \) is \((b_0, b_1, b_3)\). For a classifier in the binary tree, we define the left as "true" and right as "false". Then the probability of \( Prob(l_2|C(l_2)) = \sigma(\Psi(b_0) \cdot \Phi(C(l_2))) \cdot \sigma(\Psi(b_1) \cdot \Phi(C(l_2))) \cdot (1 - \sigma(\Psi(b_3) \cdot \Phi(C(l_2)))).\)

In the hierarchical softmax, the structure of the binary tree is important for achieving good performance [50]. The most widely used tree generating method is Huffman tree based on the frequency of items, since it can get the shortest average path length, thus obtain the best learning efficiency. However, the Huffman binary tree structure cannot reflect the geographical influence. Next, we will present a novel tree structure to incorporate the geographical information.
4.2.1.3 Incorporating Geographical Influence

The geographical influence is a very important factor to model the user mobility as shown in previous studies on POI recommendation. For example, a user is more likely to visit nearby places in the next several hours. However, the existing frequency-based structure (Huffman tree) of hierarchical softmax fails to capture the geographical influence. Since it only exploits frequency of items to build the binary tree, and geographical influence cannot be captured. To address this problem, we develop a geographical binary tree structure for the POI2Vec model, which incorporates the spatial information into the binary tree.

It is worth noting that we employ the binary tree structure to organize the POIs, rather than other common spatial structures, such as quadtree [122]. Recall that the reason of adopting hierarchical softmax is to reduce the computing cost of Eq. 4.1. By taking the binary tree, each internal node only has two children, which can be easily modeled by a binary classifier. If we utilize the quadtree, where each internal node has exactly 4 children, then each node will be represented as a 4 class classifier. The multiple-class classifier is not as efficient as the binary classifier, since we need handle multiple possible outputs in the multiple-class classifier. Therefore, the binary tree structure is better than quadtree. In fact, almost all the existing hierarchical softmax methods exploit the binary tree to organize items. In this work, we also take the binary tree to incorporate the spatial information of POIs.

Since the nearby POIs have high relevance, they should be assigned closely in the binary tree. We split the POIs into a hierarchy of binary regions such that the nearby POIs are more likely to be clustered into the same region. To build the binary hierarchy for POIs, we recursively split each region into two equal size sub-regions, until at least one edge of the regions is smaller than $2 \times \theta$, where $\theta$ is the region size threshold. In this way, the regions are rectangles larger than $\theta \times \theta$ square. The regions should also be represented in a binary tree manner. As illustrated in Figure 4.3, we firstly divide the POIs into two equal regions $R_0$ and $R_1$. Then we split $R_0$ into regions $R_{00}$ and $R_{01}$, and split $R_1$ into regions $R_{10}$ and $R_{11}$.
Figure 4.3: The illustration of building binary tree over POIs

In the resulted binary tree, each POI is assigned to a single region. For example, \( l_1 \) is in region \( R_{01} \) and \( l_2 \) is in region \( R_{11} \) in Figure 4.3. However, this fails to capture the spatial influences of POIs in different regions. For example, two POIs \( l_1 \) and \( l_2 \) are close to each other, but they belong to two different regions. To solve this issue, we further improve the method by assigning a POI to multiple regions.

To assign a POI to multiple regions whose POIs may be influenced by the POI, we consider the influence area of each POI because users’ movement can be affected by the spatial distance. The influence area is defined as a \( \theta \times \theta \) square centered at each POI. If the influence area of a POI \( l \) overlaps with a region \( R \) in the binary tree, we assign \( l \) to region \( R \). For instance, as shown in Figure 4.3, the square centered at POI \( l_1 \) overlaps with region \( R_{01} \) and \( R_{11} \), and thus POI \( l_1 \) is assigned to region \( R_{01} \) and region \( R_{11} \).

Since the size of the influence area \( \theta \times \theta \) is smaller than the regions, it is easy to prove the following result.

**Proposition 1** In the POI2Vec model, the number of regions of a POI will be 1, 2 or 4.
We denote the set of regions of POI $l$ by $\Omega^l$. For instance, in Figure 4.3, $\Omega^l_3 = \{R_{10}\}$, $\Omega^l_1 = \{R_{01}, R_{11}\}$, and $\Omega^l_4 = \{R_{00}, R_{01}, R_{11}, R_{10}\}$. The number of regions of POI for $l_3$, $l_1$ and $l_4$ are: $|\Omega^l_3| = 1$, $|\Omega^l_1| = 2$, and $|\Omega^l_4| = 4$.

If a POI is assigned to several regions, we compute the probability distribution over these regions as follows: The probability of $l$ belonging to region $R_i$ is $Pr(R_i) = \frac{S_{R_i}^l}{\sum_{R_k \in \Omega^l} S_{R_k}^l}$, where $S_{R_i}^l$ is the size of overlap between influence area of $l$ and region $R_i$. For instance, the region probability distribution of POI $l_1$'s is: $Pr(R_{01}) = 0.7$ and $Pr(R_{11}) = 0.3$.

A region may contain many POIs, and we need to further construct a hierarchy to organize these POIs. For each region, we build a Huffman tree based on the frequency of POIs by following the work [3]. Therefore, in the generated binary tree, the upper levels are the hierarchy of spatial regions, and the lower levels are the hierarchy of POIs in each region. Figure 4.4 illustrates the binary tree generated by the POI2Vec model. A POI may have several paths. For example, $l_1$ occurs twice in the generated binary tree: one is in region $R_{01}$ and the other is in $R_{11}$.

In summary, the proposed POI2Vec method has two advantages over the conventional Huffman tree structure [3]. First, it considers the geographical influence of POIs in the process of building binary tree, and POIs under the same region node are geographically close to each other. Second, in the POI2Vec model, a POI may occur...
multiple times in the binary tree, while in the conventional model, each POI only occurs once. As demonstrated in [50], the multiple occurrence model would perform better than single occurrence model, since an item may have various relationships with other items.

Once the binary tree is generated, we can effectively compute the probability in Eq. (4.1) as follows.

### 4.2.1.4 Probability Estimation

Since each POI may have several regions in the POI2Vec model, a POI may correspond to multiple paths in the generated tree. We need to consider all of them for calculating the probability. We denote the set of paths for POI \( l \) by \( \mathcal{P}(l) \). Each path \( path_k \) is associated with a probability \( \Pr(path_k) \), which is the same with the probability of its corresponding region \( \Pr(R_k) \), where \( R_k \) is the region contained in the path \( path_k \). For instance, in Figure 4.4, POI \( l_1 \) has two paths in the binary tree: \( path_1 = (b_0, b_1, b_4, b_i) \) and \( path_2 = (b_0, b_2, b_0, b_j) \). Here \( path_1 \) contains the region \( R_{01} \), and \( path_1 \) contains the region \( R_{11} \). Recall that the region probability distribution of POI \( l_1 \) is \( \Pr(R_{01}) = 0.7 \) and \( \Pr(R_{11}) = 0.3 \). Thus \( \Pr(path_1) = 0.7 \) and \( \Pr(path_2) = 0.3 \). The probability of observing \( l \) given context \( C(l) \) is redefined as

\[
\Pr(l|C(l)) = \prod_{path_k \in \mathcal{P}(l)} \Pr(path_k) \times \Pr(l|C(l))^{path_k} \tag{4.4}
\]

Note that in Eq. (4.1), the time complexity of calculating \( \Pr(l|C(l)) \) is \( O(|L|) \). The number of leaf nodes in the POI2Vec model is \((a \times |L|)\), where \( a \) is the average number of paths of all POIs, and is a constant \((1 \leq a \leq 4)\) based on Proposition 1. In a binary tree, the average path length of \((a \times |L|)\) leaf nodes is \( O(\log(|L|)) \). Therefore, in Eq. (4.4), the time complexity of calculating one observation is \( O(\log(|L|)) \), which is greatly reduced from \( O(|L|) \).

### 4.2.1.5 Parameter Learning

The objective of POI2Vec model is to maximize the posterior probability of observing all sequential POIs, assuming the observations are independent with each other:

\[
\Theta = \arg \max_{\Theta} \prod_{(l, C(l)) \in \mathcal{H}} \Pr(l|C(l)), \tag{4.5}
\]
where $\Theta = \{W(L), \Psi(B)\}$ is the set of parameters. Here $W(L)$ indicates the latent representations of all the POIs $l \in L$, and $\Psi(B)$ is the set of parameters for the inner nodes. Since there are $(a \times |L|)$ leaf nodes in the binary tree, the total number of inner nodes is $(a \times |L| - 1)$. We can learn all the parameters by Stochastic Gradient Descent (SGD) method [118].

### 4.2.2 Extend POI2Vec Model for User Preference

User preference is another important factor for modeling users’ mobility and predicting potential users for the target POI. It is necessary to model user preference in the model. Previous work [39] models user preference and sequential transition separately, which fails to capture their interaction. We extend the POI2Vec model to jointly learn the user preference and sequential transition of POIs. Similar to POIs, each user $u$ is also represented as a vector $x(u) \in \mathbb{R}^D$ in the latent space.

As reported in [123], the time interval between some consecutive POIs is large. For a check-in, if there is no check-in in the last $\tau$ hours, only the user preference will be considered. In other words, for a check-in without recent context, this check-in is only related to the user preference. Similar to Eq. (4.1), the probability that user $u$ visits POI $l$ is estimated by

$$
Pr(l|u) = \frac{e^{(w(l) \cdot x(u))}}{Z(u)},
$$

(4.6)

where $Z(u) = \sum_{l_i \in L} e^{(w(l_i) \cdot x(u))}$ is the normalization term. Similar to $Pr(l|C(l))$, $Pr(l|u)$ can be computed by the hierarchical softmax.

For a check-in for which there exist recent historical check-ins, this check-in is related to both the user preference and contextual influence. We assume that user preference and contextual influence are independent. Given a user $u$ and location context $C(l)$, the probability of observing $l$ is

$$
Pr(l|u, C(l)) = Pr(l|u) \times Pr(l|C(l)).
$$

(4.7)

Depending on the presence of context or not, the probability of observing a check-in is computed by

$$
Pr(u, l, t) = \begin{cases} 
Pr(l|u, C(l)) & \text{if } C(l) \text{ exists} \\
Pr(l|u) & \text{otherwise}
\end{cases}
$$

(4.8)
The objective of the model is to maximize the posterior probability of observing all check-ins:

$$\Theta = \arg \max_{\Theta} \prod_{(u,l,t) \in \mathcal{H}} \Pr(u, l, t),$$

(4.9)

where $\Theta = \{W(L), X(U), \Psi(B)\}$ is the set of parameters. Here $X(U)$ is the set of latent representations for all the users. We learn the parameters similarly as we do for the POI2Vec model.

### 4.2.3 Predict Future Visitors

Once we learn the latent representations of users $X(U)$ and representations of POIs $W(L)$, we can utilize them to find users who will visit a POI in the future.

We first consider users who have recent check-ins in the last few hours. Given a time point $t$, if a user has check-ins in the time period $[t - \tau, t]$, we say this user is a user with recent positions. For such users, we make use of their recent positions to determine their inclination to visit the given POI. If a user visits several POIs in the period $[t - \tau, t]$, we only keep the latest POI $l^c$, which represents his latest location. Additionally, we also utilize the user preference. Given a target POI $l$, and user $u$ with his latest position $l^c$, the likelihood that $u$ visits $l$ in the near future is defined by: $F(x(u) \cdot w(l), w(l^c) \cdot w(l))$, where $x(u) \cdot w(l)$ reflects the user preference and $w(l^c) \cdot w(l)$ indicates the sequential influence. Here $F()$ is an aggregation function to fuse them.

In this thesis, we employ two popular aggregate functions to combine the user preference and sequential influence. The first aggregation is $Max$, which chooses the more important factor by taking the larger value: $F(a, b) = Max(a, b)$. The second aggregation function is $Sum$, which linearly combines the two factors: $F(a, b) = Sum(a, b)$.

Due to the sparsity of check-in data, most of the users do not have recent check-ins in the past several hours. For such users, since their latest positions are unknown, we can only exploit the user preference to predict the likelihood that these users visit the given POI. The prediction score is computed by $x(u) \cdot w(l)$.
To identify the potential visitors for a given POI, we need to consider users with recent positions and users without recent positions. For each user $u \in U$, we compute the score:

$$s(u, l) = \begin{cases} \mathcal{F}(x(u) \cdot w(l), w(l^c) \cdot w(l)) & \text{with recent positions} \\ x(u) \cdot w(l) & \text{otherwise} \end{cases} \quad (4.10)$$

We rank all the users by their scores and select top-K users as the potential visitors for the target location.

### 4.3 Experiments

#### 4.3.1 Experimental Setup

We conduct experiments on two publicly available datasets. The first one is the Foursquare check-ins within Singapore [9] and the second one is the Gowalla check-ins within Houston [69]. We remove the users who have fewer than 5 check-ins and the POIs which have been visited by fewer than 5 users by following the previous work. The statistics of datasets are presented in Table 4.1.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>#User</th>
<th>#POI</th>
<th>#Check-in</th>
</tr>
</thead>
<tbody>
<tr>
<td>Foursquare</td>
<td>2321</td>
<td>5596</td>
<td>194108</td>
</tr>
<tr>
<td>Gowalla</td>
<td>4627</td>
<td>15234</td>
<td>362783</td>
</tr>
</tbody>
</table>

Table 4.1: Statistics of Foursquare and Gowalla datasets

For both datasets, we use the first 90% chronological check-ins as the training set, the 90 – 95% as the tuning set, and the last 5% as test set. We consider two evaluation tasks. The first one is to evaluate the quality of latent representation for sequential transition in the next POI recommendation task. The second task is to evaluate the accuracy of future visitors prediction. Following [123], we set $\tau = 6$ hours in this thesis. Based on the tuning set, the number of dimensions $D = 200$, region size threshold $\theta = 0.1$, learning rate is set at 0.005.
4.3.2 Latent Representation for Sequential Transition

To evaluate the quality of latent representation based on sequential POIs, we conduct experiments of predicting successive check-ins for users. Following the work [123], we use two metrics: Pre@N and Rec@N. Given the current location of a user, we utilize the subsequent POIs in the next 6 hours as the ground truth. Since we aim at learning latent representation for sequential transitions, we only consider the baselines that can capture sequential influence in a latent space. We compare our proposed algorithm POI2Vec with 4 baselines in the task of predicting next POIs.

- (1) FMC: the factorized Markov chain model [1], which utilizes the Matrix Factorization to model the sequential transition. This model utilizes Bayesian personalized ranking scheme to learn the factorization of POI transition matrix.

- (2) ME: Metric Embedding model [123], which projects every POI into one object in a latent Euclidean space. In this model, each node is represented as a position in the latent space, and the Euclidean distances between them reflect their transition probability.

- (3) NS: the negative sampling technique for word2vec [3], which is used to model POI sequences [39]. This method is widely used in the word2vec technique. Instead of considering all the items when calculating $Z(C(l))$ in Eq. 4.1, it randomly generates a few of (typically, 5-10) items to calculate $Z(C(l))$.

- (4) HS: conventional hierarchical softmax with Huffman tree [3]. It only exploits the frequency of POIs to construct the binary tree for hierarchical softmax.

Note that all the four evaluated baselines can learn the representation from sequential information. Different from word2vec techniques, FMC utilizes matrix factorization and ME exploits metric embedding to learn the representations. NS and HS are typical word2vec algorithms, but they can not reflect the geographical information. Meanwhile Inf2vec can exploit the geographical influence with word2vec framework.
We compare the results of various embedding methods in Figure 4.5 and Figure 4.6. The latent representation models (NS, HS and POI2vec) perform better than factorization model (FMC) and Metric Embedding model (ME). This indicates that latent representation methods are effective for modeling check-in sequences. Our POI2vec consistently performs better than HS, and this demonstrates that the proposed binary tree structure is better than the conventional Huffman tree. For example, POI2Vec outperforms HS by 16% and 9% on Foursquare and Gowalla, respectively, in terms of Pre@5. This is because POI2Vec incorporates the geographical influence into the binary tree. Overall, POI2Vec outperforms the other benchmark algorithms, which implies that POI2Vec can effectively learn the representations of POIs.
4.3.3 Future Visitor Prediction

Given a target POI and a time point, we utilize the users who visit this POI in the next $\tau$ hours as the ground truth. We use Pre@N and Rec@N as the metrics. To evaluate the performance of predicting future visitors, we compare our methods with 3 state-of-the-art baselines.

- (1) **FPMC**: factorizing personalized Markov chains, which linearly combines the user preference and Markov transition [19].

- (2) **PRME**: personalized ranking metric embedding, which linearly fuses the user preference and Markov transition [123].
(3) **CWRAP**: which explores the context of locations to model user preference [39].

In our methods, we learn the latent representations of users and POIs by using the POI2Vec method. Then we investigate 4 approaches to predict future visitors for POIs.

- (1) **U**: We only utilize user preference to predict potential visitors.
- (2) **URP**: We only consider users with recent positions.
- (3) **MAX**: Max aggregation function is used in Eq. (4.10).
- (4) **SUM**: Sum aggregation function to integrate the user preference and sequential influence.

The experimental results are presented in Figure 4.7 and Figure 4.8. SUM achieves better results than FPMC and PRME. For instance, SUM outperforms FPMC and PRME by 29% and 32%, respectively, in terms of Pre@5 on Foursquare dataset. Note that all the three methods utilize the same linear aggregation. The experimental result demonstrates that the quality of the representation learned by POI2Vec is better than that by the others. U performs better than CWRAP, both of which utilize the user preference to predict the potential visitors. This also demonstrates that POI2Vec generates better users’ representations than CWRAP. CWRAP learns user representation and POI representation separately. It first learns the POI representation. Then the user representation is obtained while the POI representation is fixed. The result shows that it is more reasonable to jointly model the user preference and sequential transition. The performance of U is not as good as that of SUM, since it does not exploit users’ recent positions. URP is worse than SUM because it only considers the users with recent positions. This result shows that it is useful to consider both users with recent positions and users without recent positions. SUM outperforms MAX, and this indicates that combining user preference and sequential transition helps to predict potential users.
4.3.4 Effects of Parameters

4.3.4.1 Effect of region size

To investigate the effect of region size threshold $\theta$, we show the precision results in Figure 4.9. We observe that the performance first increases as we enlarge the region threshold, and then it decreases slightly. Small $\theta$ means that only very close POIs are clustered into the same region, which fails to capture the relationships with other POIs. The best performance is obtained at $\theta = 0.1$, which is about $11km$ on the earth. Hence, we set $\theta = 0.1$ in other experiments.

![Figure 4.9: Effect of region size threshold $\theta$.](image)

4.3.4.2 Effect of number of dimensions

This experiment is to evaluate the influence of the number of dimensions $D$. As shown in Figure 4.10, the performance improves with the increase of $D$. We set $D = 200$ empirically in our experiments by considering the trade-off between effectiveness and efficiency.

4.4 Conclusion

In this chapter, we propose a novel latent representation model for user sequential mobility. The main contributions are summarized as follows.

- We propose a novel latent representation model POI2Vec, which incorporates the geographical influence of POIs, to predict the potential visitors for a location
in the next few hours. To the best of our knowledge, this is the first work to incorporate the geographical influence in the framework of word2vec.

- We develop a method to jointly learn the latent representation for users and POIs. We aggregate the user preference and POI sequential influence to predict the future visitors for a POI. This model can also be used to predict the next POIs for a user.

- We conduct extensive experiments to compare our algorithm with state-of-the-art approaches, and our method significantly outperforms the baselines for both next POI prediction and future user prediction.
Chapter 5

Inf2vec Representation Model for Social Influence Embedding

In this thesis, we study two common fields with sequential information: user sequential mobility and social influence propagation. In chapter 3 and chapter 4, we present the investigations about user mobility sequential transitions, i.e., check-in sequences. Subsequently, we will study another type of sequential information. In this chapter and next chapter, we state the analysis of social influence propagation on social networks. Specifically, we will examine how to learn latent representation from information spread sequences on social networks in this chapter, and how to discover influential users based on the social influence spread information in chapter 6.

In this chapter, we introduce a social influence embedding model (Inf2vec) for learning the influence diffusion sequences. We first present the definition of social influence embedding problem in Section 5.1. Then, we show some observations from two real social networks in Section 5.2. Next, we state the latent representation algorithm Inf2vec in Section 5.3. Section 5.4 reports the experimental study. Lastly, we conclude this work in Section 5.5.

5.1 Social Influence Embedding

Recently, online social networks have become increasingly popular among people. People would share information (e.g., photos and news) on the social networks, and they can also watch the online activities of their friends. Social networks have been effective channels for information propagation among uses. Social influence analysis
has drawn extensive research attention \[91, 103, 124, 125\]. In our work, we focus on estimating the parameters of influence diffusion on social networks.

A social network can be modeled as a graph \( G = (V, E) \) where \( V \) is the set of users and \( E \) is the set of edges. An edge \((u, v)\) indicates that user \( u \) is a friend of user \( v \). In online social networks, a user can watch his friends’ behaviors and further be affected by them.

We are also given an action log \( A \), which records users’ online behaviors. The action log \( A \) contains a set of tuples \((u, i, t^i_u)\), which denotes that user \( u \) performs the action \( i \), e.g., like a story or photo, at time \( t^i_u \). Each item \( i \) is corresponding to one diffusion episode \( D_i = \{(u, t^i_u)\} \), which is a set of users who adopt action \( i \) in chronological order.

Based on the social network and action log, we can extract the social influence observations. Following [33, 34, 37], we assume that if there is an edge \((u_1, u_2)\) and user \( u_1 \) performs some actions before user \( u_2 \), then we say that \( u_1 \) influences \( u_2 \). We formally define the social influence as follows.

**Definition 3 (Social Influence Pair)** Given a social network \( G = (V, E) \) and a diffusion episode \( D_i \), we claim that social influence pair \((u_1 \rightarrow u_2)\) exists if it satisfies:

1. \( u_1 \in V \) and \( u_2 \in V \);
2. \((u_1, u_2) \in E \);
3. \( t^i_{u_1} < t^i_{u_2} \).

Based on the extracted social influence pairs, we further obtain an influence propagation network, which records how the information about \( i \) propagates in the social network \( G \). The propagation network \( G_i \) is a subset of the social network \( G \).

**Definition 4 (Influence Propagation Network)** Given a social network \( G = (V, E) \) and a diffusion episode \( D_i \), the propagation network is \( G_i = (V_i, E_i) \), which satisfies:

1. \( V_i \subset V \) and \( E_i \subset E \);
2. For each \((u, v) \in E_i \), there is social influence pair \((u \rightarrow v)\) in diffusion episode \( D_i \).

Figure 5.1 illustrates the idea of obtaining influence propagation networks. The example social network contains 5 users. Given the action sequence in episode 1, since
user 1 performs the action before user 2, thus influence propagation \((1 \rightarrow 2)\) may exist. In this way, we obtain three social influence pairs: \(\{(1 \rightarrow 2), (2 \rightarrow 3), (1 \rightarrow 2)\}\). By combining these social influence observations, we further get the influence propagation network 1. Similarly, we can generate the propagation network 2 given episode 2.

The influence propagation network is a directed acyclic graph. Each node may have multiple children and also may have several parents. There is no cycle in the propagation network due to the time constraint.

As a fundamental problem of social influence analysis in social networks, modeling
influence propagation has been investigated in recent years [33,35,91]. Given a social
network and social influence observations, it aims to infer the influence diffusion prob-
abilities between users. Various methods have been proposed to learn the propagation
parameters. Most of the existing methods [33, 35, 36] directly estimate the diffusion
probability for each edge. One common method is maximum likelihood estimation
(MLE) model [33]. It directly uses the co-occurrence counting approach to estimate
the propagation probabilities. Another widely used method is utilizing the Expecta-
tion Maximization (EM) technique to learn the diffusion probabilities [34,35,37].

Figure 5.2(a) shows the basic idea of conventional social influence learning meth-
ods, which learn the propagation probability for each edge. For example, in Fig-
ure 5.2(a), the propagation probabilities $\{P_{12}, P_{13}, P_{23}, P_{14}, P_{45}\}$ need to be learned.
Generally, these conventional methods attempt to estimate the probabilities of all the
$|E|$ edges. However, due to the sparsity of diffusion data, these approaches cannot
effectively learn the propagation parameters. Especially, it is impossible to learn the
diffusion parameters for unobserved data. For instance, if no social influence pair has
been observed on a link $(u, v)$, it’s hard to estimate the propagation probability $P_{uv}$.

Recently, network embedding [40, 46, 47, 97] has been widely investigated, which
represents the network into a latent low-dimensional space. In network embedding,
each node is represented as a vector, and the network structure can be captured by
the vectors of nodes. Inspired by the network embedding, we aim to represent the
social influence propagation into a latent space. In the social influence embedding
problem, we attempt to learn the representations of $|V|$ nodes in the given network.

We show the basic idea of social influence embedding in Figure 5.2(b). Differ-
ent from the conventional approach in Figure 5.2(a), we learn the representations for
nodes: $\{1, 2, 3, 4, 5\}$. In social influence embedding, the propagation relationship be-
tween two users is modeled by the similarity between vectors. Note that propagation
network is a directed graph, and the propagation is asymmetry. We utilize two vec-
tors to represent user $u$: $S_u$ acts as the source representation and $T_u$ acts as target
representation.
In social influence analysis, we need to consider the global property for each user, which cannot be reflected by the latent vectors. Intuitively, some users, such as movie stars and politicians, are more influential than ordinary users in social networks. In addition, some users are very active on social networks and frequently check the status of online social medias. They are more inclined to be affected by others. To better model the social influence, we introduce two terms: influenceability bias $b_u$ reflects the ability of user $u$ to affect others and conformity bias $\tilde{b}_u$ reflects a user’s inclination to be influenced by others [126].

We define the social influence embedding problem as follows.

**Definition 5 (Social Influence Embedding Problem)** Given a social network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, an action log $\mathcal{A} = \{D_i\}$, where $D_i$ is a diffusion episode, number of dimension of latent space $K$. We aim to learn: (1) the embedding $(S_u, T_u)$ in $K$ dimensional latent space for each user $u$, where $S_u \in \mathcal{R}^K$ and $T_u \in \mathcal{R}^K$. (2) influenceability bias $b_u$ and conformity bias $\tilde{b}_u$.

Our study is highly related to the work by Bourigault et al. [37]. They propose an embedded cascade model, which is to learn the embeddings of all the nodes. Each node is represented by two vectors in a low-dimensional latent space. The propagation relationship between two nodes is captured by the Euclidean distance of their vectors.
But there are several limitations of this approach. First, it does not utilize the network structure. It assumes that if user \( u_1 \) performs an action before user \( u_2 \), then there exists social influence between \( u_1 \) and \( u_2 \). This assumption is not true because only if the edge \((u_1, u_2)\) exists then the information would spread from \( u_1 \) to \( u_2 \). Thus, we need to explicitly consider the social graph to learn the influence propagation parameters. Second, the proposed algorithm is not efficient and cannot solve the large-scale network. Since it utilizes the EM technique to learn the parameters following the idea of [35]. Third, it fails to incorporate other features, such as user similarity. In practice, when users have an online behavior, they may have many reasons: they may be influenced by their online friends, or motivated by their personal interest. In other words, user’s activities are not always related to social influence. In most cases, the online actions can reflect a user’s individual preference. Therefore, we need to consider user’s personal interest in modeling influence propagation.

5.2 Data Observations

5.2.1 Datasets

To study the social influence on social networks, we use two publicly available datasets. One is Digg, which contains information about stories displayed on the front page of Digg (digg.com) in June 2009 [127]. The social network in Digg has 68K users connected by 823K edges. If user \( u \) lists user \( v \) as a friend, \( u \) can see \( v \)'s activities. The dataset also contains the Digg-votes, each of which records users’ voting on a particular story and the voting time. The other dataset is Flickr, which contains a friendship graph and a list of favorite marking records of the photo sharing social network (www.flickr.com) [124]. There are 162K users connected by 10M edges. The statistics of two datasets are stated in Table 5.1.

5.2.2 Observations

Based on the social graph and action log, we extract the social influence observations. For a user \( v \), if his friend \( u \) performs the same action before him, then there exists
Table 5.1: Statistics of Digg and Flickr for social influence embedding

<table>
<thead>
<tr>
<th>Dataset</th>
<th>#User</th>
<th>#Edge</th>
<th>#Item</th>
<th>#Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digg</td>
<td>68,634</td>
<td>823,656</td>
<td>3553</td>
<td>2,485,976</td>
</tr>
<tr>
<td>Flickr</td>
<td>162,663</td>
<td>10,226,532</td>
<td>14,002</td>
<td>2,376,230</td>
</tr>
</tbody>
</table>

social influence pair \((u \rightarrow v)\) between them. In this way, we get 7.9M social influence pairs for Digg and 5.3M pairs for Flickr. Each social influence pair \((u \rightarrow v)\) contains a source user \(u\) and a target user \(v\). To examine the characteristic of the social influence observations, we plot distributions of the source user frequency and target user frequency.

Figure 5.3 illustrates the frequency distribution of source user on Digg and Flickr. We can learn that the source user frequency follows a power-law distribution. The high frequency of a user being source user indicates that this user is influential. Most of the users are not influential, while some users are extremely influential on social networks. This observation validates the idea of influenceability bias.

Similarly, in Figure 5.4, the distribution of target user frequency also follows the power-law distribution. The user frequency distribution indicates that some users are more likely to be influenced by their friends. This observation justifies the introducing of conformity bias.

![Distributions of source user frequency on Digg and Flickr.](image)

To investigate the effect of social influence in users’ online behaviors, we compute the cumulative distribution function (CDF) of the count of friends that perform the
action before a user. Figure 5.5 shows the CDF of Digg and Flickr. In Digg dataset, the CDF of \( x = 0 \) is 70%, which indicates that 70% users conduct an activity without any influence from his friends. Meanwhile, 30% users perform an action after at least one of his friends do that. Since a user is able to watch his friends’ online activity, we assume that this user would be influenced by his friends. For Flickr dataset, 50% users perform an action without any social influence. The remaining 50% users’ online activity may be affected by social influence. This observation demonstrates that social influence plays a significant role in the decision of online behaviors for users, but we still need to consider more factors.
5.3 Latent Representation Model for Social Influence Propagation

In this section, we introduce the Influence-to-vector (Inf2vec) representation method to model the social influence. We first present how to obtain the training instances. Then we state the procedure to learn the users’ embeddings.

5.3.1 Generating Influence Context

5.3.1.1 Modeling Social Influence

Given social influence propagation observation \((u \rightarrow v)\), we aim to model it with the transition probability \(P(v|u)\). Most of the existing methods assume that the propagation process is based on a predefined diffusion model. There are two widely used models: Linear Threshold (LT) model [22] and Independent Cascade (IC) model [23]. Similar to our social influence embedding problem, Bourigault et.al [37] develop an embedded cascade model, which is designed for the independent cascade model. However, the proposed model cannot be easily adopted for other diffusion models. In our work, we directly model the social influence observations, without any assumption of the influence diffusion model.

Additionally, we consider the high-order propagation to reflect influence spread in a social network. The influence would spread from one user to other users, who are not confined to the first-order neighbors. For example, given two social influence observations \((u_1 \rightarrow u_2)\) and \((u_2 \rightarrow u_3)\), we can infer that \(u_1\) may affect \(u_2\). Since \(u_2\) may affect \(u_3\), as a result, \(u_1\) can affect \(u_3\) indirectly. Based on the influence spread in social networks, we utilize a random walk process to model a user’s influence spread in the influence propagation network, which is generated by the social influence observations.

Given a influence propagation network \(G_i = (V_i, E_i)\) and a user \(u\), we generate the social influence context set \(C_u\), which contains the users that are influenced by user \(u\). We utilize a random walk with restart strategy to generate the \(C_u\). Starting from the user \(u\), it randomly chooses one neighbor to visit. Based on the currently visited
user, it randomly samples one neighbor of this user to visit next. At each step, it has some probability $\frac{1}{2}$ to go back to user $u$. To constraint the size of $C_u^i$, we utilize a length threshold $L_{\theta}$. The random walk process stops when the $L_{\theta}$ is reached. In this way, both the first-order neighbors and high-order neighbors are taken into account in the social influence neighborhood set $C_u^i$.

### 5.3.1.2 Incorporating User Similarity

As shown in Figure 5.5, users’ online behaviors are not always related to social influence. In this work, we further consider the similarity of user preference. User preference has been proven very important to model users’ behaviors, such as recommendation systems [11]. A user’s online action reflects his personalized interest. The users with similar interest are more likely to have the same behavior. Given a propagation network $G_i = (V_i, E_i)$, all the nodes in $V_i$ have performed action $i$, which means that they are interested in the same item. However, existing methods do not consider such information. To address this issue, we incorporate the similarity of user interest into the step of generating influence context $C_u^i$.

To capture the similarity of user interest, we additionally consider the users who perform the same action. Given a user $u$ in propagation network $G_i = (V_i, E_i)$, we randomly sample $L'_{\theta}$ users in $V_i$. Note that the social influence context only reflects the local neighborhood, while the sampling of similar users can reflect the global context. Therefore, it is necessary to incorporate some similar users into the social influence context $C_u^i$.

### 5.3.1.3 Generating Influence Context

Next, we propose an algorithm to generate the influence context pair $(u, C_u^i)$ in the propagation network by combining the local influence propagation and global user similarity. To reflect the contribution of these two components, we utilize a component weight $\alpha$.

---

1In our experiments, we set the restart ratio is 0.5, which makes the first-order neighbors and high-order neighbors contribute equally.
The pseudo code for generating influence neighbors is shown in Algorithm 2. Given a user \( u \in V_i \), we aim to generate the influence context \( C^i_u \). We first generate \( L \cdot \alpha \) local influence neighbors by using a random walk starting at user \( u \) (line 2). Next, we randomly sample \( L \cdot (1 - \alpha) \) users from all the nodes \( V_i \) (line 3). By this way, the influence context set \( C^i_u \) consists of two components (line 4).

**Algorithm 2: Generating Influence Context**

```
input : Propagation network \( G_i = (V_i, E_i) \), user \( u \in V_i \), length threshold \( L \), component weight \( \alpha \)
output: The user and its influence context \((u, C^i_u)\)

\( C_1 \leftarrow \emptyset, C_2 \leftarrow \emptyset, C^i_u \leftarrow \emptyset \);
\( C_1 \leftarrow (L \cdot \alpha) \) nodes by random walk from \( G_i \) starting at \( u \);
\( C_2 \leftarrow L \cdot (1 - \alpha) \) nodes by uniformly sampling from \( V_i \);
\( C^i_u \leftarrow C_1 + C_2 \);
return \((u, C^i_u)\);
```

### 5.3.2 Inf2vec Representation Model

The challenge of social influence embedding is how to effectively model the relationship between users and their influence contexts. We resort to word2vec technique [3, 5] to learn the latent representation for each user. Word2vec has been proven to well capture the semantic relationships among words. Very recently, word2vec has been adopted for learning network embedding [40, 47] and modeling user’s sequential behaviors [39].

In addition, as shown in Section 5.2.2, the social influence observations follow power law distribution, which is as same as the word frequency distribution in natural language. This connection also motivates us to utilize word2vec technique to model the social influence propagation. To the best of our knowledge, we are the first one to utilize the word2vec technique for the social influence embedding problem.

We exploit the skip-gram architecture [3], which is to predict the context of a given item. The goal of influence embedding is to estimate the probability of observing the influence context \( P(C^i_u|u) \). Assuming that the users \( \{v \in C^i_u\} \) are independent from each other, the probability \( \Pr(C^i_u|u) \) is determined by each independent probability
The transition relationship between \( u \) and \( v \) is captured by their representations: \( S_u \) is the representation of user \( u \) as sender, \( T_v \) is the representation of user \( v \) as receiver, \( b_u \) denotes the influenceability bias of user \( u \) and \( \tilde{b}_v \) denotes the conformity bias of user \( v \). Then the probability of \( \Pr(v|u) \) is defined with a softmax function.

\[
\Pr(v|u) = \frac{e^{(S_u \cdot T_v + b_u + \tilde{b}_v)}}{Z(u)} \tag{5.2}
\]

where \( Z(u) = \sum_{w \in V} e^{(S_u \cdot T_w + b_u + \tilde{b}_w)} \) is the normalization term.

We attempt to maximize the log probability of all \((u, C^i_u)\) pairs. The objective function can be written as

\[
O_I = \sum_{u \in V} \sum_{v \in C(u)} \log \Pr(v|u) \tag{5.3}
\]

It is computationally expensive to directly compute Eq. (5.3), since calculating \( \Pr(v|u) \) is time consuming. As shown in Eq. (5.2), calculating \( Z(u) \) needs to enumerate each item \( v \in V \). To alleviate this issue, we adopt the negative sampling [5], which is widely used to compute softmax functions. The idea of negative sampling is straightforward: instead of enumerating all the nodes, it only considers a small set of sampled nodes. We randomly generate multiple negative instances for each node \( v \in V \). Then we employ the sampled negative instances to approximate the softmax function.

\[
\log \Pr(v|u) \approx \log \sigma(S_u \cdot T_v + b_u + \tilde{b}_v) + \sum_{w \in \mathcal{N}} \log \sigma(-(S_u \cdot T_w + b_u + \tilde{b}_w)) \tag{5.4}
\]

where \( \mathcal{N} \) is the set of randomly sampled negative instances and \( \sigma(x) = \frac{1}{1 + e^{-x}} \) is sigmoid function. In Eq. 5.4, the first term reflects the observed instances, and the second term models the sampled negative instances.

We exploit Stochastic Gradient Descent (SGD) method [118] to learn all the parameters. In each step, we update the parameters \( \Theta \) by calculating the gradient:

\[
\Theta \leftarrow \Theta + \gamma \frac{\partial}{\partial \Theta} \log(\Pr(v|u)) \tag{5.5}
\]
where $\gamma$ is the learning rate. Based on Eq. 5.4, the gradient for corresponding parameters can be computed as follows.

\[
\begin{align*}
\frac{\partial}{\partial S_u} &= (1 - \sigma(z_1)) \cdot T_v + \sum_{w \in N} (-\sigma(z_2)) \cdot T_w \\
\frac{\partial}{\partial T_v} &= (1 - \sigma(z_1)) \cdot S_u \\
\frac{\partial}{\partial T_w} &= (-\sigma(z_2)) \cdot S_u \\
\frac{\partial}{\partial b_u} &= (1 - \sigma(z_1)) + \sum_{w \in N} (-\sigma(z_2)) \\
\frac{\partial}{\partial b_v} &= (1 - \sigma(z_1)) \\
\frac{\partial}{\partial b_w} &= (-\sigma(z_2))
\end{align*}
\]

where $z_1 = (S_u \cdot T_v + b_u + \tilde{b}_v)$ and $z_2 = (S_u \cdot T_w + b_u + \tilde{b}_w)$.

The learning algorithm of Inf2vec is summarized in Algorithm 3. It contains two parts: the first part (lines 3-7) generates the training instances, and the second part (lines 8-15) learns the parameters based on the training instances. The Inf2vec algorithm starts with initializing the parameters (line 1). We utilize a list $P$ to store the training instances $(u, C_i^u)$, where $u$ is a user and $C_i^u$ is the influence context of user $u$ in episode $D_i$. For each episode $D_i$ in action log, we obtain the propagation network $(V_i, E_i)$ (line 4) and get $C_i^u$ for each $u \in V_i$ by the randomized procedure described in Algorithm 2. For each observation $(u, C_i^u)$, we update the parameters within the skip-gram framework, which assumes that the items $v \in C_i^u$ are independent. For each pair $(u, v), v \in C_i^u$, we update the parameters by negative sampling 5.4 (lines 10-14). We first update the parameter for $(u, v)$ (line 11), and then update parameters for the sampled negative instances (lines 12-14). The parameters are updated by the SGD method in Eq. 5.5.

The time complexity of the learning algorithm is $O(I \cdot |P| \cdot N \cdot K)$, where $I$ is the number of iterations, $N$ is the number of negative sampling (typically, 5-10), $K$ is the number of dimensions and $|P|$ is the size of training instances in the dataset. The space complexity of Inf2vec is $O(|V| \cdot K)$, since we need to learn $K$-dimensional vectors for each node $v \in V$. 

83
Algorithm 3: Inf2vec

**input**: Social network $G = (V, E)$, action log $A = \{D_i\}$, learning rate $\gamma$, component weight $\alpha$, number of dimension $K$

**output**: Representation for each node $u \in V$: source embedding $S_u \in \mathbb{R}^K$, target embedding $T_u \in \mathbb{R}^K$, influenceability bias $b_u$, conformity bias $\tilde{b}_u$

Initialize $S_u$ and $T_u$ with uniform distribution $[-\frac{1}{K}, \frac{1}{K})$, $b_u \leftarrow 0$, $\tilde{b}_u \leftarrow 0$; Initialize $P \leftarrow \emptyset$;

for Each $D_i \in A$ do

Extract propagation network $G_i = (V_i, E_i)$;

for Each node $u \in V_i$ do

Generate influence context $C_i^u$ by Algorithm 2;

Insert $(u, C_i^u)$ into $P$;

repeat

for Each $(u, C_i^u) \in P$ do

for Each $v \in C_i^u$ do

Update $S_u, T_v, b_u, \tilde{b}_u$ based on SGD (Eq. 5.5);

Sample a set of negative instances $N$;

for Each $w \in N$ do

Update $S_u, T_w, b_u, \tilde{b}_w$ based on SGD (Eq. 5.5);

until convergence;

return $S_u, T_u, b_u, \tilde{b}_u$ for each user $u$

5.4 Experiments

5.4.1 Experimental Setup

5.4.1.1 Evaluation task

To measure the performance of Inf2vec algorithm, we utilize a commonly used evaluation task [33, 34]. Given a set of users (denoted as seed users) who have already performed a given action, then other users would be affected by these seed users. Therefore, given a test episode $D_i = \{(u, t_i^u)\}$, which contains the chronological order of users, we want to predict the users who will perform the given action in the future.

To make a fair comparison, we only consider the users who have edges with seed users as [33]. Given a set of seed users $S$, a user $v$ is potential adopter if at least one of its friends is in the seed set: $\exists u \in S \land (u, v) \in G$. It is possible that a user would be influenced by multiple seed users. $S_v$ is the set of seed users who may affect user $v$: $\{u | u \in S \land (u, v) \in G\}$. Conventional influence learning methods are based on
Independent Cascade (IC) model. The probability of user $v$ performing the action is calculated by the social influence from his friends:

$$\Pr(v) = 1 - \prod_{u \in S_v} (1 - P_{uv})$$  \hspace{1cm} (5.7)

where $P_{uv}$ is learned transition probability from $u$ to $v$.

Different from the Independent Cascade model, we do not learn the transition probability $P_{uv}$. Therefore, we develop a new approach to calculate the score. In Inf2vec model, we first learn the latent representations of users, then we can utilize them in the evaluation task. Given a potential adopter $v$ and his corresponding seed friends $S_v$, the likelihood that user $v$ being influenced by $S_v$ is defined by

$$\mathcal{F} \left( \{x(u,v), u \in S_v \} \right)$$  \hspace{1cm} (5.8)

where $x(u,v) = S_u \cdot T_v + b_u + \tilde{b}_v$ reflects the likelihood that user $v$ be affected by user $u$ and $\mathcal{F}()$ is an aggregation function to merge the influence from $S_v$. In our experiments, we employ four common aggregate functions as follows:

- **Sum**: linearly combine all the elements, $\mathcal{F}(x_1, x_2, ... x_n) = \sum_{i=1}^{n} x_i$.
- **Ave**: take the average of all the elements, $\mathcal{F}(x_1, x_2, ... x_n) = \frac{\sum_{i=1}^{n} x_i}{n}$.
- **Max**: choose the most significant factor by taking the largest one, $\mathcal{F}(x_1, x_2, ... x_n) = \text{Max}(x_1, x_2, ... x_n)$.
- **Latest**: only take the last element, $\mathcal{F}(x_1, x_2, ... x_n) = x_n$. In fact, in this aggregation, we only consider the first time when user know item $i$ from his friends. When user $v$ logs in the online social website, the latest news of his friends is at front. Therefore, the first seed user that would influence user $v$ is the last one $u_n$.

In this way, we obtain the score for each user. Then we rank all the users by their scores. The ground-truth users are supposed to be ranked higher than the other users. Note that we are able to make fair comparison between IC-based methods and
embedding-based methods, although they exploit two different computing schemes. IC-based methods utilize Equation 5.7, while embedding-based methods employ Equation 5.8. Since we evaluate the performance based on the ranking, rather than the scores, thus the evaluation is fair and reasonable.

To provide thorough analysis, we consider three evaluation metrics.

- **AUC.** Following [33], we utilize the area under curve (AUC) value of Receiver Operating Characteristic (ROC), which plots the true positive rate \( TPR = \frac{TP}{TP+FN} \) against the false positive rate \( FPR = \frac{FP}{FP+TN} \). In our setting, TP denotes the case when an user takes the action and the model also predicts that it takes that action. FP indicates the situation when an user does not take that action, but the model estimates it takes the action. TN is the case when an user does not take that action and the model also predicts it does not take that action. Similarly, FN means the circumstance that an user takes that action, but the model predicts it does not perform that action. Different from previous methods [33,34] that exploit threshold value to make predictions, we utilize the ranking scheme [128] to calculate the AUC value, in which the AUC represents the probability that a randomly selected positive example is accurately ranked than a random selected negative example.

- **MAP.** Since only a very small fraction of test cases are positive (be activated by its neighbors) and most of them are negative (not be activated by its neighbors), the AUC metric is not informative for such imbalanced situation [129]. In addition, we are more interested in the positive examples that users are affected by their friends. To this end, we utilize mean average precision (MAP), which roughly represents the area under precision-recall curve.

- **Pre@N.** We also investigate the precision for top-N predictions. Given a ranking list, we exploit the N users with highest scores as predicted activated users, then we calculate the ratio that predicted activated users are positive examples. In our experiments, we set N as 10, 50 and 100 respectively.
In our experimental setting, we only consider the users who have at least one of its friends has performed that action. In other words, if none of an user’s neighbors is active, then this user is ignored.

5.4.1.2 Evaluated Methods

In the experiments, we evaluate the performance of several approaches as follows.

- **DE**: the degree-based method, the probability of each edge is set as $P_{uv} = \frac{1}{\text{Indegree}(v)}$, where $\text{Indegree}(v)$ is the number of friends of user $v$. This method is widely used in the social influence analysis, such as influence maximization problem [25].

- **ST**: the static model with Maximization Likelihood Estimator method [33]. The probability $P_{uv} = \frac{A_{uv}}{A_u}$, where $A_{uv}$ is the number of action that user $u$ performs before user $v$ and $A_u$ is the number of action that $u$ performs.

- **EM**: the Expectation-Maximization method based on independent cascade model [35], which learns the propagation probability of each edge.

- **Embedded-IC**: the embedded cascade model [37], which is the state-of-the-art approach to learn representation of influence diffusion. Embedded-IC is based on the independent cascade model, and the parameters are inferred by an algorithm similar to [35].

- **FMC**: the factorized Markov chain [1], which factorizes the sequential transition matrix. This method is commonly utilized to learn the Markov chain sequences in various applications, such as user sequential mobility [19] and sequential purchase transaction [1].

- **MF**: the user-user matrix factorization method by Bayesian Personalized Ranking [55]. The entry of the matrix is the frequency that two users take the same action. Otherwise, the entry is 0. Note that this method only reflects the global user similarity.
**Node2vec**: the node2vec model [47], which is the state-of-the-art algorithm for network embedding. Note that node2vec only considers the network structure, without the social influence information.

**Inf2vec-L**: the proposed Inf2vec model, but only with the local social influence component.

**Inf2vec**: the proposed model in Algorithm 3, which considers both the local influence context and global similarity context.

Note that the first 4 methods (DE, ST, EM, and Embedded-IC) are based on IC model. Given a set of seed users $S_v$ for a user $v$, the likelihood score is computed by Eq. 5.7. While the other 5 methods are latent representation models, and the likelihood score is computed by Eq. 5.8.

### 5.4.1.3 Parameter Setting

For the action log $A = \{D_i\}$, we randomly select 80% episodes as training set, 10% as tuning set, and 10% as test set. The default number of dimension is $K = 100$. The length threshold $L$ is set as 50, which means that the size of influence context set is 50. Based on the empirical study on tuning set, we set the default component weight $\alpha = 0.1$. The learning rate $\gamma$ is set as $\gamma = 0.005$.

### 5.4.2 Performance of Methods

#### 5.4.2.1 Effect of Aggregation

To investigate the effect of aggregation function $\mathcal{F}()$ in Equation 5.8, we show the experimental results on Digg and Flickr in Table 5.2 and Table 5.3 respectively. We consider 5 evaluation metrics: $AUC$, $MAP$, $Pre@10$, $Pre@50$, $Pre@100$. For all evaluation metrics, the larger value means that the performance is better.

For Digg dataset, the $Ave$ aggregation achieves the best performance except the $Pre@100$. The highest score of $Pre@100$ belongs to $Latest$. However, the $Ave$ performs better than $Latest$, since it can achieve the highest scores for $AUC$, $MAP$, $Pre@10$ and $Pre@100$. For Flickr, the $Ave$ can obtain the highest values for all evaluation...
metrics. Therefore, in our experiments, we use the \( \text{Ave} \) as the default aggregation function if applicable.

<table>
<thead>
<tr>
<th>Aggregation</th>
<th>( \text{AUC} )</th>
<th>( \text{MAP} )</th>
<th>( \text{Pre} @ 10 )</th>
<th>( \text{Pre} @ 50 )</th>
<th>( \text{Pre} @ 100 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ave</td>
<td>0.8878</td>
<td>0.2762</td>
<td>0.6578</td>
<td>0.4410</td>
<td>0.3736</td>
</tr>
<tr>
<td>Sum</td>
<td>0.8590</td>
<td>0.1311</td>
<td>0.1955</td>
<td>0.1146</td>
<td>0.1127</td>
</tr>
<tr>
<td>Max</td>
<td>0.8838</td>
<td>0.2518</td>
<td>0.6174</td>
<td>0.3899</td>
<td>0.3421</td>
</tr>
<tr>
<td>Latest</td>
<td>0.8874</td>
<td>0.2748</td>
<td>0.6404</td>
<td>0.4402</td>
<td>0.3755</td>
</tr>
</tbody>
</table>

Table 5.2: The performance with different aggregation functions on Digg dataset

<table>
<thead>
<tr>
<th>Aggregation</th>
<th>( \text{AUC} )</th>
<th>( \text{MAP} )</th>
<th>( \text{Pre} @ 10 )</th>
<th>( \text{Pre} @ 50 )</th>
<th>( \text{Pre} @ 100 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ave</td>
<td>0.7782</td>
<td>0.0521</td>
<td>0.1602</td>
<td>0.0979</td>
<td>0.0762</td>
</tr>
<tr>
<td>Sum</td>
<td>0.7521</td>
<td>0.0220</td>
<td>0.0219</td>
<td>0.0320</td>
<td>0.0329</td>
</tr>
<tr>
<td>Max</td>
<td>0.7800</td>
<td>0.0411</td>
<td>0.0918</td>
<td>0.0812</td>
<td>0.0673</td>
</tr>
<tr>
<td>Latest</td>
<td>0.7750</td>
<td>0.0450</td>
<td>0.1241</td>
<td>0.0870</td>
<td>0.0692</td>
</tr>
</tbody>
</table>

Table 5.3: The performance with different aggregation functions on Flickr dataset

5.4.2.2 Predicting the Influenced Users

<table>
<thead>
<tr>
<th>Method</th>
<th>( \text{AUC} )</th>
<th>( \text{MAP} )</th>
<th>( \text{Pre} @ 10 )</th>
<th>( \text{Pre} @ 50 )</th>
<th>( \text{Pre} @ 100 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>DE</td>
<td>0.4144</td>
<td>0.0170</td>
<td>0.0098</td>
<td>0.0101</td>
<td>0.0099</td>
</tr>
<tr>
<td>ST</td>
<td>0.8619</td>
<td>0.1790</td>
<td>0.5221</td>
<td>0.2917</td>
<td>0.2344</td>
</tr>
<tr>
<td>EM</td>
<td>0.8623</td>
<td>0.2071</td>
<td>0.4994</td>
<td>0.3482</td>
<td>0.2959</td>
</tr>
<tr>
<td>Embedded-IC</td>
<td>0.8072</td>
<td>0.1503</td>
<td>0.4721</td>
<td>0.2621</td>
<td>0.2203</td>
</tr>
<tr>
<td>FMC</td>
<td>0.8688</td>
<td>0.1938</td>
<td>0.1924</td>
<td>0.2855</td>
<td>0.2890</td>
</tr>
<tr>
<td>MF</td>
<td>0.8568</td>
<td>0.1708</td>
<td>0.2918</td>
<td>0.2866</td>
<td>0.2627</td>
</tr>
<tr>
<td>Node2vec</td>
<td>0.6448</td>
<td>0.0326</td>
<td>0.0325</td>
<td>0.0530</td>
<td>0.0467</td>
</tr>
<tr>
<td>Inf2vec-L</td>
<td>0.8649</td>
<td>0.1835</td>
<td>0.4842</td>
<td>0.2963</td>
<td>0.2534</td>
</tr>
<tr>
<td>Inf2vec</td>
<td><strong>0.8901</strong></td>
<td><strong>0.2817</strong></td>
<td><strong>0.6457</strong></td>
<td><strong>0.4535</strong></td>
<td><strong>0.3723</strong></td>
</tr>
</tbody>
</table>

Table 5.4: The performance for various methods on Digg dataset

The experimental results of various methods on Digg and Flickr are presented in Table 5.4 and Table 5.5 respectively. In each table, we show the results of five evaluation metrics. We can find that DE always gets the lowest score. The poor performance of DE method indicates that this naive approach is not feasible for learning the diffusion probability. ST directly estimates the diffusion probabilities between edges,
Table 5.5: The performance for various methods on Flickr dataset

and this simple strategy can achieve relatively satisfying performance. Compared to ST, EM utilizes a complex Expectation-Maximization technique to learn diffusion parameters. Although the technique is much more complex and time consuming, the improvement is not significant. The performance of Embedded-IC is worse than ST and EM, which indicates that the Embedded-IC is not suitable for the infection prediction task. One possible reason is that Embedded-IC learn latent representation for each node, but map these representations to probabilities by a Sigmoid function. The Sigmoid mapping function may cause the loss of information.

Overall, the performance of ST, EM and Embedded-IC is not as high as the proposed Inf2vec model. Recall that they directly learn the transition probability of each edge. This fact demonstrates that it is hard to effectively estimate the probability of each edge. This may be caused by the data sparsity, since we do not have sufficient historical action data to exactly learn the transition parameters.

We observe that FMC performs well. This result shows that it is useful to factorize the influence propagation matrix to latent vectors. This also validates the idea of learning representation for each user. MF is also able to achieve good performance, which demonstrates that the similarity of user preference is effective to predict the future adopters. But the result of node2vec is not satisfying because it only considers the social network structure. This result indicates that the network embedding algorithms can not directly be utilized for the social influence embedding.
We can learn that the performance of Inf2vec is better than Inf2vec-L. This fact shows that the global user similarity is helpful to model the social influence embedding. Inf2vec consistently achieves the best performance. This result indicates that Inf2vec model can effectively model the social influence in a low-dimensional space. Overall, the latent representation models are able to obtain better results than the IC-based models, which demonstrates the advantage of social influence embedding over the conventional methods.

5.4.2.3 Computational Cost

Next, we investigate the computing cost for evaluated algorithms. We compare the running time between latent representation methods and IC-based methods. As described in Algorithm 3, the time complexity of Inf2vec is $O(I \cdot |P| \cdot N \cdot K)$. Similarly, the other latent representation methods (FMC, MF, Node2vec, Inf2vec-L) have the same time complexity formula $O(I \cdot |P| \cdot N \cdot K)$. The only difference is that they have different definition of $P$, which denotes the number of training instances. Consequently, we only report the running time of Inf2vec.

Note that the first 4 methods (DE, ST, EM, and Embedded-IC) are based on IC spread model. DE directly utilizes the reciprocal of degree as the diffusion probability (Time complexity is $O(|V|)$, where $|V|$ is the size of node set). ST exploits a simple counting based method to calculate the probability of edges (Time complexity is $O(|S|)$, where $|S|$ is the number of social influence pairs). Both these two methods can compute the diffusion parameters in very short time, and hence their running time will not be included. Both EM [35] and Embedded-IC [37] employ the expectation-maximization framework to learn diffusion parameters, and they are relatively time consuming. Therefore, we report the running time of EM and Embedded-IC in this chapter.

For Inf2vec, we set the number of dimension $K = 100$, and the context size $L = 50$. Given a training instance, 5 negative samples are generated to learn the embedding parameters. For Embedded-IC, we set the number of dimension $K = 25$ following [37]. Empirically, these methods would converge after 5-10 iterations.
The running time of one iteration is stated in Table 5.6. Overall, the running time of Inf2vec is less than EM and Embedded-IC model. The running time of Inf2vec is slightly larger than EM on Digg dataset, because Inf2vec exploits more training instances. In Inf2vec algorithm, we generate more training instances to consider the global similarity of user preference. In other words, we need to train with more instances. However, we can still achieve less computing time. Note that Embedded-IC is the state-of-the-art algorithm to learn embedding of influence. Our proposed algorithm can be completed in shorter time than Embedded-IC, even if the number of dimension of Inf2vec ($K = 100$) is larger than Embedded-IC ($K = 25$).

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Inf2vec</th>
<th>EM</th>
<th>Embedded-IC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digg</td>
<td>312 seconds</td>
<td>272 seconds</td>
<td>575 seconds</td>
</tr>
<tr>
<td>Flickr</td>
<td>544 seconds</td>
<td>1312 seconds</td>
<td>2415 seconds</td>
</tr>
</tbody>
</table>

Table 5.6: The running time of one iteration for various methods on Digg and Flickr dataset

For all the latent representation models, the space complexity is $O(|V| \cdot K)$, because we estimate representation vectors for all nodes. While the space complexity of IC-based methods is $O(\mathcal{E})$, since each edge is associated with a probability. Overall, the space cost is not expensive, and hence we do not conduct experiments to evaluate the memory usage.

5.4.2.4 Effect of Parameters

To investigate the effect of number of dimension $K$, we show the MAP results for different $K$ in Figure 5.6. Generally, the MAP increases with the increase of $K$. The highest MAP on Digg is obtained at $K = 200$, while the MAP on Flickr with $K = 200$ is not the maximum one. In addition, the larger value of $K$ implies that more parameters need to be learned, which leads to higher computing cost. Overall, $K = 100$ can get satisfying performance on both datasets, and the running time is not as large as $K = 200$. Therefore, we set the $K = 100$ by default in our experiments.

To study the influence of context length threshold $L$, we show the MAP results with various $L$ in Figure 5.7. The larger $L$ means that more context nodes are generated
5.5 Conclusion

In this chapter, we develop a latent representation model to learn the social influence propagation. The main contributions of this chapter are summarized as follows:

- We study the social influence embedding problem, which is to represent the influence propagation information into a low dimensional latent space. Different
from previous problems, we attempt to learn the latent representation of each node, instead of learning the propagation probability of each edge.

- In order to learn the representations, we propose a new algorithm Inf2vec, which considers both the local social influence and global user preference similarity. This is the first work to adopt the word2vec technique for social influence.

- We conduct extensive experiments on two real datasets. The empirical results demonstrate that Inf2vec significantly outperforms the baselines.
Chapter 6

Influence Maximization with Novelty Decay

In chapter 5, we study how to represent the influence diffusion information of social networks into a latent space. Next, we will study the social influence from another perspective. Given a social network and its diffusion parameters, we aim at mining useful knowledge from it.

As a fundamental research problem in social networks, influence maximization problem [25] has attracted significant attention. It selects a set of $K$ nodes in order to maximize the propagation of ideas, opinions, etc. in social networks. The influence maximization problem has many real-world applications. For example, a marketing campaign may target a small set of influential individuals and expect that the selected users would generate the largest influence coverage in the market.

It has been observed that repeated exposures of an individual to an idea may have diminishing influence on the individual [52]. We call the phenomenon novelty decay. Intuitively, people are less likely to become spreaders of repeated information. However, none of existing influence maximization studies this phenomenon. Specifically, the conventional independent cascade model can not reflect the impact of novelty decay in social networks. By considering the novelty decay, we can accurately discover influential users on social networks, compared to previous research studies. When the impact of novelty decay is taken into consideration, the influence maximization problem becomes much harder and cannot be easily addressed by conventional solutions. Therefore, we attempt to examine the influence maximization with novelty decay and develop a novel approach to solve this task.
In this chapter, we present the study of influence maximization with novelty decay. We first introduce the effect of novelty decay in influence propagation in Section 6.1. Then, we introduce a greedy algorithm and its optimization to choose the influential users in Section 6.2. Next, we present an efficient algorithm for computing influence spread in Section 6.3. Section 6.4 states the experimental evaluation. Lastly, we conclude this work in Section 6.5.

The notations used in this chapter are summarized in Table 6.1.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G = {V, E}$</td>
<td>Social Network</td>
</tr>
<tr>
<td>$</td>
<td>V</td>
</tr>
<tr>
<td>$S$</td>
<td>A set of users</td>
</tr>
<tr>
<td>$K$</td>
<td>Number of nodes in seed set</td>
</tr>
<tr>
<td>$f(n)$</td>
<td>The novelty decay function</td>
</tr>
<tr>
<td>$P_{uv}$</td>
<td>Probability that $u$ activates $v$</td>
</tr>
<tr>
<td>$T_{uv}$</td>
<td>Expected influencing delay time of edge $(u, v) \in E$</td>
</tr>
<tr>
<td>$P_{con}$</td>
<td>Probability that path $P$ is connected</td>
</tr>
<tr>
<td>$P_{blo}$</td>
<td>Probability that path $P$ is blocked</td>
</tr>
<tr>
<td>$\sigma(S)$</td>
<td>Expected number of nodes affected by $S$</td>
</tr>
<tr>
<td>$\sigma P_S(u)$</td>
<td>Probability that $u$ is affected by $S$</td>
</tr>
<tr>
<td>$PP_{ND}(u, S)$</td>
<td>Propagation paths with novel decay paths that end with $u$</td>
</tr>
<tr>
<td>$PP_{ND\theta,c}(u, S)$</td>
<td>$c$ shortest propagation paths with novel decay having a probability larger than $\theta$</td>
</tr>
<tr>
<td>$PP_{ND\theta,c}(S)$</td>
<td>All $PP_{ND\theta,c}(u, S)$ for $u \in S_p$, where $S_p$ is a set of nodes that are probably to be influenced by $S$</td>
</tr>
</tbody>
</table>

Table 6.1: Notation table

### 6.1 Novelty Decay in Influence Propagation

To explore the factor of novelty decay, we formalize the computation of the novelty decay and further confirm its effect on two publicly available datasets. With the novelty decay function, we proceed to formulate the influence maximization problem with the novelty decay.
6.1.1 Novelty Decay Function

For the sake of clarity, we assume that a user is exposed to an event $n$ times if $n$ friends of the user have been influenced. Formally, let $TP_n$ be the probability that a user is influenced after $n$ friends of the user get influenced, and $p_n$ be the probability that a user gets influenced after the $n^{th}$ friend of the user is influenced. We model the relationship between $TP_n$ and $TP_{n-1}$ in Eq. 6.1.

$$TP_n = TP_{n-1} + (1 - TP_{n-1}) \times p_n \tag{6.1}$$

Then we compute $p_n$ in Eq. 6.2

$$p_n = (TP_n - TP_{n-1})/(1 - TP_{n-1}) \tag{6.2}$$

In order to formalize the novelty decay function, $f(n)$, we isolate the novelty decay factor from $p_n$. Specifically, we compute $f(n) = p_n/p_1$, where $p_1 = TP_1$ is the average probability to be influenced when users get influenced at the first time. We further employ the exponential function, $f(n)=\gamma^{n-1}$, as a general form of the novelty decay function and apply least squares approach to estimate its parameter $\gamma$. We show the development of the novelty decay functions in both Digg and Flickr datasets.

**Digg Dataset** contains news information on Digg (digg.com) [127]. The network has 279,634 nodes and 1,731,658 edges. If user $u$ lists user $v$ as a friend, $u$ can see $v$’s activities. The dataset also lists the Digg-votes, each of which records users’ voting on a particular story and the voting time. It contains 3,018,197 votes from 139,409 distinct users on 3,553 popular stories.

Figure 6.1(a) demonstrates that the voting probability of a user over a story, $TP_n$, approaches a saturation point when a sufficient number of her friends ($n > 25$) have voted for the story. In Figure 6.1(b), the actual novelty diminishes with repeated exposures. As illustrated by the blue line in Figure 6.1(b), the best fitting function is $f(n) = 0.2969^{n-1}$, which leads to the smallest sum of squared errors (SSE=0.1941).

**Flickr Dataset** contains a friendship graph and a list of favorite marking records from Flickr (www.flickr.com) [124], which is a photo sharing social network. If a user
u lists v as its friend, u can see the activity (marking photos as favorites) of v. To study how influence propagates through the Flickr social network, we consider active users (having at least 5 markings) and active photos (having been marked by at least 100 users). There are 222,038 active users connected by 14,727,116 links and 3,125 active photos.

In Figure 6.2(a), the marking probability, $TP(n)$, increases with $n$ in the beginning and then becomes stable around $n > 23$. Figure 6.2(b) confirms the effect of novelty decay and shows the best fitting function $f(n) = 0.8918^{n-1}$ with the smallest SSE (SSE=2.7570).

### 6.1.2 IC Model with Novelty Decay

In the IC model with time delay, every node has two states: active and inactive. It is possible to switch to active state from inactive state, but not vice versa. Each edge is associated with two parameters, namely influence probability $P_{uv}$ and expected influencing delay time $T_{uv}$. We augment the IC model with the novelty decay, denoted as $IC_{ND}$, for each node in a social network.

We show an example of influence propagation with novelty decay in Figure 6.3. Four users are linked and the directed edges indicate the influence of one user over another. Each edge is associated with two values, namely influence probability $P$ and
Chapter 6. Influence Maximization with Novelty Decay

Figure 6.2: Novelty Decay on Flickr.

Figure 6.3: A social network with influence probabilities and delays on the social links.

expected influence delay time $T$. For example, user $V_1$ influences $V_3$ with a probability of 0.7 in 2 time units. Given the seed set $\{V_1, V_2\}$, the probability that $V_3$ is activated by the seed set can be computed regularly without considering the novelty decay, e.g., $0.1 + (1 - 0.1) \times 0.7$, where 0.1 (resp. $(1 - 0.1) \times 0.7$) is the probability that $V_3$ gets activated by $V_2$ (resp. $V_1$). As $V_3$ is influenced by both $V_1$ and $V_2$ probably in a certain order, the novelty decay shall be considered into the influence propagation. Consequently, the probability of $V_3$ being activated by $V_1$ will diminish (less than $(1 - 0.1) \times 0.7$) if $V_2$ first attempts to activate $V_3$.

Given a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, a seed set $S \subseteq \mathcal{V}$ and a novelty decay function $f(n)$, the ICND model works as follows. Let $A_t$ be the set of nodes activated at time $t \geq 0$, and $A_0 = S$. Every node $u \in A_t$ has a single chance to activate its out-neighbors
that are inactive at time \( t + T_{uv} \). Node \( u \) activates \( v \) with the probability \( P_{uv} \times f(n) \), where \( n \) is the number of exposures \( v \) has received. An exposure represents a chance that an active node intends to activate an inactive node. The influence diffusion process stops if and only if there is no any exposure. The number of all activated nodes is denoted as \( \sigma(S) = \sum_{t=0}^{\infty} |A_t| \).

### 6.1.3 Problem Definition and Properties

Based on the proposed \( \text{IC}_{\text{ND}} \) model, we formulate the problem of influence maximization with novelty decay (IMND).

**Definition 6 (Influence Maximization with Novelty Decay)** Given a social network \( \mathcal{G} = (\mathcal{V}, \mathcal{E}) \), a novelty decay function \( f(n) \) and a positive integer \( K \), find a seed set \( S \subseteq \mathcal{V} \) that maximizes the expected number of nodes influenced by \( S \) under the \( \text{IC}_{\text{ND}} \) model.

\[
S = \arg\max_{S \subseteq \mathcal{V}, |S| \leq K} \{\sigma(S)|f(n)\}
\]

The conventional influence maximization (IM) problem has been proved to be \( \text{NP-hard} \) [25]. As it is a special case of IMND with \( f(n) = 1 \) for all \( n \) (i.e., there is no novelty decay), we get the hardness of IMND as follows.

**Proposition 2** The Influence Maximization Problem with Novelty Decay is \( \text{NP-hard} \) for the \( \text{IC}_{\text{ND}} \) model.

Different from the conventional IM problem, the influence score \( \sigma(S) \) is non-monotone and non-submodular in IMND. For the proof purpose, we list the special cases when the novelty decay factor is considered in the example of social network (given \( f(n) = 0.3^{n-1} \)) in Figure 6.3.

**Case 1: Non-monotonicity** suppose that \( S_1 = \{V_1\} \), \( S_2 = \{V_1, V_2\} \) and \( S_3 = \{V_1, V_2, V_3\} \), then \( \sigma(S_1) = 2.7204 \), \( \sigma(S_2) = 2.3757 \) and \( \sigma(S_3) = 3.3 \). Because \( \sigma(S_3) > \sigma(S_1) > \sigma(S_2) \), thus \( \sigma(S) \) is non-monotone.

**Case 2: Non-submodularity** suppose that \( S_1 = \{V_1\} \), \( S_2 = \{V_1, V_2\} \), then \( \sigma(S_1 \cup \{V_3\}) - \sigma(S_1) = 0.3796 \), and \( \sigma(S_2 \cup \{V_3\}) - \sigma(S_2) = 0.9243 \). As \( S_1 \subseteq S_2 \) and
\[ \sigma(S_1 \cup \{V_3\}) - \sigma(S_1) < \sigma(S_2 \cup \{V_3\}) - \sigma(S_2), \] 
\( \sigma(S) \) is non-submodular.

Proposition 3 summarizes the properties.

**Proposition 3** The influence function under \( \text{IC}_{\text{ND}} \) model is neither submodular nor monotone.

### 6.2 Greedy Algorithm and Optimization

Due to the non-monotonicity and non-submodularity of the influence spread function under \( \text{IC}_{\text{ND}} \), the traditional greedy algorithm \cite{25} becomes inapplicable. We resort to the U-Greedy algorithm \cite{54} that is developed for solving profit maximization problems (with the non-monotonicity and submodularity properties) in social networks. We further improve the algorithmic efficiency with an optimization.

#### 6.2.1 R-Greedy Algorithm

The U-Greedy algorithm repeatedly adds nodes of the maximal positive marginal profit, and returns a seed set of any size that results in the maximum profit. Note that the IMND problem seeks for a seed set having not larger than \( K \) nodes (\(|S| \leq K\)). We choose the first \( K \) nodes each of which has the largest marginal influence. Then we select the set of nodes with the maximal influence spread. The adapted algorithm is called the plain restricted greedy algorithm (R-Greedy). Although the size of returned seed set could be smaller than \( K \) in a general case, the situation will seldom occur in practice due to the limited budget (where \( K \) is relatively small compared to the network size).

#### 6.2.2 Dynamic Pruning Optimization

Let \( S_k \ (s_k) \) denote the set of selected seeds (the single node) at round \( k \). The R-Greedy algorithm needs to compute the marginal influence increase for each node \( u \in \mathcal{V}\setminus S_{k-1} \) and is invoked at each iteration of the algorithm to retrieve the \( k^{th} \) node \( s_k \). To achieve better efficiency, we develop an optimization approach, namely dynamic pruning method (DP), that exploits the previous computation of influence spread to
select potential seed nodes in the R-Greedy algorithm. The complete algorithm is shown in Algorithm 4.

**Algorithm 4: R-Greedy Algorithm with DP**

**Input:** $G = (V, E), T_{uv}, P_{uv}, f(.)$ and $K$

**Output:** $S$

1. $S \leftarrow \emptyset$, $S_0 \leftarrow \emptyset$, $\sigma(S) \leftarrow 0$, $\sigma(S_0) \leftarrow 0$, $s \leftarrow NULL$

2. For every $v \in V$, calculate $\sigma(\{v\})$ and insert $(v, \sigma(\{v\}))$ into $Q_0$

3. for $k \leftarrow 1$ to $K$ do

   4. $\text{maxMarInf} \leftarrow -\infty$

   5. for node $u \in V \setminus S_{k-1}$, $\sigma(\{u\}) \geq \text{maxMarInf}$ do

      6. if $u \in Q_{k-1}$, and $(\text{Inf}_{k-1}^u + \sigma(\{s_{k-1}\})) - \sigma(S_{k-1}) < \text{maxMarInf}$ then

         7. Continue

      else

         8. Calculate $\text{Inf}_k^u$ and insert $(u, \text{Inf}_k^u)$ into $Q_k$

         9. if $\text{Inf}_k^u - \sigma(S_{k-1}) > \text{maxMarInf}$ then

            10. $\text{maxMarInf} \leftarrow \text{Inf}_k^u - \sigma(S_{k-1})$;

            11. $s_k \leftarrow u$

      end

   end

12. $S_k \leftarrow S_{k-1} \cup \{s_k\}$;

13. $\sigma(S_k) \leftarrow \sigma(S_{k-1}) + \text{maxMarInf}$

14. $S \leftarrow$ the $S_k$ with maximum $\sigma(S_k)$ from $k = 1$ to $k = K$;

15. **return** $S$

Let $Q_k$ store the checked nodes in round $k$, and elements in $Q_k$ are in the form of tuple $(u, \text{Inf}_k^u)$, where $\text{Inf}_k^u = \sigma((S_{k-1} \cup \{u\})$ means the influence after adding $u$ into selected seed set $S_{k-1}$. In each round, DP checks all the candidate nodes in decreasing order of their influence. It terminates once the influence of individual nodes is below a dynamically maintained threshold, $\text{maxMarInf}$, which records the maximum marginal influence increase in round $k$ (Line 5). Another key optimization technique is that if $u$ has been examined in round $(k-1)$, we derive an upper bound of its marginal influence, i.e., $(\text{Inf}_{k-1}^u + \sigma(\{s_{k-1}\})) - \sigma(S_{k-1})$ (Line 6 and the correctness will be given in Proposition 4). A node will be ignored if its upper bound is less than $\text{maxMarInf}$ (Line 7). Then the algorithm calculates the influence of $(S_{k-1} \cup \{u\})$, and stores the result into $Q_k$ (Line 9). If the marginal influence is larger than $\text{maxMarInf}$, it updates $\text{maxMarInf}$ as well as $s_k$ (Lines 10-12). Finally, we obtain the seed set $S_k$ and its influence spread $\sigma(S_k)$ (Lines 13-14).
As the DP operation prunes nodes whose influence is less than $maxMarInf$, the algorithm still maintains the solution quality of the plain R-Greedy algorithm. We formally prove the property.

**Proposition 4** The DP optimization preserves the solution quality of the R-Greedy algorithm.

**Proof.** The influence function of the \(IC_{ND}\) model satisfies \(\sigma(S_1 \cup S_2) \leq \sigma(S_1) + \sigma(S_2)\), \(S_1, S_2 \subseteq V\). For a node \(u\), its marginal influence \(MarInf^k(u) = \sigma(S_{k-1} \cup \{u\}) - \sigma(S_{k-1}) \leq \sigma(S_{k-1}) + \sigma(\{u\}) - \sigma(S_{k-1}) = \sigma(\{u\})\). If node \(u\) has been checked in prior round, \(MarInf^k(u) = \sigma(S_{k-1} \cup \{u\}) - \sigma(S_{k-1}) = \sigma((S_{k-2} \cup \{s_{k-1}\}) \cup \{u\}) - \sigma(S_{k-1}) = \sigma((S_{k-2} \cup \{u\}) \cup \{s_{k-1}\}) - \sigma(S_{k-1}) \leq Inf_{k-1}^u + \sigma(\{s_{k-1}\}) - \sigma(S_{k-1})\). Therefore, if the upper bound of a node \(u\) (\(\sigma(\{u\})\) or \(Inf_{k-1}^u + \sigma(\{s_{k-1}\}) - \sigma(S_{k-1})\)) is smaller than \(maxMarInf\), it is safe for DP to prune this node.

As we need to check all of the nodes in every round, the time complexity of the R-Greedy algorithm is \(O(K|V|T(\sigma(S)))\), where \(T(\sigma(S))\) is the time for computing \(\sigma(S)\). With the DP optimization, the number of checked nodes in every round is much smaller than \(|V|\).

### 6.3 Algorithms for Computing Influence Spread

With the improved R-Greedy algorithm, the remaining issue is to compute influence spread of seed nodes. For this purpose, we propose a propagation path based algorithm that overcomes the inefficiency of simulation-based techniques.

**6.3.1 Simulation Based Algorithm**

In the \(IC_{ND}\) model, due to the novelty decay, we must consider the order of repeated exposures. This renders inapplicable the existing simulation based algorithm [25], which does not consider such an order. We propose a simulation based algorithm for computing influence spread based on the \(IC_{ND}\) model. We simulate the spreading process of \(IC_{ND}\) model by considering both the spread delay time and the novelty decay effect in the model.
Algorithm 5: Compute $\sigma(S)$ based on Simulation

Input: $G = (V, E)$, $T_{uw}$, $S$, $P_{uw}$ and $f(.)$

Output: $\sigma(S)$

1. Initialize the influenced set $A \leftarrow S$
2. Initialize an array $NE(u) \leftarrow 0$ for $u \in V \setminus S$
3. Initialize a min priority queue $EH$ for potential exposures, and enqueue $E = (0, 1, u)$ for $u \in S$
4. while $EH \neq \emptyset$ do
   5. $E \leftarrow$ dequeue $(T, P, u)$ from $EH$
   6. if $u \notin A$ then
      7. $P \leftarrow P \times f(NE(u))$
      8. draw flag from $\text{Bernoulli}(P)$
      9. if flag = 1 then
         10. $A \leftarrow A \cup \{u\}$
         11. enqueue $E = (T_{uw} + T, P_{uw}, v)$ into $EH$ for every $(u, v) \in E$
      12. else
         13. $NE(u) \leftarrow NE(u) + 1$
   14. return $|A|$

The simulation based algorithm is outlined in Algorithm 5. It computes the influence $\sigma(S)$ for a seed set $S$. It employs the following variables: $NE(u)$ is the number of exposures of $u$; $A$ stores active nodes; $EH$ is a min-priority queue for storing potential exposures, each of which has exposure time $T$, activated probability $P$, and target node $u$, with the exposure time as the key.

Initially, only the nodes in $S$ are active. We then dequeue a potential exposure with minimum exposure time $T$ (line 5). If its target node $u$ is not yet activated, we compute the probability of activating $u$ under the effect of novelty decay $f(NE(u))$, where $NE(u)$ is the number of exposures to $u$ (line 7). If $u$ is activated successfully, then $u$ will try to activate each of its neighbors $v$. The new potential exposure (with exposure time $T_{uw} + T$) is enqueued into $EH$ (lines 8–11). Otherwise, $NE(v)$ is updated (line 13). When no potential exposure exists in $EH$, the algorithm terminates and the number of activated nodes is returned.

The time complexity of Algorithm 5 is $O(|V| + |E|)$. To obtain the influence spread value, a large number of Monte Carlo simulations are conducted. Combined with Algorithm 4, the time complexity of the simulation-based approach is $O(K|V|R(|V| + |E|))$, where $R$ is the number of simulations generally set at 20,000.
6.3.2 Propagation Path Based Algorithm

The simulation-based algorithm is time-consuming and not suitable for large social networks. We develop a propagation path based algorithm to efficiently estimate influence spread.

6.3.2.1 Propagation Path with Novelty Decay

Given a seed set \(S \subseteq V\), the expected influence spread \(\sigma(S) = \sum_{u \in V} AP_S(u)\), where \(AP_S(u)\) is the probability of \(u\) being activated by \(S\). To estimate \(AP_S(u)\), we define a propagation path with novelty decay (PP\textsubscript{ND}) below.

**Definition 7 (Propagation Path with Novelty Decay)** Given a seed set \(S\) and a directed graph \(G = \{V, E\}\), a path \(h = (u_1 \xrightarrow{e_1} u_2 \xrightarrow{e_2} u_3 \cdots \xrightarrow{e_{k-1}} u_k)\) in graph \(G\) is a propagation path with novelty decay (PP\textsubscript{ND}), if and only if \(u_1 \in S\) and \(u_i \notin S\) for \(i \neq 1\), where \(k > 1\).

As a node cannot be activated more than once, a PP\textsubscript{ND} path does not contain duplicate nodes. Its length is \(\text{Len}(h) = \sum_{i=1}^{k-1} T_{e_i}\) while the probability can be computed as \(\prod_{i=1}^{k-1} P(e_i) \times \hat{E}(\tau^h(u_{i+1}))\), where \(\hat{E}(\tau^h(u_{i+1}))\) is the expected novelty decay value for \(h\) on \(u_{i+1}\). Note that \(\hat{E}(\tau^h(u))\) depends on the order of all paths ending at \(u\).

6.3.2.2 Computing \(\hat{E}(\tau^h(u))\)

A PP\textsubscript{ND} path has two states, connected and blocked. The path is connected if it successfully activates \(u_1, \ldots, u_{k-1}\); otherwise it is blocked. The path has a chance to activate its ending node iff it is connected. We denote the connected probability as \(P_{\text{con}} = \prod_{i=1}^{k-2} P(e_i) \times \tau^h(u_{i+1})\), and the blocked probability becomes \(P_{\text{blo}} = 1 - P_{\text{con}}\). If the activation from \(h\) is the \(i^{th}\) exposure for \(u_k\), \(h\) is ranked as \(i^{th}\) among all the paths ending at \(u_k\). We next propose an expected novelty method to compute \(\hat{E}(\tau^h(u))\).

Suppose that \(h_c\) is the \(c^{th}\) shortest PP\textsubscript{ND} path of \(u\), i.e., there are \(c - 1\) paths shorter than \(h_c\). The shorter a path, the earlier it activates its ending node. To compute \(\hat{E}(\tau^{h_c}(u))\), we need to consider all possible combinations of states (connected
or blocked) of $c-1$ paths. For example, if $h_1$ and $h_2$ are the first two shortest paths to activate $u$, the computation of $\hat{E}(\tau^{h_3}(u))$ needs to consider 4 cases: 

$$\hat{E}(\tau^{h_3}(u)) = P_{blo}(h_1) \times P_{blo}(h_2) \times f(1) + P_{blo}(h_1) \times P_{con}(h_2) \times f(2) + P_{con}(h_1) \times P_{blo}(h_2) \times f(2) + P_{con}(h_1) \times P_{con}(h_2) \times f(3).$$

### 6.3.2.3 Finding $PP_{ND}$

For a given seed set $S$, we use $PP_{ND}(u, S)$ to denote all $PP_{ND}$ paths from $S$ to node $u$. It is obvious that each path provides a chance for $S$ to activate $u$. Since the number of paths, $|PP_{ND}(u, S)|$, grows exponentially with the number of seed nodes, finding $PP_{ND}(u, S)$ is computationally expensive for a large $S$. We apply two restrictions to eliminate the $PP_{ND}$ paths that have small influence contribution. First, we prune the paths with probabilities smaller than a specified threshold $\theta > 0$. Second, we retain at most $C$ shortest paths in $PP_{ND}(u, S)$, because a user is unlikely to be influenced after many exposures due to the novelty decay effect. The resulting $PP_{ND}$s are denoted by $PP_{ND\theta,C}(u, S)$.

Finding $PP_{ND\theta,C}(S)$ aims to search at most $C$ shortest paths for each destination node from multi-source nodes with a threshold restriction. This differs from the state-of-the-art algorithms [130, 131] for $K$ shortest path routing because the algorithms focus on a single source and the $\theta$ constraint cannot be easily incorporated. To fill this gap, we develop an adapted Dijkstra (AD) algorithm for finding $PP_{ND\theta,C}(S)$. Like Dijkstra algorithm, AD adopts a greedy search strategy to select the shortest path for the extension. To satisfy the $\theta$ constraint, AD only extends the path meeting the constraint in each iteration. Furthermore, we integrate the computation of path probabilities into AD.

In Algorithm 6, AD starts with initializing $PP_{ND\theta,C}(S)$, $Count(u)$ recording the number of paths on node $u$, and $PH_{con}(u)$ recording $P_{con}$ of the found paths (Lines 1). To implement the greedy search strategy, AD initializes a min priority queue $PH$ for storing the information of paths, each of which has $T$ as the length of time and $path$ for the information of nodes and probabilities on edges (Line 2). At each iteration,
AD chooses the path that has the minimum $T$ in $PH$ to extend (line 4). At lines 5–6, $\mathcal{P}_{wu} \times E(\tau_u(P))$ is the probability $u$ is activated by $w$ on the $\mathbb{PP}_{\text{ND}} P$, then the probability that $u$ is activated by $P$ is $\mathcal{P}^P(u) = \mathcal{P}_{\text{con}} \times \mathcal{P}_{wu} \times E(\tau_u(P))$. If $P$ satisfies the restrictions, we insert it into $\mathbb{PP}_{\text{ND}, \theta, C}(S)$, update $\text{Count}(u)$, $PH_{\text{con}}(u)$, get new paths by extending it, and then insert the new paths into $PH$ (Lines 8–12).

<table>
<thead>
<tr>
<th>Algorithm 6: Computing $\sigma(S)$ based on $\mathbb{PP}_{\text{ND}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> $G = (\mathcal{V}, \mathcal{E})$, $T_{uv}$, $S$, $\mathcal{P}_{uv}$, $f(\cdot)$, $\theta$ and $C$</td>
</tr>
<tr>
<td><strong>Output:</strong> $\sigma(S)$</td>
</tr>
<tr>
<td>1 Initialize $\mathbb{PP}<em>{\text{ND}, \theta, C}(u, S) \leftarrow \emptyset$, $\text{Count}(u) \leftarrow 0$ and $PH</em>{\text{con}}(u) \leftarrow \emptyset$ for $u \in \mathcal{V}$;</td>
</tr>
<tr>
<td>2 Initialize a min priority queue $PH$, and enqueue $P = (0, 1, \text{path} = {u})$ for $u \in S$;</td>
</tr>
<tr>
<td>3 while $PH \neq \emptyset$ do</td>
</tr>
<tr>
<td>4 $P \leftarrow$ dequeue $(T, \mathcal{P}_{\text{con}}, \text{path})$ from $PH$;</td>
</tr>
<tr>
<td>5 compute $\hat{E}(\tau^P(u))$ according to $PH_{\text{con}}(u)$;</td>
</tr>
<tr>
<td>6 $\mathcal{P}^P(u) \leftarrow \mathcal{P}<em>{\text{con}} \times \mathcal{P}</em>{wu} \times \hat{E}(\tau^P(u))$;</td>
</tr>
<tr>
<td>7 if $\text{Count}(u) &lt; C$ and $\mathcal{P}^P(u) &gt; \theta$ and $P$ is loopless then</td>
</tr>
<tr>
<td>8 Insert $\mathcal{P}^P(u)$ into $\mathbb{PP}_{\text{ND}, \theta, C}(u, S)$;</td>
</tr>
<tr>
<td>9 Count$(u) \leftarrow \text{Count}(u) + 1$;</td>
</tr>
<tr>
<td>10 Insert $\mathcal{P}<em>{\text{con}}$ into $PH</em>{\text{con}}(u)$;</td>
</tr>
<tr>
<td>11 $\mathcal{P}_{\text{con}} \leftarrow \mathcal{P}^P(u)$;</td>
</tr>
<tr>
<td>12 enqueue $P = (T + T_{uv}, \mathcal{P}_{\text{con}}, \text{path} \cup {v})$ into $PH$ for $(u, v) \in \mathcal{E}$;</td>
</tr>
<tr>
<td>13 for every $u$ with non-empty $\mathbb{PP}_{\text{ND}, \theta, C}(u, S)$ do</td>
</tr>
<tr>
<td>14 $\mathcal{A}\mathcal{P}<em>S(u) \leftarrow 1 - \prod</em>{P \in \mathbb{PP}_{\text{ND}, \theta, C}(S)} (1 - \mathcal{P}^P(u))$;</td>
</tr>
<tr>
<td>15 $\sigma(S) \leftarrow \sigma(S) + \mathcal{A}\mathcal{P}_S(u)$ ;</td>
</tr>
<tr>
<td>16 return $\sigma(S)$;</td>
</tr>
</tbody>
</table>

6.3.2.4 Computing $\sigma(S)$

After getting $\mathbb{PP}_{\text{ND}, \theta, C}(u, S)$, we compute the activation probability $\mathcal{A}\mathcal{P}_S(u)$ (Line 14). All the paths ending at $u$ are assumed to be independent following the previous work [53, 87]. Finally, the activating probabilities of all nodes are summed into $\sigma(S)$.

Let $N_{\theta, C} = \max_{|S| \leq K}|\mathbb{PP}_{\text{ND}, \theta, C}(S)|$ be the maximum number of paths starting from $S$. The time complexity of finding $\mathbb{PP}_{\text{ND}, \theta, C}(S)$ is $O(N_{\theta, C} \log N_{\theta, C})$. The calculation of $\sigma(S)$ takes $O(N_{\theta, C})$. Hence the total time complexity of Algorithm 6 is $O(N_{\theta, C} \log N_{\theta, C})$. Combining Algorithms 4 and Algorithms 6, the computing complexity is $O(K|\mathcal{V}| N_{\theta, C} \log N_{\theta, C})$. Since $N_{\theta, C} \log N_{\theta, C} \ll R(|\mathcal{V}| + |\mathcal{E}|)$ in practice, the $\mathbb{PP}_{\text{ND}}$ based solution is much faster than the simulation approach.
6.4 Experiments

6.4.1 Experimental Setup

6.4.1.1 Datasets

Apart from Digg and Flickr, two other real-world social networks are used in the experiments. Wiki is a voting network containing the Wikipedia voting data. It has around 7,000 nodes and 103,000 edges. NetPHY is a collection network of papers, extracted from ”Physics” sections in arXiv, and contains around 37,000 nodes and 181,000 edges. Wiki and NetPHY are widely used in the influence maximization problem. The datasets are summarized as follows.

<table>
<thead>
<tr>
<th>Networks</th>
<th>Wiki</th>
<th>NetPHY</th>
<th>Digg</th>
<th>Flickr</th>
</tr>
</thead>
<tbody>
<tr>
<td># nodes</td>
<td>7K</td>
<td>37K</td>
<td>279K</td>
<td>222K</td>
</tr>
<tr>
<td># edges</td>
<td>103K</td>
<td>181K</td>
<td>1.731M</td>
<td>14.727M</td>
</tr>
<tr>
<td>Average degree</td>
<td>29.4</td>
<td>9.8</td>
<td>12.4</td>
<td>132.6</td>
</tr>
</tbody>
</table>

Table 6.2: Statistics of datasets for influence maximization with novelty decay

6.4.1.2 Evaluated Methods

In experiments, we consider two tasks to evaluate the performance of various methods. The first task is to examine the efficiency of dynamic pruning strategy for R-Greedy algorithm, i.e., how to speed up the process of selecting influential nodes one by one. The second task is to investigate the performance of PPAN algorithm, i.e., how to accurately estimate the influence spread of a seed set.

We evaluate the plain R-Greedy algorithm (RGA) and the improved R-Greedy algorithm with DP (Algorithm 4) (RGA-DP). Both algorithms use either the simulation based algorithm (Algorithm 5) (MC) or the propagation path based algorithm (Algorithm 6) (PPAN) to compute influence spread. Therefore, in order to evaluate the performance of dynamic pruning optimization, we consider the following four approaches.

- RGA+MC: the original R-Greedy algorithm and Monto Carlo simulation (Algorithm 5).
RGA-DP+MC: R-Greedy algorithm with DP (Algorithm 4) and Monto Carlo simulation (Algorithm 5)

RGA+PPAN: the original R-Greedy algorithm and the proposed propagation path based algorithm PPAN (Algorithm 6)

RGA-DP+PPAN: R-Greedy algorithm with DP (Algorithm 4) and the proposed propagation path based algorithm PPAN (Algorithm 6)

We are the first one to study influence maximization with novelty decay. Hence, there is no existing baseline. We compare them to two conventional influence maximization algorithms: the classical Monto Carlo simulation based technique CIM-MC [27] and the degree based algorithm DE (choose $K$ nodes with maximum degrees). Although many methods [30, 85, 87, 132] have been proposed to solve the conventional influence maximization problem (without novelty decay effect), all of them focus on developing efficient algorithms that can select influential seed set in short time. In other words, their algorithms are proposed to economically solve the influence maximization problem. Specifically, they cannot obtain better effectiveness compared to the Monto Carlo simulation. Therefore, Monto Carlo simulation (CIM-MC) can provide the benchmark solution for the conventional influence maximization problem. If CIM-MC cannot get good performance for IMND, then other conventional solutions probably also cannot achieve satisfying performance. Consequently, we only examine CIM-MC to study whether the solution of conventional influence maximization can be exploited to address the IMND problem.

In summary, to evaluate the performance of finding influential seed users, we study four methods.

MC: exploit Monto Carlo simulation (Algorithm 5) to compute influence spread and R-Greedy algorithm with DP (Algorithm 4) to select seed nodes.

PPAN: use propagation path based algorithm PPAN (Algorithm 6) to estimate influence spread and R-Greedy algorithm with DP (Algorithm 4) to select seed nodes.
Chapter 6. Influence Maximization with Novelty Decay


- DE: a naive baseline, which always chooses the top-\(K\) nodes with maximum degrees as the seed set.

For each method, a set of seed nodes can be obtained. Then the MC method is employed to compute the influence spread of the seed set returned by each method. In this way, we can fairly compare the effectiveness of different methods.

All methods are implemented in C++. All experiments are conducted on a windows server with 6-core Intel(R) Xeon (R), 2.66 GHz CPU and 24 GB memory.

6.4.1.3 Parameter Setting

We set the influence probability \(P_{uv}\) of \(u\) on \(v\) by the weighted cascade policy [31, 53], i.e., \(P_{uv} = \frac{1}{\text{indegree}(v)}\), where \(\text{indegree}(v)\) is indegree of node \(v\). The expected influencing delay time \(T_{uv}\) of edge \(uv\) follows the geometric delay distribution [31]. The parameter for geometric distribution is set at \(5/(\text{outdegree}(v)+5)\). For simplicity, the maximum value of \(T_{uv}\) is 15. If the generated delay time \(T_{uv} > 15\), \(T_{uv}\) is reset as a random integer from 1 to 15. We also try other distributions for \(T_{uv}\) including poisson distribution and uniform distribution, and the experiment results exhibit similar trend for the evaluated techniques. Empirically, threshold \(\theta\) is set at \(\theta = 0.001\) and number of paths \(C\) is set at \(C = 5\), which achieves a satisfying trade off between influence spread and running time in our experiment.

For Digg and Flickr, we use their actual novelty decay function \(f(n)\) aforementioned. For wiki and NetPHY, we utilize the default exponential function \(f(n) = 0.3^{n-1}\) and further examine different \(\gamma\).

6.4.2 Selecting the R-Greedy Algorithms

We investigate the effect of DP optimization on the R-Greedy algorithm. Both MC and PPAN are used to compute influence spread. We terminate MC if it runs over
five days. Figure 6.4 shows the results on both Wiki and NetPHY since MC cannot finish for the large Digg and Flickr networks. Figure 6.4 demonstrates that RGA-DP is 1-2 orders of magnitude faster than RGA on both datasets. Hence we use RGA-DP for the rest of experiment if applicable. Note that DP optimization does not affect the solution quality of the RGA. Thus we do not compare their influence spread.

Figure 6.4: Performance of RGA and RGA-DP based algorithms.

6.4.3 Performance of the Algorithms for Solving IMND

We evaluate the influence spread and running time of all the methods over four datasets. Since MC and CIM-MC are extremely computationally expensive, we cannot get their results on Digg and Flickr.

6.4.3.1 Influence Spread

The influence spread reflects the quality of selected seed set. Figure 6.5 illustrates the influence spread for various $K$ values. PPAN obtains similar influence spread as MC. It indicates that PPAN is an effective approximate method. Both CIM-MC and DE achieve consistently lower influence spread than do PPAN and MC. This demonstrates that conventional influence maximization techniques are not effective for solving the new IMND problem.
Figure 6.5: The influence spread of methods.

6.4.3.2 Running Time

Figure 6.6 shows that the running time grows when $K$ increases for the studied methods. PPAN is several orders of magnitude more efficient than MC. In summary, the combination of Algorithms 1 and 2 provides the best solution to the IMND problem.

6.4.3.3 Effect of $f(n)$

We further investigate the impact of $\gamma$ on Wiki and NetPHY. Figure 6.7 shows the influence spread for various $\gamma$ with $K = 50$. We can observe that the influence spread grows as $\gamma$ increases. This is due to the fact that if $\gamma$ is small, the spread probability from one node to another is declined. Hence, the expected number of activated nodes becomes smaller. Additionally, the results of PPAN are consistently similar to that of MC for different $\gamma$. This verifies that PPAN is an effective approximate method for
Figure 6.6: The running time of methods.

different $f(n)$.

Figure 6.7: The results of influence spread for different $\gamma$. 
6.5 Conclusion

We summarize the main contributions of this chapter as follows.

- We investigate the effect of novelty decay in real social networks. With the augmented independent cascade propagation model, we formulate the problem of influence maximization with novelty decay. To the best of our knowledge, the novelty decay phenomenon has not been considered by existing studies on the influence maximization problem.

- To solve the problem, we employ the R-Greedy algorithm and improve its performance through the dynamic pruning strategy. In addition, we propose an influence propagation path based algorithm to efficiently compute influence spread of seed nodes.

- We conduct experiments on 4 real social networks. The results can demonstrate the effectiveness and efficiency of our proposed algorithms.

It is worth noting the relationship between the social influence embedding (Chapter 5) and influence maximization with novelty decay (Chapter 6). Both of them are related to social influence propagation. However, they are two different research problems. First, social influence embedding is a machine learning problem, while IMND is a discrete optimization problem. Second, the inputs of these two problems are dissimilar. For the social influence embedding problem, given an action log and a social network, we attempt to learn diffusion parameters for the given social network. Meanwhile, for IMND, we are given a social network with diffusion parameters and novelty decay function. The diffusion parameters are assumed to be known by us. Lastly, the algorithms for them are distinct. The solution to social influence embedding is latent representation model, while the approach to IMND is a graph path based method. In spite of the differences, these two problems are closely connected to each other as described in Chapter 1.
Chapter 7

Conclusions and future work

In this chapter, we first summarize the previous chapters to highlight main contributions of this thesis. Then, we discuss some promising directions of future work.

7.1 Conclusions

Nowadays, sequential mobility behaviors and social influence propagation analysis have attracted extensive research attention. The main challenge of these problems is how to effectively model the sequential information. We resort to the latent representation models, which have been adopted for various applications. In this thesis, we develop three latent embedding approaches to model the user sequential mobility and social influence propagation. In addition, we also investigate the effect of novelty decay in influence propagation, and further propose algorithms to identify a set of influential users. We summarize the research studies as follows.

- We propose some novel research problems related to sequential information. For user mobility, we study the recommending successive POIs for a given user and predicting potential visitors for a given POI. For social influence propagation, we examine a new task that embed social influence into a latent low-dimensional space. Furthermore, we investigate the novelty decline effect in social influence propagation, which has not been studied before.

- We develop several effective new techniques to explore and exploit the sequential information in various tasks. For user check-in transitions, we propose a novel
pair-wise ranking metric embedding (PRME), which is the first one to exploit metric embedding for user mobility behaviors. Moreover, we develop a POI2Vec method, which is the first one to incorporate geographical influence into the word2vec framework. For social influence propagation, we invent a novel Inf2vec model, which can jointly consider the spread of social influence and similarity of user preference. In addition, to estimate the influence spread with novelty decay, we propose a propagation path based approach (PPAN).

- We conduct extensive experiments on real datasets. The experimental results demonstrate the effectiveness of our proposed solutions. Based on Foursquare and Gowalla datasets, we study the next POI recommendation and future user prediction. Based on several real social networks, especially the Digg and Flickr datasets, we investigate social influence embedding task and influence maximization with novelty decay. The results on real datasets imply that our models can be used in real world to provide better services for users.

## 7.2 Future Work

Based on our current studies, there are a lot of promising directions for future research. Here, we present few of them as follows.

(i) We can consider more information for the future visitor prediction problem. The POI recommendation problem has been extensively studied, but the visitor prediction problem got much less research attention. Because users’ movements are influenced by multiple factors, we can consider other information. First, we can investigate the time-aware visitor prediction problem. Temporal information, such as weekday/weekend and day/night, has been proven to be effective for POI recommendation tasks [9, 76, 77]. Users have different preference patterns at different time. Given a POI and current time (e.g., 7 pm.), we want to predict who will visit this POI in the next several hours. Second, it is interesting to incorporate the effect of social influence. People’s mobility behaviors would be
affected by their friends [8,64]. We can additionally exploit the social influence to enhance the accuracy of finding potential visitors.

(ii) The method of building binary tree for POI2Vec is not tied to spatial items and can be adopted for other applications, such as using the taxonomy of items for online education [133] or next product recommendation [98]. In fact, we have studied how to exploit the hierarchical structure of course syllabus to learn latent similarity from MOOC (Massive Open Online Course) data. Since the syllabus is carefully designed and organized by instructors, and thus contains informative relations between knowledge points. Naturally, a syllabus can be viewed as a tree structure with one or more children nodes. We have collected textual data of two online Moocs, including their syllabuses [133,134]. By using the syllabus, we develop a novel neural language model, which can achieve satisfying results on several tasks. In addition, the category hierarchy of products is helpful to recommend new products for users [135], especially when the data is sparse. This is useful to solve the cold start problem. It’s natural to build a hierarchical tree based on the given taxonomy. Consequently, we can incorporate the product taxonomy into a latent representation model.

(iii) We can investigate the topical features for social influence embedding models. Users may exhibit different influence on different topics [125]. Barbieri et al. [34] consider the topic information in influence propagation model. They exploit an EM technique to learn the diffusion parameters. However, the proposed algorithm needs to estimate too many parameters and fails to handle the data sparsity. Thus, we can develop some latent representation approaches to model the topic-aware influence propagation. For instance, in order to reflect the topical information, each user would be represented by multiple vectors rather than one single vector.

(iv) Recently, deep learning techniques, such as Recurrent Neural Networks (RNN), are widely utilized in many tasks, especially in natural languages [136,137]. RNN
is designed to make use of sequential information by neural network structure. Comparing with conventional methods, RNN has shown promising performance in next POI prediction [76]. Temporal and spatial context have been considered in RNN. But we can adapt RNN to exploit more features to improve the prediction performance. Furthermore, to the best of our knowledge, RNN has not been adopted to model the social influence propagation. Therefore, it would be interesting to exploit RNN to learn information spread in social networks.

(v) We can employ the proposed algorithms for other applications, such as water distribution network, which consists of engineered hydraulic and hydrologic components. Generally, a water distribution network can be viewed as a directed graph. There are many junctions (nodes) connected by pipes (edges) in networks. Leskovec et al. [27] examine the social networks and the water networks and point out that the seemingly different networks actually share a similar structure. In water distribution networks, the contaminant would spread from one position to other positions, which is similar to the influence spread in social graphs. Therefore, the algorithms in social networks can be applied to water networks. First, we can study the sensor placement problem in water networks. Sensors are placed in water networks to effectively detect contamination [138,139]. However, due to the limited budget, we can only place K sensors in the water supply network. We aim at finding K positions to put sensors such that the benefit is maximized. Second, we can embed a water supply network into a low-dimensional latent space to capture historical contamination propagation data. This would be beneficial to predict the pollution spread in water distribution systems.
References


travel sequences from gps trajectories,” in Proceedings of the 18th international

[3] T. Mikolov and J. Dean, “Distributed representations of words and phrases and


geographical and social influence in location-based social networks,” in AAAI,

[9] Q. Yuan, G. Cong, Z. Ma, A. Sun, and N. M. Thalmann, “Time-aware point-

modeling and matrix factorization for point-of-interest recommendation,”


REFERENCES


REFERENCES


REFERENCES


REFERENCES


Author’s Publications


[6] Zhuoxuan Jiang, Shanshan Feng, Gao Cong, Chunyan Miao, Xiaoming Li: A Novel Cascade Model for Learning Latent Similarity from Heterogeneous Se-