A Study of Recommendation Systems with Temporal and Geographical Information

時間的・地理的情報を用いた推薦システムに関する研究

February, 2024

Fan MO
バク　ボン

A Study of Recommendation Systems with Temporal and Geographical Information

時間的・地理的情報を用いた推薦システムに関する研究

February, 2024

Waseda University Graduate School of Fundamental Science and Engineering

Department of Computer Science and Communications Engineering,
Research on Parallel and Distributed Architecture

Fan MO
バク　ボン

Abstract

In the age of big data, the information overload problem costs users much time and energy to obtain useful information. Recommendation systems, as an effective tool to alleviate information overload, are receiving increasing attention. Recommendation systems intend to improve user satisfaction. Therefore, in industry as well as in academia, researchers have studied and proposed a variety of metrics to evaluate user satisfaction. The effectiveness of recommendation, measured by recall and precision ratios, is considered the most responsive to user satisfaction because high effectiveness indicates the system is better able to predict user interest. Adopting side information is a promising and worthwhile approach to improving recommendation accuracy. This thesis targets two side information- time and geographical information- because time and geographical information are common in recommendation systems, modeling them to improve performance and leaves other attributes such as category as future work. Previous works attempted to model geographical information as distance and time information as a sequence. However, previous work still suffers from insufficient use of side information.

This thesis aims to improve recommendation accuracy by fully using temporal and geographical information. This thesis first focuses on accelerating model updates to achieve real-time periodic recommendation systems. If the periodic updates can be implemented, the model can capture changes in user behavior over time, improving the accuracy of behavior estimation. This thesis uses advertisement (ad) recommendations as an example of applying the technique. In recent years, the market for ad recommendations has been growing. Advertisement recommendation has become an important service to help users mine their interests. For an ad recommendation system, handling time information is important, as user interests constantly change over time. This requires ad recommendation optimization algorithms to be frequently updated to accommodate the user's latest interests. To achieve that, this thesis models temporal information to propose a real-time periodic recommendation technique. The technique adopts a quantum-inspired computer, specifically, Fujitsu digital annealer, to accelerate the optimization process. Experiments on the real Geniee dataset confirmed that the proposed real-time periodic technique achieved a 35.86% improvement in ad recommendation compared with the

state-of-the-art XGBoost+GA baseline.

After that, this thesis works on modeling geographical information to improve recommendation accuracy. A typical application of geographical information is the Point-of-interest (POI) recommendation. This thesis, therefore, uses the POI recommendation as an example to illustrate our modeling. As an important application in location-based social networks, POI recommendations help users filter massive amounts of information and make decisions. POIs have a special attribute- geographical information. To handle the geographical information, this thesis starts by mining users' active areas and transforming the geographical information into the proposed "active area neighbor," which incorporates and extends the definition of neighbor in graph convolution network (GCN) models for POI recommendation, where normally, "neighbor" is defined as check-ins in a GCN model. Then, this thesis designs the user node for aggregating information from check-ins and the proposed active area neighbors. The active area neighbors-based technique improved the $Recall@5$ from 0.0788 to 0.0815 on the Gowalla dataset and from 0.0453 to 0.0469 on the Yelp dataset compared with the state-of-the-art LightGCN.

Then, this thesis focuses on proposing a methodology to simultaneously combine temporal and geographic information. The thesis first proposes a new time-aware GCN model to mine users' time-based high-order connectivity. Specifically, this thesis divides the 24 hours of the day into multiple time slots, generating a subgraph for each time slot and making the target user aggregate information from newly defined time-based high-order connectivity. Time-based high-order connectivity refers to the relationship between indirect neighbors with similar preferences in the same time slots. After modeling the time information, this thesis integrates the active area neighbor-based technique for modeling geographic information proposed in the previous paragraph. Experimental on real datasets confirm that our model further improved $Recall@5$ to 0.0874 on the Gowalla dataset and from 0.0360 to 0.0388 on the New York dataset compared with state-of-the-art GCN-based models after mining abundant time information.

This thesis models the time and geographical information as auxiliaries to be combined into recommendation systems. After constructing the models, this thesis conducted experiments on real datasets, validating the effectiveness of the proposed techniques.

Acknowledgments

I want to thank Prof. Hayato Yamana first. Prof. Yamana gave me a lot of care and advice during my doctorate course, teaching me how to start from a survey to forming my own ideas, how to conduct experiments, and how to describe my thoughts and experimental results in the most straightforward tone from the readers' point of view. I still remember what Prof. Yamana said about the importance of "thinking differently," which gave me a lot of encouragement to conduct my brainstorming. The presentation skills I learned from my professor helped me to be more courageous in attending international conferences and communicating my ideas clearly. Besides, thank you for the opportunity to conduct collaborative research with companies, which gave me a platform to practice further and work on my thoughts. Such an experience would be very beneficial to me.

Next, I would like to offer my respects and thanks to Prof. Tetsuya Sakai and Prof. Kasai. Prof. Sakai gave me a lot of helpful advice on refining my idea and improving the presentation of my paper. Thoughtful suggestions and comments have significantly boosted my research and studies. Prof. Kasai also provided a lot of valuable advice to standardize the writing of notations, improving the quality of the thesis while providing thought-provoking comments for exploring future research directions. I have significantly benefited from the help.

Mr. Tsuneo Matsumoto, Mr. Nao Fukushima, and Ms. Fuyuko Kido also gave me a lot of help and support. Thank you to them for injecting new exploration directions for my research on applications of recommendation systems. Combining the recommendation system with the IUI domain made me think about the recommendation systems in a unique manner. Thank you to Mr. Matsumoto for providing me with a lot of introductions and explanations on legal regulations and economics. Without the support and effort, the study could not have been conducted.

Next, I would like to thank all the lab members of Yamana lab. Mr. Takuya Suzuki has

contributed significantly to maintaining the server and solving problems in the area of life. Mr. Huida Jiao and Mr. Shun Morisawa were instrumental in making the experimental process run smoothly. In some of my work and papers, I listed them as co-authors. Thanks to Mr. Xin Fan, Mr. Chongxian Chen, and Mr. Changhao Bai for discussing with me to improve my proposed method.

Finally, I would like to thank my family and friends. Thank you for your continued help and support in my daily life.

Fan Mo, 2023/12

Contents

List of Figures

List of Tables

1 Introduction

1.1 Background

Recommendation systems have gained increasing attention for their ability to alleviate the information overload problem. Information overload is described as making it difficult for users to find their real needs in the face of large amounts of data. The issue of information overload seriously affects user experience and wastes user time in the age of big data. If recommendation systems can accurately predict and recommend user preferences, it would significantly increase user satisfaction because accurate recommendations can help users filter out the massive amount of information. Thus, most researchers [3] [5] [11] [61] cite improving accuracy as a primary research goal for recommendation systems. Based on related works [30] [39], adopting side information like time and geographical information and integrating the information into recommendation generation models in a potentially effective manner to improve accuracy.

Ye et al. [30] pioneered an attempt to use power-low distribution to model the geographical distance between two POIs. Inspired by Ye et al., in recent years, deep learning models [3] [4] also embedded distance-based geographic information to improve the model representation ability. However, neglecting the area information prevents further model performance improvement because of the geographical continuity. i.e., multiple POIs can form a geographical area. In addition to geographic information, time information is an essential side information in recommendation systems. After Yuan et al.'s [39] first attempt to use time information to describe cosine similarity among users, the adoption of time information has received continued attention from researchers. In recent years, the effort to model time information into sequences and combine it with sequence mining techniques to drive model accuracy is in the dominant position and widely used. Zhang et al. [4] adopted two long short-term memory (LSTM) networks to represent the arrival and departure times of POIs. However, the technique still suffers from insufficient use of time information. Users' time-based preferences can be divided

into time slots for better representation. For instance, two users who prefer Supermarket A may not have a time-based high-order relationship based on differences in their visit time slots.

In this thesis, we target to fully use two attributes- temporal and geographical information, because temporal and geographical information is common in recommendation systems [3] [4] [39] [40]. We model these two attributes as instances and leave other attributes like categorical information as future work.

1.2 Objective and Goal

Our research goal is to improve recommendation accuracy by modeling temporal and geographical information. We first focus on shortening model update time to achieve real-time periodic recommendation systems. User preferences change constantly over time. Once we can implement periodic updates, the accuracy of estimating user behavior can be improved because the model can capture changes in user behavior in time. We adopt advertisement (ad) recommendations as an example to apply the technique. With the advent of the big data era, ad recommendation has been gradually integrated into people's lives as an essential manner of data filtering. In ad recommendation, a demand-side platform (DSP) needs to recommend ads with a high probability of being clicked by the target user under specific constraints called recommendation optimization problems (for example, 1,000 ads for category A and 5,000 ads for category B). Accurate ad delivery reduces information overload problems for users and generates a high conversion rate for DSP. However, the fickleness of user interests over time makes it difficult for recommendation systems to capture users' latest interests, preventing further improvement in accuracy. Frequent model update is an important method to break down the barriers. However, ad recommendation optimization is an NP-hard problem [81], which is difficult to implement on a common computer. In this thesis, we propose to use digital annealers (DA) [1], which are quantum-inspired annealing computers, to capture the changes in user interests and realize real-time periodic recommendations.

We then work on modeling geographical information to improve recommendation accuracy. Geographical information contains rich user behavioral characteristics. Our study is based on the intuition that users in the same geographical area tend to have similar check-in behaviors. A typical application of geographical information is the Point-of-interest (POI) recommendation. Thus, the thesis explores POI recommendations as an example of applied geographical information. Personalized POI recommendation systems recommend the target user's unvisited POIs, which are matched to his/her preference by analyzing check-in history. In recent years, open-source datasets from location-based social networks (LBSNs), such as Gowalla[1] and Yelp[2], allow users to share their check-in experience, making a detailed analysis of users' behavior and provision of better recommendation services possible. POIs have an intrinsic attribute- geographical information, which makes them different from recommending other items like movies and music. To model the geographical information, previous research made some attempts [27] [30]. With the development of deep learning techniques, especially for graph convolution networks (GCN), Chang et al. [3] pioneered the modeling of geographical information and integrated it into GCN. Based on Chang et al.'s work [3], we simplify the model design and emphasize the geographical continuity of POIs. i.e., Multiple POIs can comprise an entire area, which is essential for modeling and designing methodologies to integrate geographical information into GCN. We first mine users' active areas and transform the geographical information into the proposed "active area neighbor," Then, we design the user node to aggregate information from both check-ins and the proposed active area neighbors to generate the final representations (embeddings).

The last goal is to combine temporal and geographical information to co-complement the deep learning models. When combining multiple side information, it is essential to consider the harmonization between the information. We begin by modeling the time information. In deep learning models, sequence-based techniques [3] [4] [42] have

[1] http://snap.stanford.edu/data/loc-gowalla.html
[2] https://www.yelp.com/dataset_challenge

received a great deal of attention. Inspired by the famous word2vec framework [43], previous works [42] [4] proposed to utilize a sequence-based model to capture the users' temporal interests. However, simply adopting time information by modeling users' check-in sequences is insufficient and ignores users' time-based high-order connectivity. Note that time-based high-order connectivity refers to the relationship between indirect neighbors with similar preferences in the same time slot. The time slots-based technique allows for more granular representations of user interests. For example, Assume the target user and user A usually go to the supermarket after breakfast. However, user B prefers to go to the supermarket in the evening. In this case, even though user B shares the same preference as the target user, the system categorizes user A as a time-based high-order neighbor because they have a similar time preference. Therefore, user B is filtered out even if user B has the same preference. After modeling the temporal information, we use the active area-based approach mentioned in the previous paragraph to model the geographic information, combining both time and geographical information into the deep learning model. In a graph convolution network, for a target user node, we aim to propose a novel methodology that makes time information to control the graph structure division, i.e., from which nodes to aggregate information while setting geographical information to control the amount of aggregated information, i.e., how much information is aggregated.

1.3 Challenges

1.3.1 Acceleration of Recommendation Optimization (Challenge 1)

Recommendation optimization - that is, improving accuracy while considering constraints such as budget or cost-per-click (CPC) - is a complex problem in recommendation systems. Achieving high accuracy and quick optimization can be contradictory and present an NP-hard complexity, making the periodic updates of the recommendation optimization problem remain an open question. Although previous works [7] [20] attempted to accelerate optimization based on heuristics and linear

programming, the latency is still too high when dealing with real-time recommendation optimization problems. A training process that fits the real-time task needs to be explored to capture the user's preferences change because user preferences change rapidly over time. Even if the system captures users' preferences, applying the predictions over a long time without updating the model is difficult. Besides, even though we optimize the problem once, the optimized result cannot be applied to the real recommendation system for a long time because the preconditions for the optimization vary over time, resulting in a decrease in the effectiveness of the optimization result. Thus, real-time periodic optimization is the key to breaking down the barriers. In the proposed periodic training framework, we embed a Fujitsu digital annealer (DA), a quantum-inspired annealing computer, to accelerate the recommendation optimization. DA itself is fast in dealing with optimization problems. However, preprocessing the input data for DA is time-consuming, i.e., expanding the objective function polynomials and organizing the coefficients, which is a bottleneck to using DA in real-time periodic recommendation systems. This thesis proposes an element-based method to quickly derive the inputs of DA directly.

1.3.2 Insufficient Use of Side Information in Recommendation Systems

This section describes the development of side information (geographical and time information) in POI recommendation systems, as well as the challenges and problems.

Insufficient Use of Geographical Information (Challenge 2)

Geographical information, as a latent attribute of POI, holds the potential to improve recommendation accuracy. However, how to fully use geographical information remains an open and challenging question. Previous research ignored that POIs have unique geographical continuity, i.e., multiple POIs can form a geographical area. Simply modeling the geography as distance [30] [3] in a deep learning model is inadequate, causing loss of area information and thus preventing further enhancement of model

performance. Besides, in a deep learning-based recommendation system [3], multiple trainable embeddings used to integrate geographical information can significantly increase the number of parameters, making model training difficult and decreasing the utility in practice. Thus, mining users' geographical areas in a lightweight manner (without increasing the number of trainable parameters) is essential in deep learning-based recommendation systems.

Insufficient Use of Temporal Information (Challenge 3)

Time information is generated when a user interacts with a POI. In addition to geographical information, time information is also utilized in models as important side information to improve recommendation performance. However, methodologies for modeling temporal information are still in the exploratory phase. In deep learning models, simply modeling time information by sorting user check-ins in chronological order (sequence) [39] [42] [4] cannot fully exploit collaborative signals in time information, which is insufficient. Time information can be divided into time slots to represent the users' preferences during a certain period of time. For example, even two users who like Supermarket A may be calculated as having no time-based high-order relationship due to the differences in visit time slots. In this thesis, we adopt a subgraph technique to divide the 24 hours of the day into multiple time slots and generate one subgraph for each time slot. In the deep learning model, users only aggregate information from the nodes in the same subgraph. In the proposed model, an edge propagation module is proposed to adjust edge affiliation in subgraphs, where edges represent check-ins, to propagate the user's time-based preference to multiple time slots (subgraphs).

1.4 Contributions

This section describes the contributions of this thesis. The contribution of this thesis is three-fold.

Contribution 1. We are the first to adopt Fujitsu digital annealers (DA) to accelerate advertisement recommendation, along with a corresponding methodology to compute DA

inputs element by element and realize periodic recommendation optimization.

Contribution 2. Proposal of active area neighbor to model geographical information in the graph convolution network.

Contribution 3. Proposal of subgraph (time slot) and edge (check-in) propagation-based technique to model time information in the graph convolution network.

The last part of the section introduces the details of the contribution and how to deal with the corresponding problems and challenges.

Contribution 1. Adoption of digital annealers (DA) to accelerate advertisement recommendation and realize periodic recommendation optimization

Low-speed optimization limits the ability of recommendation models to capture the changes in user interest over time. To solve the problem, we first propose a periodic recommendation optimization framework. i.e., the model is periodically retained. The proposed real-time recommendation system divides users' behaviors into three stages. 1) We collect users' data when they visit the websites. 2) we use the collected data to train and optimize the prediction model in the training stage. 3) In the third step, we return the optimized result to users and start the next model cycle. Then, a digital annealer (DA) is adopted for acceleration. As a quantum-inspired computer, Fujitsu DA can only support the inputs in a quadratic unconstrained binary optimization (QUBO) model [1]. How to cast a recommendation optimization problem to a QUBO model quickly and thus can be input to the DA to achieve periodic optimization is a challenging task. Specifically, we analyze the objective function of the recommendation optimization problem, decomposing the objective function to organize the binomial, monomial, and constant terms. These three terms correspond exactly to the binomial, monomial, and constant terms in the QUBO model. Thus, the objective function can be transformed into a QUBO model. DA has n units. To assign DA units to users, we propose a novel element-based methodology to derive the inputs of DA directly: weight matrix $\boldsymbol{W} \in \mathbb{R}^{n\times n}$, vector $\boldsymbol{b} \in \mathbb{R}^{1\times n}$, and constant $con \in \mathbb{R}^{1\times 1}$. The methodology allows for fast inference of DA inputs

and be applied to real-time recommendation systems. The adoption of DA with the proposed training technique on advertisement (ad) recommendations results in improved accuracy from 0.3703 to 0.5080 (37.19%) and acceleration of 10.6 times compared to a genetic algorithm-based optimization technique. Note that the realization of acceleration is contributed by the proposed fast DA input derivation methodology and the DA's ability to solve optimization problems. Although DA itself is fast in dealing with optimization problems, preprocessing DA input based on polynomial expansion[3] is time-consuming. In the thesis, the proposed method avoids the use of polynomial expansion with the time complexity $O(|T|^2)$, where $|T|$ indicates the number of terms in the objective function. Instead, the proposed methodology directly derives the DA inputs element by element with time complexity $O(|U|n)$. $|U|$ indicates number of users, a real number much smaller than the DA units n. In addition, in the objective function of DA, $|T|$ is larger than n^2. The method provides a new idea to calculate DA inputs without polynomial expansion, not only for real-time recommendation systems but also for other optimization tasks that require transforming DA inputs.

Contribution 2. Proposal of active area neighbor to model geographical information in the graph convolution network.

To make full use of geographical information in a lightweight manner, we start by exploring the user active areas because POIs are geographically contiguous and thus form areas. To achieve the goal, for each user, we cluster the POIs that he/she visited to extract the user's active areas. Note that a user may have multiple active areas among cities. The users whose active areas are close (at least one pair of active regions is less than λ km apart) are defined as active area neighbors. Then, we incorporate newly defined neighbors into deep learning, specifically a graph convolution network (GCN), to improve model representation. In a GCN-based recommendation system, a user's neighbors are described as checked items. We extend the traditional definition of neighbor to active area neighbor. As a result, we can enhance a GCN model by adopting geographical information, which

[3] https://www.sympy.org/en/index.html

can extract high-order connectivity over collaborative filtering information. Note that the technique does not cause any increase in trainable parameters and keeps the model easy to train. In the related GCN-based recommendation system [3], modeling geographical information adds additional geography-based embeddings (trainable parameters), making the model size twice as large and reducing practicality. In contrast, extending the neighbor definition in a lightweight manner is valuable and challenging. Experiments on real dataset confirm that the proposed method improves $Recall@5\ from$ 0.0788 to 0.0815 on the Gowalla dataset and from 0.0453 to 0.0469 on the Yelp dataset compared with state-of-the-art LightGCN model.

Contribution 3. Proposal of subgraph (time slot) and edge (check-in) propagation-based technique to model time information in the graph convolution network.

We analyze and model the time information in our third contribution to train more accurate representations (embeddings) of user preferences. We first divide user check-ins into multiple subgraphs, i.e., time slots, based on time information. In a GCN model, aggregating information only from nodes in the same subgraph enables better mining of users' time-based interests. However, a monotonous subgraph division has drawbacks. i.e., the Monotonous subgraph division cannot propagate the learned time preference features over multiple time slots because the subgraphs are constructed in advance. Thus, we further propose an edge propagation module to adjust edge affiliation, where edges represent check-ins, to propagate the user's time-based preference to multiple time slots. The propagation module is based on an unsupervised learning algorithm and does not require additional ground-truth labels. This approach to modeling time breaks with the traditional approach of treating temporal information simply as sequence information. Experimental results show that our proposed model further improved $Recall@5$ to 0.0874 on the Gowalla dataset while from 0.0360 to 0.0388 on the New York dataset compared with state-of-the-art GCN-based models.

1.5 Organization of the Thesis

In this section, we briefly describe the composition of the remainder of the thesis. The structure of the thesis is listed as follows:

- Chapter 2 introduces the related works in three aspects, including 1) acceleration of recommendation optimization, 2) modeling geographical information in recommendation systems, and 3) modeling temporal information in recommendation systems.
- Chapter 3 proposes a real-time periodic advertisement recommendation optimization model by using DA. (**Contribution 1**)
- Chapter 4 describes basic knowledge of POI recommendation, followed by previous works on the use of time and geographical information.
- Chapter 5 introduces a novel graph convolution network, combining the geographical neighbor concept to model geographical information for POI Recommendation. (**Contribution 2**)
- Chapter 6 introduces a users' interest propagation-based time-aware graph convolution network to model time information for POI Recommendation. After that, we combine the time-aware GCN with geographical information to further improve recommendation performance. (**Contribution 3**)
- Chapter 7 consists of two parts. We first show the conclusion of the thesis. Then, we discuss the promising future research directions.

We summarize our contributions and the corresponding chapters in Table 1-1. For convenience, we abbreviate contribution as C, and challenge as Ch.

Table 1-1: Contributions and the corresponding chapters

Chapter	Description	Contribution	Challenge
Chapter 3	A real-time periodic advertisement recommendation optimization model by using DA	C1	Ch1
Chapter 5	A novel Graph Convolution Network, combining the geographical neighbor concept to model geographical information for POI Recommendation	C2	Ch2
Chapter 6	A users' interest propagation-based time-aware graph convolution network to model time information for POI Recommendation	C3	Ch3

2 Related Work

This thesis proposes the methodologies to model temporal and geographical information from three aspects- acceleration of recommendation optimization, modeling geographical information, and modeling temporal information in recommendation systems. Thus, in this chapter, we introduce the related works in the corresponding three aspects.

2.1 Studies on Acceleration of Recommendation Optimization

To solve the recommendation optimization- an NP-hard problem, methodologies [7] [20] based on heuristics and linear programming were pioneered to accelerate optimization. Grigas et al. [20] transformed the optimization problem with budget constraints into a linear programming problem. Linear programming is an acceleration technique for solving approximate solutions of NP-hard problems. However, the method sacrifices the accuracy of the model. Deep learning models [94] [79] [80] provided new solution ideas for combinational optimization problems. Dai et al. [80] combined reinforcement learning with graph embedding to target the problem. They first store the state of the combinational optimization problem as graph nodes, followed by learning a greedy selection policy to construct a solution incrementally. Since the learned greedy selection policy reduces the search space of the problem, the technique can accelerate optimization. Based on Dai et al.'s work, Li et al. [79] modeled the optimization problem using a graph convolution network and proposed using a guided tree to reduce graph size. The technique reduced the local search range. Thus, it can accelerate the optimization. Ma et al. [93] also adopted graph networks to construct combinatorial optimization problems hierarchically for acceleration. They used reinforcement learning to learn a hierarchical policy to find the optimal combinations under constraint at each layer. In their method, each layer of the hierarchy is designed with a separate reward function for stable training. Due to the localized search of the hierarchy, they achieve acceleration. However, the high latency makes related work suffer from applying real-time recommendation

optimization.

2.2 Studies on Modeling Side Information

This section describes the related works on side information (geographical and time information) in POI recommendation systems.

2.2.1 Studies on Modeling Geographical Information

The geographical information-specific attribute of POI is integrated into recommendation systems as auxiliary side information. Ye et al. [30] set a precedent for using geographical information to improve the accuracy of recommendation systems by advocating that users tend to check in POIs that are close to their familiar areas. Based on Ye et al.'s work, Zhang et al. [29] proposed to adopt a multi-center Gaussian distribution of geographical information to fit a user's check-ins, which has profoundly impacted subsequent research on geographical information. The estimated geographical score is fused into the predicted preference score to complete the side information combination. With the wide application of machine learning techniques in POI recommendation, matrix factorization (MF)-based models [32] [33] can effectively solve data sparsity problems by adopting geographical information. Li et al. [33] argued for the adoption of two latent vectors to represent user information, where one is adopted to calculate user preferences and the other for scoring geographical information. In recent years, deep learning models [3] [38] are widely used to integrate geographical information. Chang et al. [3] pioneered the adoption of integrating geographical information into a graph convolution network (GCN) by modeling a power-law distribution, aggregating less information from distant neighbor nodes. Chang et al. trained two embeddings to represent a POI: one for check-in information and the other for geographical information. However, the related studies [3] [33] ignored the fact that POIs have unique geographical continuity, causing a loss of area information when modeling geographical information.

2.2.2 Studies on Modeling Temporal Information

In addition to geographical information, temporal information is also used as important side information to improve recommendation effectiveness. Yuan et al. [39] and Gao et al. [40] are pioneers in adopting time information to POI recommendations. In Yuan et al.'s work, time slots were used as a new dimension to refine user preferences by calculating the cosine similarity among users. The fused temporal information into a classical collaborative user-based filtering (UCF) algorithm [95]. After entering the machine learning [41] [42] era, sequence-based techniques [3] [4] [42] have received a great deal of attention. Zhao et al. [42] introduced a sequential mining model that utilized the word2vec framework [43] to capture the users' temporal preferences. The model can reflect different time characteristics on different days by training POI time embeddings. Zhang et al. [4] further optimized the sequential model; in addition to the structure of graph mining, they adopted two long short-term memory (LSTM) networks to extract the features related to the arrival and departure times of POIs. In their model, each user is assigned two trainable embeddings, one representing the user's preferences when arriving at a POI and the other representing the preferences when leaving the POI. Similar to Zhang et al., Liu et al. [96] also used LSTM networks to model temporal information. In their work, they introduced time weight decay to exponentially forget a user's premature history of clicks. Then, the learned temporal information is incorporated into the graph embeddings to accomplish the final prediction. However, modeling time information by sorting user check-ins in related work ignored users' time slot-based preferences, which is insufficient in modeling temporal information.

2.3 Remained Problems

In this chapter, we clarify the remained problems in previous studies from acceleration and modeling side information.

2.3.1 Problems with Acceleration

For acceleration, although previous studies attempted to speed up the solution of recommendation optimization problems, such as the use of deep learning algorithms in recent years to reduce the search size of the problem, the latency is still too high when dealing with real-time recommendation optimization problems. When solving highly time-sensitive tasks, i.e., the optimized result cannot be applied for a long time; we still need a novel periodic optimization strategy to update the optimization results in real time. Based on this starting point, this thesis introduces a periodic training framework and combines it with Fujitsu's digital annealer (DA) to accelerate recommendation optimization.

2.3.2 Problems with Modeling Side Information

For the modeling side information part, this chapter explains the shortcomings in terms of geographical and temporal information. In previous studies, geographical information was simply modeled as geographical distance in a deep learning model, which is inadequate. We still need a strategy to enable deep learning models to capture information about spatial geographical areas because POIs have unique geographical continuity. i.e., multiple POIs can form a geographical area. Based on this point, this thesis explores the user active areas and integrates geographical information into a graph convolution network (GCN) as extended geographical neighbors.

Temporal information was also applied in previous studies as a side information. In deep learning models, temporal information was simply modeled as a sequence of user check-ins in chronological order, which is insufficient. We still need a learning strategy to make the deep learning model learn the rich collaborative signals enriched in temporal information; time information can be divided into time slots to represent the users' preferences during a certain period of time. i.e., learning user preferences for time slots. To solve the problem, this thesis adopts a subgraph technique to divide the 24 hours of

the day into multiple time slots and generate one subgraph for each time slot, along with an edge propagation module to adjust edge affiliation in subgraphs.

3 Real-Time Periodic Advertisement Recommendation Optimization under Delivery Constraint using Quantum-inspired Computer[4]

In this chapter, we present our **Contribution 1**, using the quantum-inspired computer (Fujitsu digital annealer) to accelerate recommendation optimization to capture changes in user interest over time. Fujitsu digital annealer is one of the commercialized quantum-inspired computers[5]. Combining time information into a recommendation model has the potential to improve recommendation accuracy. However, simply using time information as users' check-in sequence to items is insufficient, especially for time-sensitive recommendation tasks, such as advertising (ad) recommendations. i.e., users' interest changed over time quickly. We are the first to adopt quantum-inspired computers with new proposed real-time periodic training techniques that can accelerate ad recommendation optimization and solve the problem. The training technique is suitable for periodic updating of user representations. However, the quantum-inspired computer Fujitsu DA can only accept one form of input, the quadratic unconstrained binary optimization (QUBO) model. In this chapter, we propose a technique to fast transform optimization tasks into QUBO model and realize real-time recommendations. More specifically, DA itself is fast in dealing with optimization problems. However, preprocessing the input data for DA is time-consuming, which is a bottleneck to use DA in real-time periodic recommendation systems. This chapter proposes an element-based method to fast derive the inputs of DA directly: weight matrix $\boldsymbol{W}$, vector $\boldsymbol{b}$, and constant con. This work is the first to explore the application of quantum-inspired computers in the field of ad recommendation, providing a new mindset for the field. DA, based on

[4] This chapter is based on "Real-time Periodic Advertisement Recommendation Optimization under Delivery Constraint using Quantum-inspired Computer"[68], by the same authors, which appeared in Proceedings of 2021 International Conference on Enterprise Information Systems (ICEIS 2021), pp. 431-441, 2021. Copyright(c) 2021.

[5] Fujitsu digital annealer provides APIs for general users to research and use. The APIs allow the user to set the input states and parameters of the DA and return the optimized results to the user after the DA is executed. In addition, DA supports a Jupyter-based development platform for general users to code and visualize results easily.

quantum computing, is the state-of-the-art solver to calculate combination optimization problems. By emulating qubits in a digital circuit, DA can quickly solve the NP-hard problem, inspiring us to use DA in ad recommendations. Besides, the proposed novel transformation method is for the QUBO model. QUBO model is general and widely used in solvers and is one of the research directions for solving NP-hard problems. If we displace the DA for other fast solvers, the proposed transformation method still possesses generality.

3.1 Introduction

The development of recommendation technique helps the market size of online advertising increase yearly. Real-time bidding (RTB) has become a typical delivery mechanism of online advertisements (hereafter, ads). In RTB, the advertisers publish their ads with the help of a demand-side platform (DSP). The DSP enables RTB and tracks the delivery of ads. Ad delivery aims to increase the number of conversions, defined as the cases when a customer completes a specific action with the advertiser's product, such as buying or subscribing. Whether a user converts or not reflects the performance of the ad delivery. Thus, a DSP needs to choose ads with a high conversion rate (CVR) according to each user's behavior.

A common task of DSP is to meet the needs of advertisers to obtain as much user engagement as possible. Previous studies [6] [7] aimed to optimize ads from advertisers' perspective with budget constraints. Yang et al. [8] focused on maximizing the DSP's profit while helping advertisers obtain valuable impressions under a given bidding budget. However, related studies neglected another critical requirement of DSP delivery constraints. DSP may want to deliver a specific number of ads in each category from many advertisers during a specific period because some categories have higher benefits for DSP and have a higher probability of matching target users' interests in a specific time. Because maximizing the CVR while satisfying delivery constraints is a combinatorial optimization problem, it is challenging and time-consuming to train, causing difficulty in

capturing changes in user interest over time and updating the ad optimization models under the delivery constraints periodically with a general-purpose computer.

In this chapter, the adoption of an Ising computer — Fujitsu digital annealer (DA), a quantum-inspired annealing machine [9], is proposed to accelerate and satisfy the delivery constraints. We aim to improve the CVR by periodic ad recommendation optimization. Periodic updates of the user model improve CVR because we can capture the users' latest behaviors to tune the recommendation model in real-time.

We model the periodic ad recommendation optimization problem as follows: in a short, fixed period (e.g., 20 min), DSP needs to update the user model while satisfying the constraints, such as delivering a specific number of ads in each category to users (for example, 1,000 ads for category A and 5,000 ads for category B). Due to the massive number of ads and users, it is challenging for the DSP to train the model quickly and accurately to decide the ad category with the highest probability of conversion for the target user. To solve the problem, we first predict the conversion probability of each ad category for each user by adopting two prediction models. Then, a technique to transform the optimization task into a quadratic unconstrained binary optimization (QUBO) model [9] quickly is proposed to solve the optimization problem. The contributions of our work are as follows.

-We propose a new real-time periodic recommendation model to speed up ad recommendations while satisfying the ad delivery constraints. With offline experiments on a real dataset, we show that the ad recommendation accuracy can be improved while satisfying the constraints.

-Our model is the first attempt to combine ad recommendation with a quantum-inspired computer DA, which can solve the combinatorial optimization problem quickly and accurately. We propose how to use a DA computer to achieve ad recommendations under the constraints, including transforming the problem to the QUBO model.

The remainder of this chapter is organized as follows. Related work is introduced in

Section 3.2. Our proposed method is presented in Section 3.3. Section 3.4 presents the experimental evaluation, followed by the conclusion in Section 3.5.

3.2 Related Work

In this section, we review the previous works and techniques on computational advertisement, including click-through rate (CTR) and conversion rate (CVR) prediction, ad recommendation, and constrained bidding optimization, which are related to our work.

3.2.1 CTR and CVR Prediction

CTR and CVR predictions [11] [12], which play an essential role in the online advertising industry, are modeled as classification problems. Logistic regression [11] [13] and generalized linear models are the most popular techniques to model a prediction task for achieving a high area under the curve (AUC). Shan et al. [11] proposed a triplet-wise learning model, adopting regression to rank the impressions in the following order: conversions (most valuable impressions), click-only impressions, and non-click impressions (least valuable ones). In recent years, factorization machines (FMs) [14] [15] have also been adopted for this purpose. FMs can work on large, sparse data to resolve cold-start problems. Pan et al. [15] presented a field-weighted FM for improved capturing of feature interactions between different fields. To further enhance the prediction accuracy, several deep learning-based models [16] [17] have been proposed for learning nonlinear features and historical information. Huang et al. [18] proposed a hybrid model using deep neural networks as a deep layer to capture nonlinear relationships in advertisement data while utilizing FM as a shallow layer to finish the prediction task. Their model successfully overcame the obstacle where a shallow-layer model could not use high-order features and reduced computational complexity.

Ad recommendation resembles CTR or CVR prediction. Kang et al. [19] proposed a real-time ad recommendation system that preprocesses a user's history data with a tree structure to obtain accurate recommendation results.

3.2.2 Constrained Bidding Optimization

Although our work is close to the ad recommendation task, the difference is that we need to satisfy constraints, making our problem more challenging. Maximizing the conversion ratio under constraints is a combinatorial optimization problem- an NP-hard problem.

In computational advertising, most constraints, such as budgets, are set from the advertiser's perspective. In particular, the advertisers want to maximize their benefits under budget constraints through a DSP. Abrams et al. [6] were among the first to consider bidder's budgets to optimize ad delivery while predicting bid prices. Wu et al. [7] combined the Markov decision process with a model-free reinforcement learning framework to address the complexity of optimizing the bidding strategy under budget constraints. Yang et al. [17] considered two types of constraints: bidder budgets and cost-per-click (CPC). They chose CPC as a crucial performance indicator constraint. After defining two constraints, they proposed an optimal bidding strategy to maximize CVR based on a linear programming problem. To the best of our knowledge, the study most similar to ours is that of Grigas et al. [20]. They optimized ads from the DSP's perspective: under budget constraints, DSP aims to accurately predict users' interest and maximize users' clicks while helping advertisers obtain valuable impressions. To achieve this goal, they used Lagrangian relaxation to develop their model and then transformed the problem into an optimization problem.

The research above aimed to optimize ads under various constraints, including budgets and CPC; however, periodic updates of the optimization problem remained an open question because of the time complexity. Even if we optimize the problem once, the optimized result cannot be applied to the real system for a long time because the preconditions for the optimization vary over time, which results in decreasing the effectiveness of the optimization result. Thus, periodic updates of the optimization problem are necessary to improve performance. Once we can realize periodic updates, we

may increase the accuracy of estimating the users' behavior and improve the optimization since the model can obtain users' behavior changes.

3.3 Proposed Method

We propose a DA-based method to optimize ads periodically to meet the needs of DSP for the ad delivery constraints and to reflect users' behavior changes. We aim to achieve a higher CVR by updating the optimization periodically in a short time. In each period, we execute a prediction algorithm, such as the Logistic regression model or XGBoost, to capture the probabilities of each user's candidate ad category, after which we solve the optimization problem by using DA, a quantum-inspired computer provided by Fujitsu.

3.3.1 Problem Formulation

Our goal is to optimize the delivered categories of ads for each user—with a high possibility of user conversions—while satisfying the number of ad deliveries for each category in a fixed period with periodic updates. We analyze each user's web page visit history to predict what ad category will be converted. For this, we adopt 26 categories (shown as C) of ads defined by the IAB taxonomy[6].

We formulate our problem as follows. Figure 3-1Figure 3-1: Prediction model shows our prediction model of the training and testing phases. In the training phase, we create a feature vector for each user $u \in U_{train}$, using his/her visit history during period $t_{session}$. U_{train} means the entire user set who converted during period t_{train}. By using the feature vector, we train a classification model to predict the category of ads converted by each user. In the testing phase, we predict and optimize ads to be delivered for every user, shown as U_{test}, who visited web pages during $t_{session}$ just before the prediction starting time $t_{pred,start}$. After the prediction and the optimization, the results are adopted during the next period t_{window} for the users in U_{test}. The process is different from the usual

[6] IAB Tech Lab - Taxonomy, https://www.iab.com/guidelines/taxonomy/

machine learning models. We pre-calculate the ad delivery category for each user U_{test} regardless of his/her future appearance in t_{window} because we do not have enough time to decide the ad category to deliver after knowing that he/she appears. We ignore predicting the ad category for the users not included in U_{test}; that is, a different strategy is adopted to deliver ads. Based on the know-how that users will appear in the log data continuously in a short period, updating both the prediction and the optimization frequently is necessary to achieve high accuracy. Besides, to satisfy the constraints while capturing changes in user interest over time, frequent updates of the optimization problem are indispensable.

Assume that each ad in 26 categories has constraints, where r_c is a non-negative real number, representing the delivery ratio of category c against the entire categories C satisfying $\sum_{c \in C} r_c = 1$. The actual constraint is the number of deliveries defined for each ad. We calculate r_c based on the given number of ads in each ad category during t_{window}. Subsequently, for each ad category $c \in C$, we estimate the conversion probability for each user u in the set U_{test}, shown as $p_{u,c}$, based on the pre-trained prediction model and his/her access log during $t_{pred,start} - t_{session}$ to $t_{pred,start}$. Because the ratio of delivered ad categories for test users set U_{test} must satisfy the number of delivery constraints $\forall c \in C, d_c = r_c \cdot |U_{test}|$, we optimize to choose the category for each user u in U_{test} with as high $p_{u,c}$ as possible under the delivery constraints. $|U_{test}|$ denotes the size of U_{test}. Although some users appear in t_{window} multiple times, for simplicity, we assume that each user appears only once during t_{window}, which is acceptable if we can shorten t_{window} by adopting our proposed method.

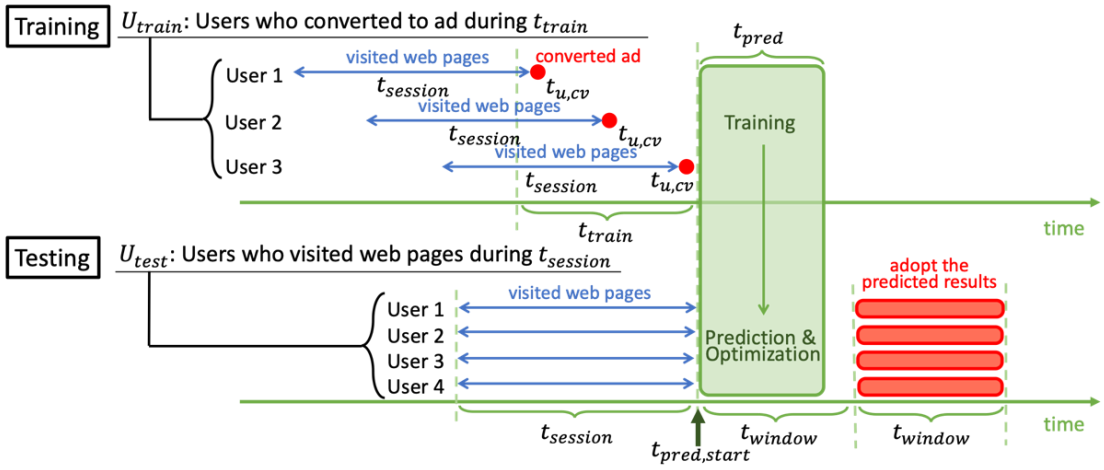


Figure 3-1: Prediction model

3.3.2 Overview of Proposed Method

Our framework consists of two steps: 1) a preprocessing step on standard CPUs and 2) an optimization step on DA. In the preprocessing step, for each user, our method predicts the CVR of each candidate category by using a pre-trained prediction algorithm. In the optimization step, we combine the predicted CVR with the delivery constraints and generate the final category for each user using DA. We use DA to accelerate the optimization of the delivery categories under the constraints. Note that the prediction algorithm and the optimization method are independent, which makes our method highly portable.

3.3.3 Conversion Probabilities of Ad Categories for Each User

In this subsection, we describe a method to calculate the probability of the ad category that a user will convert. Training data is collected to extract each user's visited web pages' categories and his/her converted ads' categories. Each user $u \in U_{train}$ has a feature vector $\boldsymbol{h}_u = (h_{u,1}, \ldots, h_{u,|C|})$, where $h_{u,c}$ represents the ratio of the web page category $c \in C$ user u visited during $t_{session}$ weighted by time, as shown in Eq. (3.1). In the thesis, we denote vectors and matrices in italicized bold type. Besides, in the thesis, a vector is represented as a row vector if not otherwise specified. Here, the weighting is

linear from 0 to 1, where the recent history has a more significant weight.

$$h_{u,c} = \frac{h'_{u,c}}{\sum_{c \in C} h'_{u,c}}, \tag{3.1}$$

$$\text{where } h'_{u,c} = \sum_{(t,c) \in V_u} \left(1 - \frac{s-t}{t_{session}}\right), \tag{3.2}$$

$$V_u = \left\{ \begin{matrix} (t,c) | \text{ user } u \text{ visited a web page} \\ \text{of ad category } c \in C \\ \text{at time } t \text{ in } t_{session} \end{matrix} \right\},$$

$$s = \begin{cases} t_{u,cv} & \text{(when training)} \\ t_{pred,start} & \text{(when predicting)} \end{cases}$$

We use a prediction algorithm to calculate the conversion probabilities of each ad category. To train the prediction algorithm, $\boldsymbol{h}_u$ is used as the input vector, and the converted category c_u is used as the output label for each user $u \in U_{train}$ who converted during t_{train}. At $t_{pred,start}$, we input the feature vector of each user $u \in U_{test}$ and calculate the conversion probability $p_{u,c}$ for each candidate ad category $c \in C$.

3.3.4 Optimizing Category Predictions

DA and QUBO Model

DA by Fujitsu Ltd. [9] aims to solve NP-hard combinatorial optimization problems, which are difficult to solve by using today's classical computers at high speed. DA can search for the minimum value of the energy function of a QUBO model. As a quantum computer, DA can only adopt the input of the QUBO model, as shown in Eq. (3.3).

$$E(\boldsymbol{x}) = -\frac{1}{2}\sum_i \sum_{j \neq i} W_{i,j} x_i x_j - \sum_i b_i x_i + con, \tag{3.3}$$

where $\boldsymbol{W} \in \mathbb{R}^{n \times n}$, $\boldsymbol{b} \in \mathbb{R}^{1 \times n}$, and $con \in \mathbb{R}^{1 \times 1}$ are the inputs of DA, and $x_i \in \{0,1\}$ is a

bit. n means the number of units in DA. Weight matrix $\boldsymbol{W}$ reflects the quadratic coefficients of the model, while vectors $\boldsymbol{b}$ and con represent linear coefficients and a constant, respectively. The value of *con*, the elements in $\boldsymbol{W}$, and the elements of $\boldsymbol{b}$ must be integers. Subscripts represent elements of a matrix or vector. i.e., $W_{i,j}$ represent the element in row i and column j of matrix $\boldsymbol{W}$ while x_i, b_i denoting the i-th element of the vector $\boldsymbol{x}$ and $\boldsymbol{b}$, respectively. DA calculates the global minimum value of $E(x)$ and outputs the value of all bits x, when $E(x)$ reaches a minimum.

DA-Based Category Prediction

Even after the conversion probabilities $p_{u,c}$ for each user are calculated in Section 3.3.3, we cannot simply choose the category with the highest probability as the prediction result because the number of ads in each category must satisfy the number of delivery constraints. Maximizing accuracy while satisfying the constraints is a combinatorial optimization problem, which is time-consuming and challenging to solve using a conventional computer. Instead, we use DA to accelerate the optimization. Note that accelerating and capturing changes in user interest is very important for time-sensitive services, like ad recommendation, because users' interest constantly changes over time.

We aim to maximize the prediction accuracy under the constraints of delivery distribution. The outputs of the DA must satisfy two constraints: 1) each user should be assigned only one category (constraint 1); 2) the number of ads to be delivered in each category must meet the delivery constraint (constraint 2).

We combine the predicted probabilities with the constraints and apply them to the QUBO model. To achieve the QUBO model format, we define an objective function with three terms in Eq. (3.4).

$$E'(\boldsymbol{q}) = -\alpha \sum_{u=1}^{|U|} \sum_{c=1}^{|C|} p_{u,c} q_{u,c} + \beta \sum_{u=1}^{|U|} \left(\sum_{c=1}^{|C|} q_{u,c} - 1 \right)^2 + \gamma \sum_{c=1}^{|C|} \left(\sum_{u=1}^{|U|} q_{u,c} - d_c \right)^2, \tag{3.4}$$

where $p_{u,c}$ is the probability from 0 to 100 (in percent) that user u will convert to category c, which is calculated from the prediction algorithm in Section 3.3.3; $\boldsymbol{q} \in \mathbb{R}^{1 \times n}$ is the output result vector of DA. $q_{u,c}$ is an element of $\boldsymbol{q}$, denoting that we partition the vector $\boldsymbol{q}$ to assign a unit to represent the allocation result of user u to category c. $q_{u,c} \in \{0,1\}$ shows that ads of category c are assigned to user u when $q_{u,c} = 1$ and vice versa, are not assigned to user u when $q_{u,c} = 0$. We adopt one-hot encoding to represent each user's assigned ad category with $|C|$ bits. $|U|$ and $|C|$ are the numbers of users and categories, respectively. Moreover $d_c = r_c \cdot |U_{test}|$ is the delivery constraint of category c that we must satisfy, where r_c is the delivery ratio of category c. Furthermore, α, β, and γ are three positive parameters. We assign category c as a predicted result for user u if and only if $q_{u,c} = 1$.

The constraints in Eq. (3.4) are soft, which causes several users to violate the constraint. Thus, a following post-process is applied. If he/she has multiple assigned categories, the category with the highest probability is assigned from the multiple assigned categories that do not have full assignments, i.e., from remaining categories among the multiple assigned categories. Besides, if he/she has no categories, the category with the highest probability among the remained categories is assigned.

3.3.5 Transforming Objective Function to The QUBO Model

To utilize DA, we must transform our defined objective function into a QUBO model and to derive three necessary inputs: weight matrix $\boldsymbol{W}$, vector, $\boldsymbol{b}$, and constant con of DA in Eq. (3.3). For convenience, we denote each bit x_k as $q_{u,c}(k = u \cdot |C| + c)$. Same

as in a QUBO model, our objective function also has quadratic, linear, and constant terms. In our objective function, we mix quadratic, linear, and constant terms in the function's three terms. However, in a QUBO model, the input of the quadratic coefficient is a weight matrix $\boldsymbol{W}$, the input of the linear coefficient is vector $\boldsymbol{b}$, and the input constant is con. Thus, we must expand the objective function to extract coefficients of each term and reorganize them into $\boldsymbol{W}$, $\boldsymbol{b}$, and con of the QUBO model. Subsequently, we feed three terms to DA as inputs. Because the function has three parts, for convenience and clarity, we introduce those three parts in the order below.

The first part $-\alpha \sum_{u=1}^{|U|} \sum_{c=1}^{|C|} p_{u,c} q_{u,c}$ in Eq. (3.4) is to maximize the accuracy because the term can reach a lower value linearly when a category with higher probability is selected for the user. We extract the linear coefficient into $\boldsymbol{b}^{prob} \in \mathbb{R}^{1\times n}$, as in Eq. (3.5).

$$\boldsymbol{b}_i^{prob} = \alpha \lfloor p_{u,c} \rfloor, \qquad where\ i = u * |C| + c \tag{3.5}$$

$\lfloor p_{u,c} \rfloor$ indicates to apply the floor function to $p_{u,c}$. The second part $\beta \sum_{u=1}^{|U|} (\sum_{c=1}^{|C|} q_{u,c} - 1)^2$ ensures the existence and uniqueness of the assigned category for each user. If and only if there exists one assigned category recommended to one user, both $\sum_{c=1}^{|C|} q_{u,c} - 1$ term and its square are 0. If there are no or multiple solutions, $(\sum_{c=1}^{|C|} q_{u,c} - 1)^2$ becomes larger than 0, producing a penalty value. This part generates quadratic terms, linear terms, and constants of the QUBO model shown in Eq. (3.3). We sort quadratic coefficients, linear coefficients, and constants into $\boldsymbol{W}^{user} \in \mathbb{R}^{n\times n}$, $\boldsymbol{b}^{user} \in \mathbb{R}^{1\times n}$, and $c^{user} \in \mathbb{R}^{1\times 1}$, as shown in Eq. (3.6), Eq. (3.7), and Eq. (3.8).

$$\boldsymbol{W}_{i,j}^{user} = 2\beta, \qquad where\ \left\lfloor \frac{i}{|C|} \right\rfloor = \left\lfloor \frac{j}{|C|} \right\rfloor \tag{3.6}$$

$$\boldsymbol{b}_i^{user} = -2\beta \tag{3.7}$$

$$c^{user} = |U|\beta \tag{3.8}$$

The third part $\gamma \sum_{c=1}^{|C|}(\sum_{u=1}^{|U|} q_{u,c} - d_c)^2$ ensures that the number of ads for each category satisfies the delivery constraints. For each category, the closer the number of the predicted category to the upper bound, the smaller $(\sum_{u=1}^{|U|} q_{u,c} - d_c)^2$ will be obtained. This part also generates a quadratic term, a linear term, and a constant of the QUBO model. Again, we sort quadratic coefficients, linear coefficients, and constant into $\boldsymbol{W}^{cate} \in \mathbb{R}^{n \times n}$, $\boldsymbol{b}^{cate} \in \mathbb{R}^{1 \times n}$ and $c^{cate} \in \mathbb{R}^{1 \times 1}$ in Eq. (3.9), Eq. (3.10), and Eq. (3.11).

$$\boldsymbol{W}_{i,j}^{cate} = 2\gamma, \qquad \text{where } i \bmod |C| = j \bmod |C|, \tag{3.9}$$

$$\boldsymbol{b}_i^{cate} = -2\gamma d_c, \qquad \text{where } c = j \bmod c \tag{3.10}$$

$$c^{cate} = \gamma \sum_{c=1}^{|C|} {d_c}^2, \tag{3.11}$$

We combine quadratic, linear, and constant terms in three parts to form the final weight matrix $\boldsymbol{W}$, vector $\boldsymbol{b}$, and constant *con* of the QUBO model and feed them to DA as inputs, where $\boldsymbol{W} = \boldsymbol{W}^{user} + \boldsymbol{W}^{cate}$; $\boldsymbol{b} = \boldsymbol{b}^{prob} + \boldsymbol{b}^{user} + \boldsymbol{b}^{cate}$; $con = c^{user} + c^{cate}$. The process of transformation to the QUBO model is shown in Algorithm 1.

Algorithm 1: Transforming an objective function to the QUBO model

Input: $\boldsymbol{p}$**:** conversion probability of all users

α, β, γ: parameters of trade-off

$\boldsymbol{d}$: delivery constraint of all ad categories

$|C|$: number of ad categories

$|U|$: number of users

Output: $\boldsymbol{W}, \boldsymbol{b}, con$: coefficients of the QUBO model

$n \leftarrow |C| \cdot |U|$

Initialize $\boldsymbol{W}, \boldsymbol{W}^{user}, \boldsymbol{W}^{cate}$ as $n \times n$ zero matrices

Initialize $\boldsymbol{b}, \boldsymbol{b}^{user}, \boldsymbol{b}^{cate}, \boldsymbol{b}^{prob}$ as $1 \times n$ zero vectors

for $i \leftarrow 1$ to n **do**

    $\boldsymbol{b}^{user}[i] \leftarrow -2$ ▷$\boldsymbol{b}^{user}[i]$ means the i-th element in $\boldsymbol{b}^{user}$

    $\boldsymbol{b}^{cate}[i] \leftarrow -2$

7 **end for**

8 **for** $i \leftarrow 1$ to $|U|$ **do**

9 ▷enumerate each pair of categories, $\boldsymbol{W}^{user}[x][y]$ means the x row y column element in $\boldsymbol{W}^{user}$

10 **for** k, j in combinations $(|C|, 2)$ **do**

11 $\boldsymbol{W}^{user}[i \cdot |C| + k][i \cdot |C| + j] \leftarrow 2$

12 $\boldsymbol{W}^{user}[i \cdot |C| + j][i \cdot |C| + k] \leftarrow 2$

13 **end for**

14 **end for**

15 $c^{cate} \leftarrow 0$

16 **for** $i \leftarrow 1$ to $|C|$ **do**

17 ▷enumerate each pair of users

18 **for** k, j in combinations $(|U|, 2)$ **do**

19 $\boldsymbol{W}^{cate}[k \cdot |C| + i][j \cdot |C| + i] \leftarrow 2$

20 $\boldsymbol{W}^{cate}[j \cdot |C| + i][k \cdot |C| + i] \leftarrow 2$

21 **end for**

22 $c^{cate} \leftarrow c^{cate} + d_i^2$

23 **end for**

24 $c^{user} \leftarrow |U|$

25 **for** $i \leftarrow 1$ to $|U|$ **do**

26 **for** $j \leftarrow 1$ to $|C|$ **do**

27 $\boldsymbol{b}^{\boldsymbol{prob}}[(i-1) \cdot |U| + j] \leftarrow \lfloor p_{i,j} \rfloor$

28 **end for**

29 **end for**

30 $\boldsymbol{W} \leftarrow \beta \cdot \boldsymbol{W}^{user} + \gamma \cdot \boldsymbol{W}^{cate}$ ▷ ·denotes the scalar multiplication

31 $\boldsymbol{b} \leftarrow \alpha \cdot \boldsymbol{b}^{\boldsymbol{prob}} + \beta \cdot \boldsymbol{b}^{\boldsymbol{user}} + \gamma \cdot \boldsymbol{b}^{\boldsymbol{cate}}$

32 $con \leftarrow \beta \cdot c^{user} + \gamma \cdot c^{cate}$

33 return $\boldsymbol{W}, \boldsymbol{b}, con$

3.3.6 Utilization of DA

After we feed the weight matrix**,** $\boldsymbol{W}$, vector**,** $\boldsymbol{b}$, and constant *con* to DA as input, DA provides two annealing modes to be selected: normal mode and replica-exchange mode[9]. Because the normal mode requires us to train annealing parameters, for convenience, we choose the replica-exchange mode, which performs “parallel tempering”

and sets the temperature automatically. When the energy is stable, the DA returns the status of all bits. For each user, we check the status of the corresponding bits and judge whether both constraints are satisfied. We adopt the result only when the following two constraints are satisfied: a user is assigned to only one category *c* (constraint 1), and the total number of users receiving category c ads does not violate the maximum number *DC* (constraint 2). Otherwise, the post-process described in Section 3.3.4 is adopted. The process of utilizing DA is shown in Algorithm 2.

3.4 Experiment Evaluation

3.4.1 Dataset

We used real log data for the experimental evaluation to verify our proposed method. The log data consists of an auction and conversion log accumulated by Geniee DSP[7]. The auction log is generated when a user visits a web page with an advertisement tag, and RTB is performed. The conversion log is generated when a user who views an advertisement performs a conversion.

In this experiment, the identifier (id) assigned to each unique browser is assumed to be the user's unique id. The visit history of web page categories used as input features can be aggregated from the auction log utilizing the user's unique id and time stamp. We use the ratio of each advertisement category in the auction log in each t_{window} as the delivery constraint.

We used raw data collected from November 6th, 2019, to November 8th, 2019. The 24-hour data on November 7th was used to tune time parameters, i.e., t_{train}, $t_{session}$, and t_{window}. As for t_{pred}, it must satisfy less than t_{window} so that we will confirm it in the experiment. The 24-hour data on November 8th was used for the experimental evaluation. We divided the evaluation data by t_{window} to simulate the proposed method.

[7] Geniee, Inc. https://en.geniee.co.jp/

For example, 24-hour evaluation data are divided into72 windows when t_{window} = 20 min.

Algorithm 2: Utilizing DA

Input: $\boldsymbol{p}$: conversion probability of all users
α, β, γ: parameters of trade-off
$\boldsymbol{d}$: delivery constraints of all ad categories
$|C|$: number of ad categories
$|U|$: number of users

Output: ***result***: predicted ad category for all users

$\boldsymbol{W}, \boldsymbol{b}, con \leftarrow \text{Transform}(p_{u,c}, \alpha, \beta, \gamma, d_c)$, shown in Section 3.3.5
$\boldsymbol{q} \leftarrow \text{DigitalAnnealing}(\boldsymbol{W}, \boldsymbol{b}, c)$ ▷ DigitalAnnealing means the use of DA to obtain results
$\boldsymbol{U}' \leftarrow \emptyset$
for $i \leftarrow 1$ to $|U|$ **do**
  $s \leftarrow \sum_{j=1}^{|C|} q_{ij}$
  if s=1 **then** ▷only 1 result bit with value 1
    for $j \leftarrow 1$ to $|C|$ **do**
      if $q_{i,j}$=1 **then**
        $result_i \leftarrow j$
        $d_j \leftarrow d_j - 1$
      end if
    end for
  else
    $\boldsymbol{U}' \leftarrow \boldsymbol{U}' \cup \{i\}$▷user i needs a post-process
  end if
end for
apply post-process to $\forall u$ in $\boldsymbol{U}'$
  ▷described in Section 3.4.2
return ***result***

As shown in Figure 3-2, t_{window} slides over time, and we use the data during t_{train} period as training data. Importantly, when tuning time parameters with data on November 7th, in several t_{window} (such as 00:00 to 00:20), we need to use data on November 6th to generate $t_{session}$ and t_{train}. The number of converted users was 9,823 on November

6th, 9,328 on November 7th, and 9,874 on November 8th. The number of users in the training and test datasets, U_{train} and U_{test}, depends on the time parameters. Notably, some of the converted users in t_{window} did not visit the web pages during $t_{session}$, so they were not included in U_{test}. The number of converted users included in U_{test} was 4,706 out of 9,823 on November 8th.

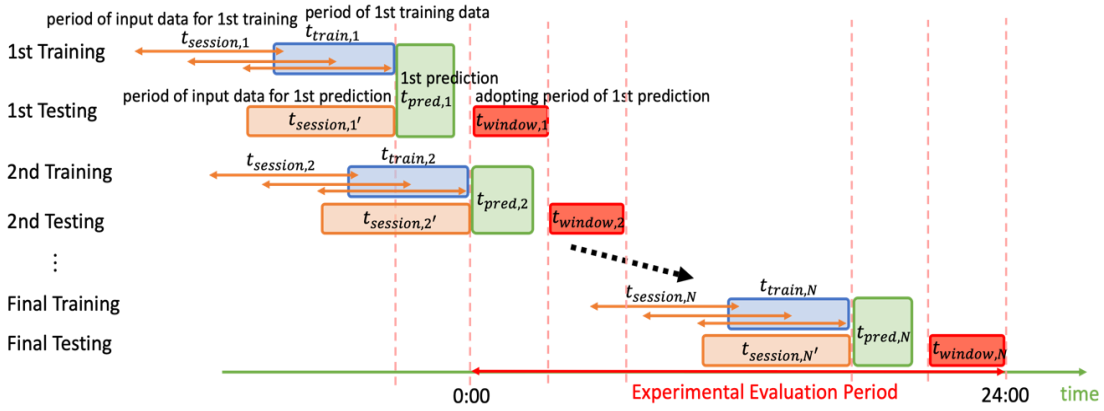


Figure 3-2: Overview of periodic recommendation

3.4.2 Evaluation Metrics

The novelty of our proposed method is to solve the ad optimization problem periodically around a short period of time to capture changes in user interest over time, maximizing the CVR while satisfying the number of delivery constraints. To confirm that our proposed method predicts an ad category for each user with high accuracy while satisfying the delivery constraints in an appropriate duration, we use three metrics: $Accuracy_{window}$, $Accuracy_{all}$ and execution time. Here, we assume that the ground truth is the category in which each user converts in t_{window}. We do not use the AUC metric (which is common in CVR prediction) because our task is different from predicting the conversion category under the delivery constraints. We need to verify whether our prediction is correct. Thus, we adopted accuracy instead of AUC.

$Accuracy_{window}$ is the average ratio of correctly predicted users to all converted users

in t_{window}, shown in Eq. (3.12).

$$Accuracy_{window} = avg_{all\ windows} \frac{\left|U_{correct \cap cv}^{window}\right|}{\left|U_{cv}^{window}\right|}, \tag{3.12}$$

where $U_{correct \cap cv}^{window}$ is the set of converted users with the same predicted category as the category in the ground truth; U_{cv}^{window} is the set of all converted users in t_{window}. avg denotes the average value function. The input is a set of real numbers and the output is a real number.

$Accuracy_{all}$ shown in Eq. (3.13) is the ratio of correctly predicted users to all converted users in the test dataset. We introduce $Accuracy_{all}$ as a fair comparison between the different time parameters because when we change t_{window}, it affects the set of converted users.

$$Accuracy_{all} = \frac{\sum_{all\ windows}\left|U_{correct \cap cv}^{window}\right|}{|U_{cv}|}, \tag{3.13}$$

where U_{cv} is the set of total converted users in the test dataset.
Finally, the execution time measures the time (in seconds) spent to generate the recommendation.

All the experiments were executed on a server with the following configuration: two Intel Xeon Gold 6148 CPUs, 2.40 GHz (20 cores, 40 threads), with 192 GB of memory, running on CentOS 7.6. The optimization process (finding the minimum value and bits of the QUBO function) was run on DA [9].

3.4.3 Prediction Algorithm

To generate the conversion probabilities of ad categories for each user described in Section 3.3.3, we need to adopt a base algorithm to receive the input feature vector $\boldsymbol{h}_u$ and output the conversion probability $p_{u,c}$ for each candidate ad category $c \in C$. In our experiment, we chose Logistic regression and XGBoost [21] as prediction algorithms because of their effectiveness and high speed.

3.4.4 Baseline Methods

We compared our proposed DA-based method[8] with two baselines: "Random" and the genetic algorithm (shown as GA).

The "Random" method omits the optimization step and adopts a random selection of ad categories but adopts the post-process shown in Section 3.3.4 to satisfy the delivery constraints. By comparing our method with Random, we can confirm the effectiveness of solving delivery constraints.

The genetic algorithm (GA) [22] was also chosen to solve the combinational problem as a popular and efficient method to confirm the effectiveness of DA in solving delivery constraints more strictly. GA runs on common CPUs and does not require binary bits. Instead of one-hot encoding, we can use one variable to represent each user's candidate results so that the objective function is simplified as in Eq. (3.14).

$$E''(\boldsymbol{q}) = -\delta \sum_{u=1}^{|U|} p_{u,q_u} + \varepsilon \sum_{c=1}^{|C|} \left(\sum_{u=1}^{|U|} f_{u,c} - d_c \right)^2, \tag{3.14}$$

where p_{u,q_u} is the probability that user u converts to category q_u; d_c is the delivery number of category c that we must satisfy; $f_{u,c} \in \{0,1\}$ is a binary variable where $f_{u,c}$ equals 1 when the converted category q_u equals category c, as shown in Eq. (3.15); $\sum_{u=1}^{|U|} f_{u,c}$ is used as a count for each category. i.e., how many ads are delivered; δ and ε are two parameters.

$$f_{u,c} = \begin{cases} 1, & when\ q_u = c \\ 0, & otherwise \end{cases} \tag{3.15}$$

Compared with (3.4), (3.14) omits the constraint, ensuring that each user has only one

[8] https://github.com/bakubonmo/Rec

prediction result. As in DA, GA does not guarantee the satisfaction of the given constraint. Therefore, we also adopt the post-process described in Section 3.3.4.

3.4.5 Time Parameters Tuning

In this section, we tune the parameters t_{window}, t_{train}, *and* $t_{session}$ to achieve the best average $Accuracy_{window}$ by evaluating the classification using the prediction algorithm without considering the delivery constraints. We used the 24-hour data on November 7th to tune the parameters.

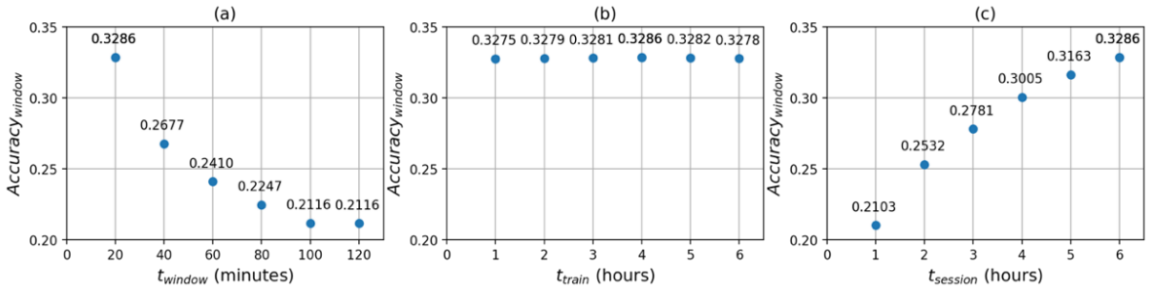


Figure 3-3: Result of Accuracy_window without constraints when changing the time parameters: (a) Fixed at t_train = 4 h, t_session= 6 h, and varying t_window; (b) Fixed at t_window = 20 min, t_session= 6 h, and varying t_train; (c) Fixed at t_window = 20

Figure 3-3 shows the results of $Accuracy_{window}$ when parameters t_{window}, t_{train}, and $t_{session}$ are varied. As shown in Figure 3-3(a), the accuracy increases with a decrease in the model update interval t_{window} because the latest action of the user can be reflected by a decrease t_{window}. In Figure 3-3(b), the accuracy peaks when the training data period t_{train} is four hours because if t_{train} is small, the number of data points in t_{train} becomes small, resulting in poor learning outcomes. However, if t_{train} is extremely large, the accuracy decreases due to training on old data. In Figure 3-3(c), a larger $t_{session}$ increases the accuracy because more visit history of the user is reflected by increasing $t_{session}$.

Finally, we set the parameters as t_{window} = 20 min, t_{train} = 4 h, and $t_{session}$ = 6 h

for the rest of the experiments. Further tuning such as decreasing t_{window} and increasing $t_{session}$ will be available as long as $t_{pred} \leq t_{window}$ holds.

3.4.6 Experimental Results Under the Delivery Constraints

We used the 24-hour data on November 8th for the evaluation, which was split into 72-time slots because of t_{window} = 20. Parameters $\alpha, \beta,$ and γ in our objective function in Eq. (3.4) and parameters δ and ε in the GA's objective function in Eq. (3.14) were tuned on the first 10 time slots of the data. In contrast, the remaining 62 time slots data were used for evaluation. By adopting a grid search, we chose $\alpha = 1, \beta = 5, \gamma = 10$, $\delta = 1$, and $\varepsilon = 10$.

Table 3-1 shows the experimental results. Because the constraints in Eq. (3.4) and Eq. (3.14) are soft, we show the percentage of users who violated the constraints, shown as the violation rate in Table 3-1. The constraints in Eq. (3.4) are soft, which causes several users to violate the constraints. So, we define the violation rate as the percentage of users who violate the constraints. The During the post-process for violated users described in Section 3.3.4, we chose each user's ad category among his/her top six ad categories. Recall that $Accuracy_{window}$ shows the average accuracy per window. Thus, we can compare with $Accuracy_{window}$ only when the same parameters ($t_{session}$, t_{window}, and t_{train}) are used among the methods. On the contrary, if the different parameters are used, we cannot use $Accuracy_{window}$ for fair comparison because the converted users in each window are different. In such a case, we must use $Accuracy_{all}$ which shows the correctly predicted users against all converted users in the whole test dataset. Compared with the result in Mo et al. [10], the $Accuracy_{all}$ of GA-based method improves because of fine-tuned batch size.

We conducted a paired t-test for accuracies between each baseline and our proposed method. As a result, we confirmed that our proposed method outperforms the baselines, which is statistically significant at $p < 0.01$. In addition, we demonstrated that our

Table 3-1 Experiment Results (t_{train} = 4 h and $t_{session}$ = 6 h)

Method	Prediction algorithm	Optimization Technique	Acc_w	Acc_a	Violation rate	Execution time (s) (t_{pred})
Baseline	Logistic regression	Random (t_{win}=20 min)	0.180	0.219	0.595	
		GA (t_{win}=20 min)	0.202	0.239	0.030	525
Proposed		DA (t_{win}=20 min)	0.229*	0.278*	0.020	108
		DA (t_{win}=5 min)		0.324*	0.020	108
Baseline	XGBoost	Random (t_{win}=20 min)	0.180	0.216	0.595	
		GA (t_{win}=20 min)	0.198	0.237	0.029	526
Proposed		DA (t_{win}=20 min)	0.229*	0.277*	0.013	109
		DA (t_{win}=5 min)		0.322*	0.013	108

* Statistically significant at $p < 0.01$ when comparing with our proposed method, DA, with Random and GA

proposed method achieved the shortest execution time. Notably, we do not compare the execution time with the Random method because the method is not a combinational optimization algorithm and has the lowest recommendation accuracy.

We also conducted an experiment on different t_{window} to confirm the effectiveness of shorting window size. Because the DA completed the execution within 5 min, we set

t_{window} to 5 min with the other time parameters as in the previous setting ($t_{session}$= 6 h and t_{train} = 4 h). In Table 3-1, for convenience, we abbreviate $Accuracy_{window}$ as Acc_w, t_{window} as t_{win}, and $Accuracy_{all}$ as Acc_a. As shown in Table 3-1, we confirmed $Accuracy_{all}$ increased drastically as t_{window} was shortened, which means that if the optimization algorithm runs faster, the number of users that we correctly predict their ad categories increase. Hence, shortening the periodic optimization of DA and capturing changes in user interest is essential.

To summarize the experimental results, with Logistic regression, we successfully shortened the periodic advertisement recommendation from 525s to 108s and increased the accuracy from 0.239 to 0.324 (35.56%) compared to GA. With XGBoost, we also shortened the execution time from 526s to 108s while improving accuracy from 0.237 to 0.322 (35.86%).

3.4.7 Experiment on Comparing Proposed Transformation Method with Polynomial Expansion

We also notice the experiment with the polynomial expansion(baseline) to the QUBO model as Algorithm 3. The execution time of polynomial expansion is long (551s for preparing the objective function of the QUBO) and even longer than the execution time (108s) of DA. Note that polynomial expansion involves symbolic computations and takes longer execution time than numerical computations. Our proposed method drastically reduces the transformation time to 0.7s, bridging the gap between DA and real-time ad recommendation applications. The proposed method overcomes the problem of too long a transformation time to adopt DA to ad recommendation.

3.5 Conclusion

In this chapter, we proposed a new method, namely the DA method, to optimize ads periodically in a short period by using DA to solve the optimization problem: maximizing

Algorithm 3: Transforming an objective function to the QUBO model using polynomial expansion and sympy library[9]

Input: $\boldsymbol{p}$: conversion probability of all users

α, β, γ: parameters of trade-off

$\boldsymbol{d}$: delivery constraint of all ad categories

$|C|$: number of ad categories

$|U|$: number of users

Output: $\boldsymbol{W}, \boldsymbol{b}, con$: coefficients of the QUBO model

$n \leftarrow |C| \cdot |U|$

$\boldsymbol{v} \leftarrow [{x_1}', {x_2}', \ldots, {x_n}']$ ▷Initialize n variables for polynomial expansion

Initialize $\boldsymbol{t1}, \boldsymbol{t2}, \boldsymbol{t3}$ as empty symbolic lists to save three terms in Eq. (3.4)

▷calculate the $\boldsymbol{term1}$ in Eq. (3.4)

for $i \leftarrow 1$ to $|U|$ **do**

  for $j \leftarrow 1$ to $|C|$ **do**

▷Note that this operation is a symbolic computation, not a numerical computation, hence the execution time is long

    $\boldsymbol{t1} \leftarrow \boldsymbol{t1} + \lfloor p_{i,j} \rfloor \cdot \boldsymbol{v}[i \cdot |C| + j]$

  end for

end for

▷calculate the $\boldsymbol{term2}$ in Eq. (3.4)

for $i \leftarrow 1$ to $|U|$ **do**

  $\boldsymbol{sum\_list}$ ←empty symbolic list ▷Used to calculate the constraint 1 for a user

  for $j \leftarrow 1$ to $|C|$ **do**

    $\boldsymbol{sum\_list} \leftarrow \boldsymbol{sum\_list} + \boldsymbol{v}[i \cdot |C| + j]$ ▷Symbolic computation

  end for

  $\boldsymbol{sum\_list} \leftarrow (\boldsymbol{sum\_list} - 1)^2$

  $\boldsymbol{t2} \leftarrow \boldsymbol{t2} + \boldsymbol{sum\_list}$

end for

▷calculate the $\boldsymbol{term3}$ in Eq. (3.4)

for $i \leftarrow 1$ to $|C|$ **do**

  $\boldsymbol{sum\_list}$ ←empty symbolic list

  for $j \leftarrow 1$ to $|U|$ **do**

    $\boldsymbol{sum\_list} \leftarrow \boldsymbol{sum\_list} + \boldsymbol{v}[i \cdot |U| + j]$ ▷Symbolic computation

  end for

[9] https://www.sympy.org/en/index.html

25 $\boldsymbol{sum\_list} \leftarrow (\boldsymbol{sum\_list} - d_i)^2$

27 $\boldsymbol{t3} \leftarrow \boldsymbol{t3} + \boldsymbol{sum\_list}$

28 **end for**

29 $\boldsymbol{polynomial} \leftarrow -\alpha \cdot \boldsymbol{t1} + \beta \cdot \boldsymbol{t2} + \gamma \cdot \boldsymbol{t3}$ ▷merge three terms

30 ▷use sympy library to transform QUBO model

31 $\boldsymbol{expanded\_polynomial}$ ←expand($\boldsymbol{polynomial}$) ▷Expand the polynomials using sympy library with time complexity $O(|T|^2)$ of symbolic computation

32 $\boldsymbol{W}, \boldsymbol{b}, con$ ←simplify($\boldsymbol{expanded\_polynomial}$) ▷Get the final output using sympy library, simplify() is a function to simplify polynomials

33 return $\boldsymbol{W}, \boldsymbol{b}, con$

CVR while satisfying the delivery constraints, that is, the number of ads delivered for each category. Our method consists of two steps: 1) prediction to generate ad candidates for each user, and 2) optimization of candidates to meet the number of ad delivery constraints, which is difficult to solve within an acceptable period on a general-purpose computer. Experiments on a real dataset showed that our proposed method successfully improved the accuracy by shortening the periodic advertisement recommendation: 0.239 to 0.324 (35.56%) with prediction algorithm Logistic regression while shortening the execution time from 525s to 108s; and 0.237 to 0.322 (35.86%) with XGBoost while shortening the execution time from 526s to 108s.

4 Preliminary of Combination of Side Information with Graph Convolution Network (GCN) for Point-of-interest (POI) Recommendation

4.1 Introduction

In this chapter, we introduce the related work and basics knowledge of combination of side information, including time and geographical information, with a graph convolution network to improve POI recommendation accuracy. The advent of the information age has improved people's standard of living but has raised the problem of information overload. Recommendation systems are effective tools to help users filter massive amounts of information and assist them in making decisions. In location-based social networks (LBSNs), point-of-interest (POI) recommendation systems recommend unvisited POIs by analyzing user check-in history based on user and POI locations. Open-source datasets from LBSNs, such as Gowalla[10] and Yelp[11], allow users to share their check-in experience, making a detailed analysis of users' behavior possible. Accuracy is an important metric of the effectiveness of recommendation results and user satisfaction. Thus, many recommendation systems have improvement of accuracy as their primary goal.

In recent years, the use of neural network techniques has led to a boom in the use of recommendation systems. Graph convolution network (GCN) models [2] [3] [4] have become state-of-the-art recommendation algorithms due to the effectiveness of calculating the embeddings of users and items while ranking higher predicted preference scores as the preferred items. By aggregating information from neighbor nodes and passing collaborative signals from high-order connectivity, GCNs combine the core idea of collaborative filtering, i.e., similar users have similar preferences, with machine

[10] http://snap.stanford.edu/data/loc-gowalla.html
[11] https://www.yelp.com/dataset_challenge

learning. Several related works have attempted to adopt GCNs for POI recommendation, considering that using various side information is a potential method to improve model performance. Inspired by lightweight models (e.g., LightGCN [5] and LR-GCCF [23]), Chang et al. [3] were the first to integrate the power-law distribution of the geographical distance between two POIs into a GCN model. The use of two GCNs, one for check-in information and the other for geographical information, improves the representation ability of user and POI embeddings. However, simply modeling geographic information as geographical distances is insufficient, ignoring users' active areas in cities, which leads to sub-optimization of the model. The methodology of combination of users' active areas with GCN models to improve recommendation performance need to be explored and designed. Besides, it leads to a significant increase in the number of trainable parameters, making the model training difficult. The proposed technique called GN-GCN to solve the above problem and make full use of geographical information will be introduced in Chapter 5.

In addition to geographical information, time information is an essential side-module in GPR [3] and GNN-POI [4]. However, simply adopting time information by sorting user check-ins in chronological order and modeling user check-in sequences cannot fully exploit collaborative signals in time information. Although similar users can be extracted according to the check-in time for POIs, this is not sufficient. For example, both the target user and user A tend to go to a supermarket after breakfast, but for user B, checking a supermarket in the evening is a good selection. In this case, user A can be a time-based high-order neighbor to the target user, and the model filters out user B, even if user B has the same preference (going to a supermarket) as the target user. Thus, learning unique time-based embeddings for users and items is essential. A novel GCN model- EPT-GCN will be introduced in Chapter 6 in detail to combine time information and solve the above problem.

4.2 Related Work

Our work for Contribution 2 and 3 are related to POI recommendations and GCN-based recommendations. Thus, in this chapter, we introduce two main aspects of previous works: 1) side information, i.e., time and geographical information, used in POI recommendation, and 2) GCNs in recommendation systems.

4.2.1 Side information used in POI Recommendation

Geographical information: As an intrinsic attribute of POIs, geographical information is widely used in POI recommendation systems to model regions frequently checked by a target user. Ye et al. [30] were the first to apply geographical information to recommendation systems to improve accuracy. They suggested users are less likely to check in to POIs far from their familiar areas. Based on Ye et al.'s work, Baral et al. [27] and Zhang et al. [29] optimized geographical models. Zhang et al. proposed the famous multi-center Gaussian distribution of geographical information to fit a user's check-ins, which has profoundly impacted subsequent research on geographical information. With the widespread use of machine learning techniques in POI recommendation, matrix factorization (MF)-based models [32] [33] can effectively solve data sparsity problems by adopting geographical information. Li et al. [33] argued for the adoption of two latent vectors to represent user information, where one is adopted to calculate user preferences and the other for scoring geographical information. Deep learning algorithms [3] [38] are increasingly being adopted for integrating geographical information. To the best of our knowledge, Chang et al. [3] were the first to integrate geographical information into a GCN by modeling a power-law distribution and aggregating less information from distant neighbor nodes. Chang et al. trained two embeddings to represent a POI: one for check-in information and the other for geographical information. However, simply modeling geographical information as geographical distances is insufficient, ignoring users' active areas in cities, which leads to sub-optimization of the model. How to combine users' active areas with GCN models to improve recommendation performance is still an open question.

Time information: Time information is generated when a user interacts with a POI. Time information helps recommendation systems capture user preference changes over time. Yuan et al. [39] and Gao et al. [40] are pioneers in considering time information when making POI recommendations. Yuan et al. used time slots as a new dimension to refine user preferences by calculating the cosine similarity among users. After the era of the prevalence of machine learning models [41] [42] in recommendation systems, sequence-based techniques [3] [4] [42] have received a great deal of attention. Zhao et al. [42] proposed a sequential mining model to capture the temporal preferences of users inspired by the word2vec framework [43]. The well-trained POI time embeddings can reflect various time characteristics on different days. Zhang et al. [4] further optimized the sequential model from Ying et al.'s work; in addition to the structure of graph mining, they adopted two LSTM networks to extract the features of the arrival and departure times of POIs.

The above models adopt time information; however, they struggle to mine users' time-based high-order connectivity, even for state-of-the-art GCN models [44] [4], limiting the representation ability and preventing further accuracy improvement. The users' time-based high-order connectivity enables the extraction of indirect neighbors with similar preferences in a time slot.

4.2.2 Graph Convolution Network in Recommendation System

With the development of machine learning techniques, network embedding-based models [45] [46], which condense the representation of a user or an item into an embedding, have gained breakthroughs and attracted the attention of researchers in the area of recommendation systems in recent years. Graph convolution network (GCN) models [44] [47] [48] [49] [50], as one of embedding-based models, are widely adopted to improve performance. Wang et al. [25] pioneered the application of GCNs to flesh out the concept of collaborative filtering. They used check-ins to compose a user-item interaction graph, where they performed neighbor aggregation to learn the unique

embeddings of users and items. Their performance improvement laid a solid foundation for subsequent GCN-based research. In subsequent developments, researchers realized that the nonlinear activation function might have a detrimental effect on the GCN model performance. Chen et al. [23] revisited and explored a technique for applying GCNs to recommendation systems. They suggested that eliminating the nonlinear activation function and constructing a residual network can simplify the model and improve accuracy. In their proposed LR-GCCF model, the embeddings of users and items are updated by a linear accumulation of self-connection and neighbor information aggregation. He et al. [5] further simplified the design of the GCN model; they not only eliminated the nonlinear activation function but also eliminated self-connection. Neighbor information is linearly propagated on the user-item interaction graph to output the learned embeddings at each hidden layer. Then, a weighted sum operation is adopted to gather the learned layer embeddings as the final embeddings. Experiments on real datasets confirmed that the simplified model has better representational ability, resulting in improved recommendation performance.

Based on previous work, the application of the subgraph technique [2] [52] [53] [54] presents a new direction for GCN research. Liu et al. [2] pointed out that the learning process of user embeddings in GCN models may affect high-order nodes with no common interests with a target user. To solve this problem, they adopted three MLP layers to form an unsupervised subgraph generation module, aiming to filter out nodes of no common interest. They successfully avoided propagating negative information during the training process. As an application of the subgraph technique, Peng et al. [53] confirmed that only a small portion of latent rather than smoothed or rough features positively influence recommendation accuracy, whereas most noise features reduce accuracy. They then partitioned user-item interaction graphs into smooth, rough, and noisy graphs, which enabled the design of an effective graph denoising encoder (GDE) model to emphasize smoothed features while filtering out noise.

Despite the above techniques, applying the subgraph technique to POI

recommendation systems remains an open problem, especially for time slot partitioning. However, the subgraph technique matches well with time slot applications, and two issues still need to be addressed. First, previous node-based subgraph techniques divide each node (user and item) into one subgraph, which is inconsistent with the real situation. Because a POI can have multiple suitable time slots for check-ins, the subgraph module should divide the check-ins into multiple time slots (subgraphs) based on the learned high-order time slot embeddings. Second, although subgraph-based models, such as IMP-GCN [2], filter high-order neighbors with no common features, a subgraph generation module composed of only low-order features has difficulty representing high-order similarity. Low-order features mean that the module only considers zero- and first-layer embeddings to create a cluster maker, which lacks high-order information. The proposed model EPT-GCN introduced in Chapter 6 will combine the time information with GCN by using a proposed novel subgraph technique.

4.2.3 Preliminary

In this section, we introduce the definitions within the POI recommendation domain. Note that due to the different structure of the models, different notations are used for different models. For convenience to find relevant notation, we list the notation for the models separately in Section 5.2 and Section 6.2.

Definition 1 (POI recommendation): Assume that the dataset contains M users and N items. $U = \{u_1,\ u_2,\ u_3, \dots, u_M\}$ and $P = \{p_1,\ p_2,\ p_3, \dots, p_N\}$ represent sets of users and POIs, respectively. Each POI has an intrinsic attribute representing geographical latitude and longitude, written as $\left\{lat_{p_j},\ lon_{p_j}\right\}$. To mine users' active areas to represent his/her geographical information, we define $A_{u_i} = \left\{a_{u_i,1},\ a_{u_i,2}, \dots, a_{u_i,m}\right\}$ as the set of active areas of user u_i. Note that a user may have multiple active areas among cities. A geographical distance-based clustering algorithm sets the number of each user's active areas. For example, check-ins less than 1km away are merged into the same cluster. We

define a set of POI and timestamp pairs $C_{u_i} = \{(p_j, timestamp) | u_i \in U, p_j \in P,\ u_i$ checked $p_j\}$ as user u_i's check-in log. Shown in Figure 4-1, the purpose of POI recommendation for target user u_i is to recommend a list of ranked POIs that target user u_i has not visited using the historical check-in logs $C_{u_i} \in C$, where $C = \{C_{u_1}, C_{u_2}, C_{u_3}, \dots, C_{u_M}\}$.

Definition 2 (User's active area neighbor): The target user's active area neighbors are set as the users who have close active areas to the target user's active areas, shown as $GN_{u_i} = \{u_j \mid u_j, u_i \epsilon U, j \neq i, u_j's\ active\ areas\ are\ close\ to\ u_i's\}$.

Definition 3 (Item's active area neighbor): We call the POIs that are geographically close to the target POI its active area neighbors, shown as $GN_{p_j} = \{p_i \mid p_i, p_j \epsilon P,\ i \neq j, p_i\ is\ close\ to\ p_i\}$.

Definition 4 (Preference score): Experiments in our work are based on log datasets; therefore, we compared the prediction results with users' real check-ins to determine the model performance. The predicted preference score of user u_i for POI p_j is indicated as $\widehat{r_{u_i,p_j}}$, where $\widehat{r_{u_i,p_j}}$ is calculated as the inner product similarity between user u_i's trained final embedding and POI p_j's trained final embedding. If a recommendation system can give higher preference scores to real checked POIs than unchecked POIs, the model is considered to have good performance.

Definition 5 (User-POI subgraph construction): Based on the check-in information C, we construct a bipartite user-POI subgraph $\mathcal{G}_t = (U, P, \mathcal{E}_t)$ for time slot t, e.g., 6 to 12 o'clock, where U and P are the sets of vertices of users and POIs, respectively, and $\mathcal{E}_t$ is a set of edges for time slot t generated by check-in information C. We further define $\mathcal{G} = \{\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3, \dots, \mathcal{G}_{|T|}\}$ to denote a set of all subgraphs that cover 24 h, where $T = \{t_1, t_2, t_3, \dots, t_{|T|}\}$ is the set of time slots.

Definition 6 (User's and item's representation): Our proposed models learn the unique representation (embedding) of each user and each POI by iteratively aggregating

the information of user-item check-ins and neighbors, where e_{u_i} and e_{p_j} are the embeddings of user u_i and POI p_j, respectively. The GCN models output both e_{u_i} and e_{p_j}.

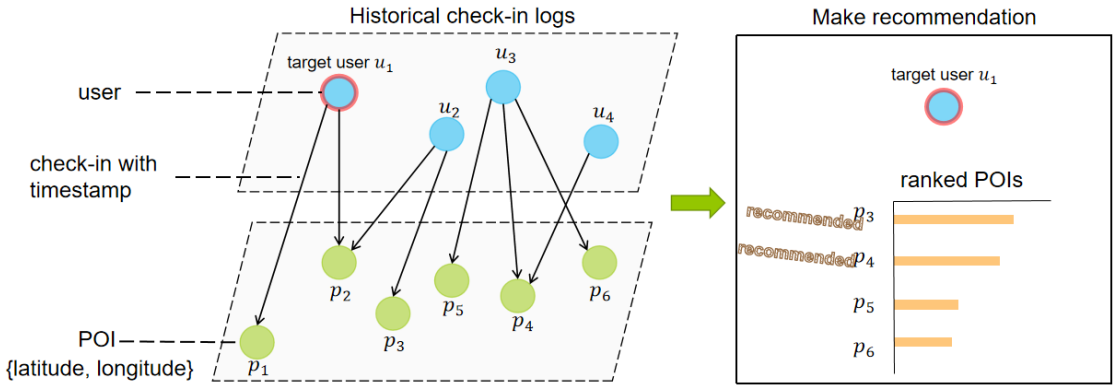


Figure 4-1: POI recommendation.

5 GN-GCN: Combining Geographical Neighbor Concept with Graph Convolution Network for POI Recommendation[12]

In this chapter, we introduce our **Contribution 2**. Although graph convolution networks (GCN) technique, like LightGCN [5], has been applied to recommendation systems to improve model performance, when the GCN models target Point-of-interest (POI) recommendation, setting up the goal of improving the accuracy of POI recommendation, how to apply the special attributes of POI - geographical information is worth considering since integrating side information always indicates higher accuracy. In this chapter, we propose a new technique to model geographical information as users' active areas. The users whose active areas are close are defined as "active area neighbors". Then, we further extend the definition of "neighbor" in GCN models to aggregate information from newly defined active area neighbors. Note that the proposed technique is lightweight, improving recommendation accuracy while keeping the model easy to train.

5.1 Introduction

By analyzing the vast amount of users' check-in history, a POI recommendation system helps users filter information and discover their interests to improve users' quality of life. To improve user satisfaction, the accuracy of the recommendation system has always been of great concern. A highly accurate recommendation system is like a close friend who understands a target user's preferences and provides insightful suggestions.

To combine the geographical information with GCN, in this chapter, inspired by the fact [27] [28] [29] that the use of power-law distribution and multi-center Gaussian

[12] This chapter is based on "GN-GCN: Combining Geographical Neighbor Concept with Graph Convolution Network for POI Recommendation"[57], by the same authors, which appeared in Proceedings of the 24th International Conference on Information Integration and Web Intelligence, pp. 153-165, 2022. Copyright(c) 2022.

distribution to model users' geographical information of visited POIs could effectively increase recommendation accuracy, we newly define active area neighbors. Then, we adopt the active area neighbors when adding geographical information of users and POIs to a GCN, enabling the efficient extraction of indirect relationships caused by their locations. Moreover, similar to LightGCN [5], we simplify the design of GCN, which improves the performance of recommendations without increasing both the trainable parameters and the model complexity.

We are the first to propose a lightweight-geographical neighbor concept-based graph convolution network (GN-GCN) model to integrate geographical information into GCN and keep the model easy to train. The contributions of this work are as follows:

- We propose a new concept called *active area neighbor* to alleviate the problem of relationship sparsity when applying GCN models to POI recommendation systems. We model user-item check-ins and active area neighbors to mine high-order connectivity to improve recommendation accuracy.

- Compared with Chang et al.'s work [3], we do not need to prepare an additional GCN to handle geographical information, enabling no increase in trainable parameters, which improves performance.

- We explore the effect of nonlinear activation functions on geographical aggregation functions because nonlinear activation functions usually have no positive impact on GCN [9]; however, nonlinear functions are often used in POI recommendation systems to model geographical information.

The rest of the chapter is organized as follows. First, Section 5.3 presents our proposed method, followed by the experiment in Section 5.4. Then, in Section 5.5, we present our conclusions.

5.2 Preliminary

This section summarizes the notations used in this chapter into Table 5-1. Note that all

of the embeddings of users and POIs have the same size of $\mathbb{R}^{1\times D}$, where D means embedding size.

Table 5-1: Notations

Notation	Definition
U	Set of users in the dataset
P	Set of POIs in the dataset
lat_{p_j}, lon_{p_j}	Geographical latitude and longitude coordinates of POI p_j
A_{u_i}	Set of active areas $\{a_{u_i,1}, a_{u_i,2}, \ldots, a_{u_i,m}\}$ of user u_i
CN_{u_i}	User u_i's check-in set
CN_{p_j}	Set of users that checked POI p_j
GN_{u_i}	User u_i's active area neighbor set
GN_{p_j}	POI p_j's active area neighbor set
$\boldsymbol{e}_{u_i}$	Final trained embedding of user u_i
$\boldsymbol{e}_{p_j}$	Final trained embedding of POI p_j
$\boldsymbol{e}_{u_i}^k$	User u_i's embedding output at layer k
$\boldsymbol{e}_{p_j}^k$	POI p_j's embedding output at layer k
$\widehat{r_{u_i,p_j}}$	Estimated preference of user u_i for POI p_j

5.3 Proposed Method

To further improve the performance of the POI recommendation system, we propose a lightweight geographical neighbor concept-based graph convolution network (GN-GCN) model for integrating geographical information on GCNs. First, we extend the definition of neighbor in GCN. In addition to check-in relations (user-checked items and items checked by users), we introduce active area neighbors for each user and each POI as newly added relations to solve the relationship sparsity problem. For each user, we define his/her active area neighbors whose active areas in the city are near to his/her active areas. Similarly, for each POI, we describe the POI's active area neighbors that are geographically close to the POI. Then, the GCN-based model learns each node's unique representation (embedding) by iteratively aggregating information from the neighbors, similar to LightGCN [5]. Compared to Chang et al.'s model [3], we do not assign

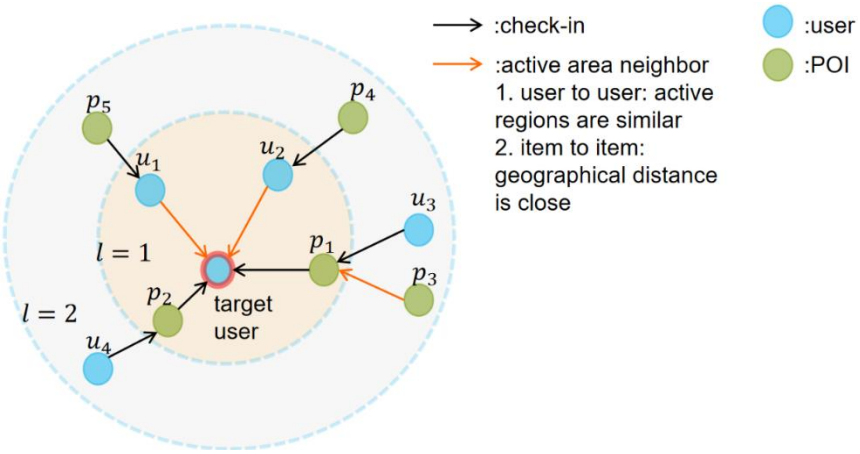


Figure 5-1: An example of the proposed GN-GCN model to aggregate high-order information from check-ins and active area neighbors.

additional trainable parameters to mine geographical information because we only have one GCN.

5.3.1 Overview

The GN-GCN model, shown in Figure 5-1, involves two steps: 1) In addition to user-item check-in interactions, we set the *active area neighbors*. We extract each user's active areas by clustering the POIs that he/she visited. The users who have at least one nearby active area (the distance between active areas is less than λ km) are set to be active area neighbors. Similarly, for the POIs whose geographical distance is less than λ km, we consider them active area neighbors by calculating the distance between all combinations of two POIs; 2) We design the neighbor-aggregation-based neural network to train the representation (embedding) of each user and each POI followed by calculating the inner product similarity between user's embedding and POI's embeddings to score. As shown in Figure 5-1, through check-ins (black arrows) and active area neighbors (orange arrows), we aggregate high-order connectivity information to the target user node. As a result, we can enhance a GCN model to adopt geographical information, which can extract high-order connectivity over collaborative filtering information.

5.3.2 Modeling Active Area Neighbor

User Active Area Neighbor

When analyzing user u_i's check-in history c_{u_i}, we first use the DBSCAN algorithm [34] to cluster user u_i's visited POIs based on POIs' geographical latitude and longitude. Among cities and in a city, users may have multiple active areas where they have frequently checked in, such as the workplace and home. After the clustering, we have a set of active areas A_{u_i} for user u_i. Then, we define user pairs whose active areas' minimum distance is less than threshold λ as active area neighbors, as in Eq. (5.1), and Eq. (5.2). When the minimum distance between the active areas of two users is less than λ, we define these two users as having similar geographical interests and are therefore defined as geographical neighbors.

$$GN_{u_i} = \left\{u_j \mid u_j \epsilon U, i \neq j, min\_dis_{u_i,u_j} < \lambda\right\} \tag{5.1}$$

$$min\_dis_{u_i,u_j} = \min\left(dis\left(A_{u_i}, A_{u_j}\right)\right) \tag{5.2}$$

$$dis\left(A_{u_i}, p_j\right) = \left\{geodis\left(a^c_{u_i,m}, p_j\right) | a_{u_i,m} \in A_{u_i}\right\}$$

where $geodis()$ is a function to receive two real-value geographical coordinates (latitude: [-90, 90], longitude: [-180, 180]) and outputs a real-value number ([0, 40075], 40075 means the longest distance between two points on Earth in km), representing the geographical distance between two points on the Earth, calculated in radians. $a^c_{u_i,k}$ is the center of POIs located in user u_i's active area $a_{u_i,k}$, where the center position is calculated by the arithmetic mean of latitude and longitude, respectively. Note that we set threshold λ as a small value so that too far away check-ins cannot be included in the same active area to prevent center shift problems. The process of calculating user active area neighbor is shown in Algorithm 4.

POI Active Area Neighbor

Similar to the user's active area neighbor, we define POI pairs whose distance is less than threshold λ as active area neighbors, shown in Eq. (5.3).

$$GN_{p_j} = \left\{ p_i \mid p_i, p_j \epsilon P, i \neq j, min\_dis_{p_j,p_i} < \lambda \right\}, \tag{5.3}$$

where $min\_dis_{p_j,p_i}$ is calculated by the distance function $geodis()$.

5.3.3 Geographical Neighbor Concept-based Graph Convolution Network (GN-GCN)

The basic idea of our proposed GN-GCN model is to extend the definition of the neighbor in a GCN. In addition to user-item check-in interactions, the GN-GCN model also mines rich geographical information from active area neighbors and learns the unique representation (embedding) for each node by smoothing features over the graph.

The architecture of the GN-GCN model is illustrated in Figure 5-2. At each layer k, we aggregate information from check-ins and active area neighbors separately. Subsequently, a simple weighted (α and β) addition operation is executed to output the representation (embedding) of each node at the layer.

The representations of users and POIs in the model are calculated by Eq. (5.4).

$$\boldsymbol{e}_{u_i}^{k+1} = \alpha * \sum_{p_j \in CN_{u_i}} \frac{1}{\sqrt{\left|CN_{u_i}\right| * \left|CN_{p_j}\right|}} e_{p_j}^{k} + \beta * \sum_{u_j \in GN_{u_i}} \frac{1}{\sqrt{\left|GN_{u_i}\right| * d_{u_i,u_j}}} \boldsymbol{e}_{u_j}^{k} \tag{5.4}$$

$$\boldsymbol{e}_{p_j}^{k+1} = \alpha * \sum_{u_i \in CN_{p_j}} \frac{1}{\sqrt{\left|CN_{p_j}\right| * \left|CN_{u_i}\right|}} \boldsymbol{e}_{u_i}^{k} + \beta * \sum_{p_i \in N_{p_j}} \frac{1}{\sqrt{\left|GN_{p_j}\right| * d_{p_j,p_i}}} \boldsymbol{e}_{p_i}^{k}$$

where $1/\sqrt{|CN_{u_i}|}\sqrt{|CN_{p_j}|}$ is a normalized discount factor as same as the standard GCN-based model [3][9][14]; α and β are parameters from 0 to 1; $1/d_{u_i,u_j}$ is the min-max normalized value of $1/min\_dis_{u_i,u_j}$ in the range of 0 to 1; $1/\sqrt{|GN_{u_i}|}$ ranges from 0 to 1; $1/d_{p_j,p_i}$ is the min-max normalized value of $1/min\_dis_{p_j,p_i}$ in the range of 0 to 1. When the left and right sides of the multiplication sign are two real numbers, "*" represents a multiplication of two real numbers. When the left and right sides of the multiplication sign are a real number and a vector, "*" denotes scalar multiplication.

We adopt $1/d_{p_j,p_i}$ instead of the number of neighbors to distinguish the importance of geographical neighbors without increasing trainable parameters.

5.3.4 Geographical Neighbor Concept-based Graph Convolution Network (GN-GCN) with Nonlinear Active Function

$$\boldsymbol{e}_{u_i}^{k+1} = \alpha * \sum_{p_j \in CN_{u_i}} \frac{1}{\sqrt{|CN_{u_i}| * |CN_{p_j}|}} \boldsymbol{e}_{p_j}^{k} + \beta * Active\left(\sum_{u_j \in GN_{u_i}} \frac{1}{\sqrt{|GN_{u_i}| * d_{u_i,u_j}}} \boldsymbol{e}_{\boldsymbol{u_j}}^{\boldsymbol{k}} \right) \tag{5.5}$$

$$\boldsymbol{e}_{p_j}^{k+1} = \alpha * \sum_{u_i \in CN_{p_j}} \frac{1}{\sqrt{|CN_{p_j}| * |CN_{u_i}|}} \boldsymbol{e}_{u_i}^{k} + \beta * Active\left(\sum_{p_i \in GN_{p_j}} \frac{1}{\sqrt{|GN_{p_j}| * d_{p_j,p_i}}} \boldsymbol{e}_{\boldsymbol{p_i}}^{\boldsymbol{k}} \right)$$

Both He et al. [5] and Chen et al. [23] mentioned that nonlinear active function has no positive effect on a GCN-based recommendation system. However, in POI recommendation systems, the relationship between user check-in probability and geographical distance is always nonlinear [30] [29], which inspires us to add a nonlinear

active function to the integrated information of active area neighbors. Based on the above intuition, our updated user's and POI's representation (embedding) are altered as Eq. (5.5). Note that "$Active$" means the $LeakyReLU$ function. In this paper, we compare Eq. (5.5) with Eq. (5.4) to confirm the effect of the nonlinear active function.

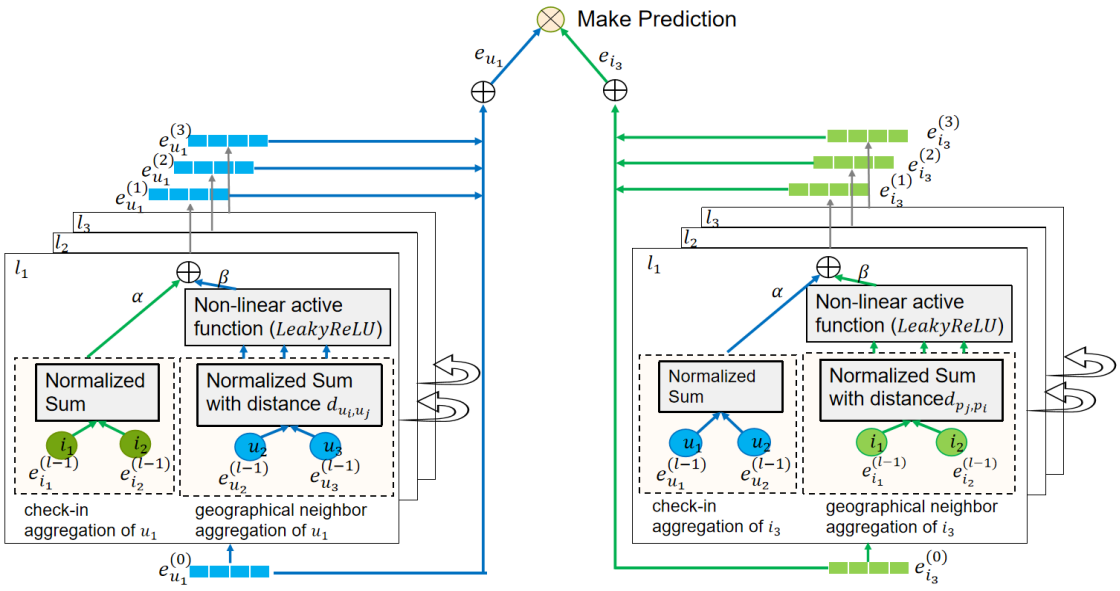


Figure 5-2: The architecture of GN-GCN model

5.3.5 Model Prediction for POI Recommendation Task

Algorithm 4: Calculating User Active Area Neighbor

Input: $C = \{c_{u_1}, c_{u_2}, c_{u_3}, \dots, c_{u_M}\}$: user check-in history
$Lat = \{lat_{p_1}, lat_{p_2}, lat_{p_3}, \dots, lat_{p_N}\}$: Latitudes of POIs
$Lon = \{lon_{p_1}, lon_{p_2}, lon_{p_3}, \dots, lon_{p_N}\}$: Longitudes of POIs
λ: Threshold

Output: $GN = \{GN_{u_1}, GN_{u_2}, GN_{u_3}, \dots, GN_{u_M}\}$: Users' active area neighbour

1 **for** $i \leftarrow 1$ *to* M **do**
2 $GN_{u_i} \leftarrow \emptyset$ ▷ initialize user u_i's active area neighbour set
3 **end for**
4 **for** $i \leftarrow 1$ *to* M **do**

5 ▷ use DBSCAN algorithm to cluster user u_i's visited POIs based on $latitude$ and $longitude$

6 $A_{u_i} \leftarrow$DBSCAN (c_{u_i})

7 **for** $k \leftarrow 1\ to\ |A_{u_i}|$ **do**

8 $a^c_{u_i,\mathrm{k}} \leftarrow$ geographical centre of the cluster k

9 **end for**

10 **end for**

11 **for** $i \leftarrow 1\ to\ M-1$ **do** ▷ enumerate each pair of users

12 **for** j$\leftarrow i+1\ to\ M$ **do**

13 ▷ calculate active areas' similarity of user pair

14 $min\_dis_{u_i,u_j} \leftarrow min\ distance$ between A_{u_i} and A_{u_j} (calculated from Eq. (5.2))

15 **if** $min\_dis_{u_i,u_j} < \lambda$ **then**

16 $GN_{u_i} \leftarrow GN_{u_i} \cup \{u_j\}$

17 $GN_{u_j} \leftarrow GN_{u_j} \cup \{u_i\}$

18 **end for**

19 **end for**

20 **return** GN

It is worth mentioning that, in our GN-GCN model, although the concept of active area neighbor is adopted compared with LightGCN, it does not cause an increase in the number of trainable parameters, which allows our method to maintain the ease of training the GCN model. Same as LightGCN [5], the trainable parameters in our GN-GCN model are the user and the POI representation (embedding) at layer 0.

After we input the embeddings of user and POI at layer 0 to the GN-GCN model and output high-layer embeddings, we adopt a weighted accumulator to calculate the final embedding of each node, shown in Eq. (5.6).

$$\boldsymbol{e}_{u_i} = \sum_{k=0}^{K} \gamma_k \boldsymbol{e}^k_{u_i}, \quad \boldsymbol{e}_{p_j} = \sum_{k=0}^{K} \gamma_k \boldsymbol{e}^k_{p_j} \tag{5.6}$$

where $\gamma_k = 1/(k+1)$, indicating the importance of embedding decreases with the layer

increasing. To complete the prediction of user u_i's interest in POI p_j, we use an inner product of the user's and POI's representation (embedding), as shown in Eq. (5.7).

$$\widehat{r_{u_i,p_j}} = \boldsymbol{e}_{u_i}{}^T \boldsymbol{e}_{p_j} \tag{5.7}$$

When the embedding of user and POI are more similar, the POI has a higher prediction rate and is ranked at the top of the recommendation list.

5.3.6 Model Training

The widely used Bayesian personalized ranking (BPR) loss [5] [23] [35] is adopted to train the GN-GCN model. BPR loss considers observed user check-ins as positive cases and assigns several negative cases (unobserved counterparts) for each positive one. The BPR loss is trained so that the positive cases are rated higher and rank ahead of negative ones, as Eq. (5.8).

$$L_{BPR} = -\sum_{u_i=1}^{M} \sum_{p_j \epsilon c_{u_i}} \sum_{p_k \epsilon\, P - c_{u_i}} ln\sigma(\widehat{r_{u_i,p_j}} - \widehat{r_{u_i,p_k}}) + \mu\|\theta\|_2^2 \tag{5.8}$$

where μ controls the importance of ℓ_2 regularization. θ indicates the trainable parameters ($e_{u_i}^0$ and $e_{p_j}^0$ at layer 0) in our model. M is the number of users in the dataset.

5.4 Experimental Evaluation

This section provides the experimental evaluation to confirm whether the proposed GN-GCN[13] model can improve the performance while keeping the number of trainable parameters the same as LightGCN [5].

[13] https://github.com/bakubonmo/Rec

5.4.1 Datasets

We chose two widely used public datasets for offline testing, Yelp and Gowalla, collected by Liu et al. [36]. Both datasets contain geographical information. The Yelp dataset contains many geo-tagged businesses among several cities, including 30,887 users, 18,995 items, and 860,888 check-ins, with a sparsity of 99.86% for the user-item check-in matrix. The Gowalla dataset contains 18,737 users, 32,510 items, and 1,278,274 check-ins, with a sparsity of 99.87% for the user-item check-in matrix. For each dataset, we follow the definition of Liu et al. [36], setting the earliest 70% of check-ins as a training set, using the latest 20% of check-ins as a testing set, and the remaining 10% as a tuning set. We summarize the statistics of two datasets in Table 5-2.

Table 5-2: The statistics of datasets

Dataset	Number of users	Number of items	Number of check-ins	Sparsity
Gowalla	18,737	32,510	1,278,274	99.87%
Yelp	30,887	18,995	860,888	99.86%

5.4.2 Baselines

In this chapter, we chose four state-of-the-art algorithms as baselines.

1) RankGeoFM [33]: We selected the RankGeoFM model as a representative non-GCN-based model to confirm a cross-sectional comparison between GCN-based and non-GCN-based models. RankGeoFM is an MF model that uses two latent matrices, the check-in preference matrix and geographical preference matrix, to represent user preferences. RankGeoFM has the best performance on POI recommendation according to a fine-grained comparative experiment conducted by Liu et al. [36].

2) LR-GCCF [23]: LR-GCCF constructs a residual network and learns the embeddings of users and items through a linear aggregation function. This model suggests that eliminating the activation function during the aggregation operation can improve the performance of the GCN model.

3) LightGCN [5]: LightGCN removes self-connection to further simplify the GCN model. In each hidden layer, the update of the embeddings depends only on the aggregated information from the neighbor nodes. In addition, the same as LR-GCCF, the nonlinear activation function is eliminated in LightGCN.

4) GPR [3]: To the best of our knowledge, GPR is the state-of-the-art GCN-based model for POI recommendation using time and geographical information. The other side information, such as social or categorical information, is not used. GPR learns two embeddings for each POI representation, incoming and outgoing influence, to explore the wealth of information contained in consecutive check-ins.

The comparison with baselines allows us to verify whether the performance of the recommendation is improved with the same number of trainable parameters as LightGCN, i.e., not increasing the parameters like GPR. Note that we do not compare with Elmi et al.'s work [37] due to the different research purposes. They focus on the next POI recommendation. That is, the ground truth contains only the last POI in chronological order of the target user.

5.4.3 Metrics

The experiment considered three widely adopted ranking metrics: $Precision@k$, $Recall@k$, and $NDCG@k$ for the top k recommended POIs. The recommendation results of each algorithm were compared with the ground-truth to calculate the performance. For these three-evaluation metrics, larger values indicate better performance.

$Precision@k$ and $Recall@k$ are two classic metrics for measuring the performance

of a recommendation system, where $Precision@k$ is the average of $Precision_{u_i}@k$, and $Recall@k$ is the average of $Recall_{u_i}@k$ for all users. $Precision_{u_i}@k$ and $Recall_{u_i}@k$ represent the probability that the recommended top$k$ items are relevant to user u_i's preference and that the items relevant to user u_i's preferences are recommended in the top-k list, respectively, defined as Eq. (5.9) and Eq. (5.10).

$$Precision_{u_i}@k = \frac{\left|RecList_{u_i}^{k} \cap GT_{u_i}\right|}{\left|RecList_{u_i}^{k}\right|} \tag{5.9}$$

$$Recall_{u_i}@k = \frac{\left|RecList_{u_i}^{k} \cap GT_{u_i}\right|}{\left|GT_{u_i}\right|} \tag{5.10}$$

where $RecList_{u_i}^{k}$ denotes user u_i's top$k$ recommended POIs, and GT_{u_i} is the ground-truth for user u_i (POIs in user u_i's testing set).

Because the dataset is divided into training and testing sets, the average number of ground truth POIs for target users in the testing set may be less than k, which results in a theoretical maximum value of precision and recall lower than 1. Therefore, we calculated the maximum precision and recall values for the two datasets. On the Gowalla dataset, the theoretical maxima were Precision@5 =0.949, Recall@5 =0.736, $Precision@10$=0.673, and $Recall@10$=0.909. On the Yelp dataset, the theoretical maximums were Precision@5=0.789, Recall@5=0.881, $Precision@10$=0.483, and Recall@10=0.961.

$NDCG@k$ [23] [5] is the average of $NDCG_{u_i}@k$, which considers the position (i.e., rank) of the recommended items, as shown in Eq. (5.11). The metric gives a higher score to the relevant items that appear at the top of the recommendation list than to those that appear at the bottom.

$$NDCG_{u_i}@k = \frac{DCG_{u_i}@k}{IDCG_{u_i}@k} \tag{5.11}$$

where $DCG_{u_i}@k$ is calculated using Eq. (5.12). $IDCG_{u_i}@k$ is the maximum and ideal value of $DCG_{u_i}@k$, indicating that the hit items are ranked at the top of the recommendation list.

$$DCG_{u_i}@k = \sum_{i=1}^{k} \frac{2^{bin_i} - 1}{log_2(i+1)} \tag{5.12}$$

where bin_i is a binary value: 1 if and only if the recommended POI at position i hits and 0 otherwise.

5.4.4 Hyperparameter Settings

Hyperparameters were tuned in the predefined ranges and set as shown in Table 5-3, except for the hyperparameters of RankGeoFM. We used the same parameters of Liu et al. [36] for RankGeoFM.

Hyperparameters for DBSCAN Algorithm

In our proposed GN-GCN model, the DBSCAN algorithm is adopted to cluster user check-ins for active regions calculation. DBSCAN algorithm has two hyper-parameters (eps and $minPts$). We search eps from 0.25 to 2 with the interval 0.25 and $minPts$ from 2 to 5 with the interval 1. Finally, we find that eps equals to 1; $minPts$ equals to 2 has the best performance. To calculated active area neighbors, we adopt a threshold λ, introduced in Section 5.3.2. λ is grid searched from the range 0.25 to 1 with the interval 0.25, and finally we set λ equals to 0.75.

Hyperparameters for Recommendation Algorithms

The only trainable parameters in our model are the embedding of the user and POI at layer 0. We fix the embedding size as 64 and initialized the embedding by using a Gaussian distribution with a mean of 0 and a standard deviation of $1e^{-2}$. The learning rate is searched from $\{1e^{-4}, 1e^{-3}, 1e^{-2}, 1e^{-1}\}$ and we find that the learning rate equals to $1e^{-2}$

has the best performance. The regularization coefficient μ is searched in the range $\{1e^{-6}, 1e^{-5}, \ldots, 1e^{-1}\}$ and finally we set μ equals to $1e^{-4}$. We have another two important hyper-parameters, α and β to balance the importance of check-ins and active area neighbors. We adopt grid search in the range from 0 to 1 with the interval 0.25 and find that α equals to 0.5; β equals to 1 has the best performance. We also set the hyperparameters for the baseline algorithms by using the same strategy to reach the optimal performance for a fair comparison.

5.4.5 Experimental Results

Table 5-4 and Table 5-5 show the experimental results on the Yelp and Gowalla datasets, respectively. We abbreviate $Precision@k$ as $P@k$, $Recall@k$ as $R@k$, and

Table 5-3: Grid Search of Hyper-parameters. Grid Search of Hyper-parameters

Algorithms	Hyper-parameters setting		
	Hyper-parameter	Search-range description	Adopted value
DBSCAN [34]	eps	$\{0.25, 0.5, 0.75, \ldots, 2\}$	1 km
	$minPts$	$\{2, 3, 4, 5\}$	2
active area neighbor threshold λ		$\{0.25, 0.5, 0.75, 1\}$	0.75 km
LR-GCCF [23], LightGCN [5], GPR [3], GN-GCN (proposed)	embedding size	same as Chen et al. [23] and He et al. [5]	64
	embedding initialization	same as Chen et al. [23] and He et al. [5]	Gaussian dist. (Mean:0, SD:$1e^{-2}$)
	learning rate	$\{1e^{-4}, 1e^{-3}, 1e^{-2}, 1e^{-1}\}$	$1e^{-2}$
	regularization coefficient μ	$\{1e^{-6}, 1e^{-5}, \ldots, 1e^{-1}\}$	$1e^{-4}$
GN-GCN (proposed)	check-in coefficient α	$\{0, 0.25, 0.5, 0.75, 1\}$	0.5
	geographical coefficient β	$\{0, 0.25, 0.5, 0.75, 1\}$	1

$NDCG@k$ as $N@k$. The proposed methods with and without the nonlinear active function mentioned in Section 5.3.3 and Section 5.3.4 are also compared, where GN-GCN represents our method without the nonlinear active function, whereas GN-GCN+Active represents the method with the nonlinear active function. The highest scores are noted boldly. The maximum values other than those of our proposed models are underlined. We adopt a two-tailed paired t-test to confirm the statistically significant improvement of our proposed method over the baselines.

Table 5-4: Evaluation Result on Yelp Dataset

Algorithms		P@5	R@5	N@5	P@10	R@10	N@10
Baselines	RankGeoFM[33]	0.0320	0.0304	0.0332	0.0273	0.0541	0.0300
	LR-GCCF [23]	0.0286	0.0350	0.0342	0.0252	0.0519	0.0420
	GPR [3]	0.0366	0.0382	0.0386	**0.0321**	0.0577	0.0345
	LightGCN [5]	0.0385	0.0453	0.0468	0.0303	0.0562	0.0507
Proposed	GN-GCN	0.0393*	0.0467*	0.0475+	0.0311*	**0.0590***	0.0523*
	GN-GCN+Active	**0.0396***	**0.0469***	**0.0486***	0.0312*	0.0588*	**0.0529***

* Statistically significant for p<0.01 when comparing with all baselines.

+ Statistically significant for p<0.05 when comparing with all baselines.

Table 5-5: Evaluation Result on Gowalla Dataset

Algorithm		P@5	R@5	N@5	P@10	R@10	N@10
Baselines	RankGeoFM[33]	0.0684	0.0479	0.0719	0.0559	0.0755	0.0622
	LR-GCCF [23]	0.0640	0.0669	0.0721	0.0504	0.0762	0.0749
	GPR [3]	0.0775	0.0580	0.0671	**0.0640**	0.0881	0.0613
	LightGCN [5]	0.0760	0.0788	0.0859	0.0597	0.0865	0.0870
Proposed	GN-GCN	**0.0784***	**0.0815***	0.0883*	0.0614*	0.0894*	0.0896*
	GN-GCN+Active	0.0782*	0.0812*	**0.0886***	0.0619*	**0.0898***	**0.0900***

* Statistically significant for p<0.01 when comparing with all baselines.

The results show that our proposed model obtains the highest performance for most metrics. Our method successfully improves $Recall@5$ from 0.0453 to 0.0469 (3.53%)

and $Recall@10$ from 0.0562 to 0.0590 (4.98%) on the Yelp dataset, while improving $Recall@5$ from 0.0788 to 0.0815 (3.43%) and $Recall@10$ from 0.0865 to 0.0898 (3.82%) on the Gowalla dataset, compared with LightGCN.

On both datasets, comparing GN-GCN+Active and GN-GCN, we cannot confirm a statistically significant difference between GN-GCN+Active and GN-GCN, which concludes that the nonlinear active function cannot affects the performance.

5.4.6 Discussion on the Number of Trainable Parameters

Same as LightGCN, the trainable parameters in our GN-GCN model are the embeddings of users and items at layer 0, even with integrating geographical information. Since we set the embedding size to 64, both the LightGCN model and our GN-GCN model have 64*(M+N) trainable parameters, where M and N are the numbers of users and items, respectively. On the contrary, the GPR model has more than 64*(M+2N) trainable parameters, which does not include the trainable transformation matrixes (64*64) [3].

To further improve the performance of neural network recommendation systems, integrating multiple information, such as geographical information, is potentially feasible. However, integrating multiple information tends to introduce more trainable parameters, making the model more difficult to train and reducing its practicability. Thus, keeping the number of trainable parameters not increasing is indispensable. As shown in Table 5-4 and Table 5-5, although GPR integrates geographical information, $NDCG@10$ is smaller than LightGCN which does not integrate geographical information, which shows the difficulty in training.

5.5 Conclusion

We proposed the GN-GCN model to mine users' active areas and integrate geographical information into GCNs in a lightweight manner by adopting a new concept called active area neighbors. Our experimental evaluation confirms that our model

outperforms all the baselines. Compared with LightGCN, $Recall@10$ improves from 0.0562 to 0.0590 (4.98%) on the Yelp dataset and from 0.0865 to 0.0898 (3.82%) on the Gowalla dataset.

6 EPT-GCN: Edge Propagation-based Time-aware Graph Convolution Network for POI Recommendation[14]

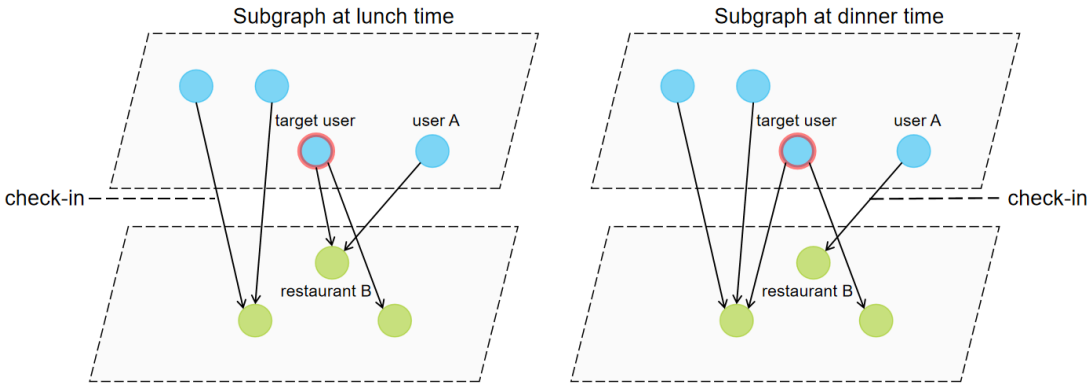
In this chapter, we introduce our **Contribution 3**, an edge propagation technique to model time information. For POI recommendation, time information expresses users' interest at different time slots of the day. How to model users' time-based preference is the key to integrating time information to graph convolution network (GCN) models. Existing GCN-based techniques simply adopt time information by modeling users' check-in sequences, which is insufficient and ignores users' time-based high-order connectivity. Note that time-based high-order connectivity refers to the relationship between indirect neighbors with similar preferences in the same time slot. In this paper, we propose a new time-aware GCN model to extract rich collaborative signals contained in time information. Our work is the first to divide user check-ins into multiple subgraphs, i.e., time slots, based on time information. We further propose an edge propagation module to adjust edge affiliation, where edges represent check-ins, to propagate the user's time-based preference to multiple time slots. The propagation module is based on an unsupervised learning algorithm and does not require additional ground-truth labels. This work is the first to cast time slots into multiple subgraphs to improve POI recommendation accuracy. After modeling time information, we further combine time and geographical information to jointly assist the GCN in filtering high-order collaborative signals.

6.1 Introduction

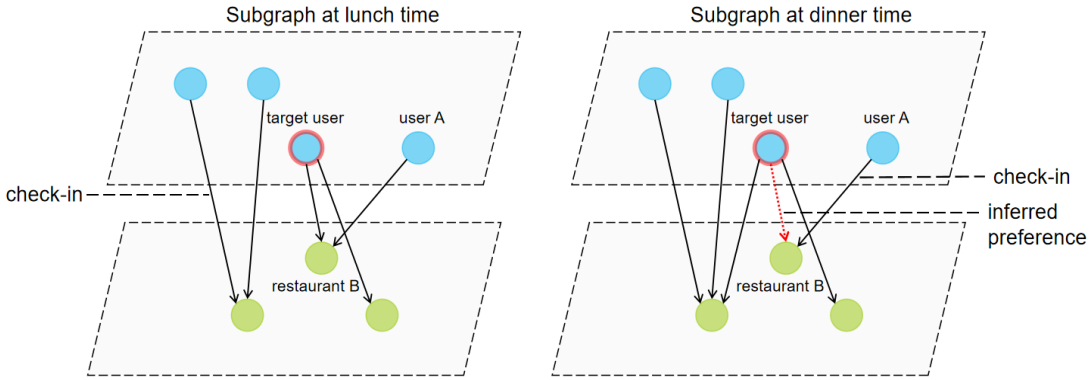
In addition to geographical information, time information is also an important side information for improving POI recommendation performance. To combine time

[14] This chapter is based on "EPT-GCN: Edge propagation-based time-aware graph convolution network for POI recommendation"[69], by the same authors, which appeared in Neurocomputing, 543, 126272, pp. 1-15, 2023. Copyright(c) 2023.

information with GCN, similar to IMP-GCN [2], a subgraph technique is well-matched with time slot disentanglement to solve the problem of learning unique time-based embeddings for users and items. A simple method is to divide the edges (check-ins) into individual subgraphs (time slots) according to timestamps, e.g., four 6-hour intervals over 24 hours; however, a monotonous subgraph division has drawbacks. Monotonous subgraph division cannot propagate the learned time preference features over multiple time slots because the subgraphs are constructed in advance, and there is no reconstruction module to transmit the time-based preference. In Figure 6-1, for example, say a preference feature has been extracted that both the target user and user A prefer to check in at Chinese restaurant B at lunchtime. At the same time, another preference feature has been extracted in a different subgraph: user A visits the same restaurant B at dinner time. Although the target user has no experience checking in to restaurant B for dinner, we could infer that it is a good choice. The transmitted edge is indicated by the red dotted line in Figure 6-1. Therefore, an edge-based (check-in-based) propagation and subgraph reconstruction mechanism are important for automatically transmitting users' time-based preferences. However, existing node-based subgraph construction techniques cannot be directly applied to POI time information mining. In IMP-GCN [2], each node (user or item) is clustered into one subgraph by an unsupervised learning module. However, a POI can have multiple suitable time slots for the target user to check in. In this paper, we propose an edge propagation module to repartition user check-ins and reconstruct subgraphs based on the similarity between the learned users' high-order time slot embeddings and the POI's initial layer embedding. Unlike IMP-GCN, our method is an edge-oriented (check-in-oriented) algorithm that does not focus on nodes.



(a) without transmitting users' time-based preferences



(b) transmitting users' time-based preferences

Figure 6-1: Example of check-in-based propagation to transmit users' time-based preferences.

The contributions of this chapter can be summarized as follows:

- To better exploit time information, we are the first to combine time information with the GCN subgraph technique for POI recommendation, namely, subgraph mining GCN (SGM-GCN).

- We propose a trainable diagonal matrix and attention mechanism to compute the disentangled embeddings of users and POIs for each time slot.

- We propose a novel edge sampling-based propagation algorithm to adjust the subgraph structure to improve the propagation ability of the SGM-GCN.

The remainder of this paper is organized as follows. In Section 6.2, we present the notations used in this chapter. Our main idea, including the structure of the model, is described in Section 6.3, followed by the experiment in Section 6.4, where we introduce the dataset and compare the results with those of state-of-the-art models. Finally, the conclusions of this contribution are presented in Section 6.5.

6.2 Preliminary

Table 6-1: Notations

Notation	Definition
U	Set of users in the dataset, where $U = \{u_1,\ u_2,\ u_3, \dots, u_M\}$
P	Set of POIs in the dataset, where $P = \{p_1,\ p_2,\ p_3, \dots, p_N\}$
lat_{p_j}, lon_{p_j}	Geographical latitude and longitude coordinates of POI p_j
C_{u_i}	User u_i's check-in logs, where $C_{u_i} = \{(p_j, timestamp) \mid u_i \in U, p_j \in P,\ u_i$ checked $p_j\}$
$\mathcal{G}_t = (U, P, \mathcal{E}_t)$	User-POI check-in subgraph for time slot t
$\mathcal{E}_t$	Edge set for time slot t in subgraph $\mathcal{G}_t$
T	Set of time slots, where $T = \{t_1, t_2, t_3, \dots, t_{\lvert T\rvert}\}$
$\mathcal{G}$	Set of user-POI check-in subgraphs, where $\mathcal{G} = \{\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3, \dots, \mathcal{G}_{\lvert T\rvert}\}$
ε_{u_i, p_j}	Edge in $\mathcal{G}$, where the two ends are user u_i and POI p_j
$CN_{u_i}^t$	Extracted set of user u_i's check-in neighbors at time slot t
$CN_{p_j}^t$	Extracted set of neighbor users that checked POI p_j at time slot t
CN_{u_i}	User u_i's check-in neighbors in the entire check-in set C_{u_i}
CN_{p_j}	Set of neighbor users that checked POI p_j in C
$\boldsymbol{e}_{u_i}^{Init}$	User u_i's embedding at initial layer
$\boldsymbol{e}_{p_j}^{Init}$	POI p_j's embedding at initial layer
$\boldsymbol{e}_{u_i}^{(t,k)}$	User u_i's embedding at layer k for subgraph $\mathcal{G}_t$

$\boldsymbol{e}_{p_j}^{(t,k)}$	POI p_j's embedding at layer k for subgraph $\mathcal{G}_t$
$\boldsymbol{e}_{u_i}^{t}$	Learned user u_i's time slot embedding for subgraph $\mathcal{G}_t$
$\boldsymbol{e}_{p_j}^{t}$	Learned POI p_j's time slot embedding for subgraph $\mathcal{G}_t$
$\boldsymbol{e}_{u_i}^{Time}$	Learned user u_i's time embedding, calculated from $e_{u_i}^{1}$, $e_{u_i}^{2}$, …, $e_{u_i}^{\lvert T\rvert}$
$\boldsymbol{e}_{p_j}^{Time}$	Learned POI p_j's time embedding, calculated from $e_{p_j}^{1}$, $e_{p_j}^{2}$, …, $e_{p_j}^{\lvert T\rvert}$
$\boldsymbol{e}_{u_i}$	Final learned embedding of user u_i
$\boldsymbol{e}_{p_j}$	Final learned embedding of POI p_j
$\widehat{r_{u_i,p_j}}$	Predicted preference score of user u_i for POI p_j, where p_j is user u_i's unchecked POI

In this section, we summarize the notations used in this chapter in Table 6-1. Note that all of the embeddings of users and POIs have the same size of $\mathbb{R}^{1\times D}$, where D means embedding size.

6.3 Proposed Method

This section proposes three new time-aware GCNs for POI recommendation: 1) time-aware subgraph mining GCN (SGM-GCN), 2) edge propagation-based time-aware GCN (EPT-GCN), and 3) EPT-GCN+Geo, which integrates EPT-GCN with geographical information.

6.3.1 Overview

Previous time-aware POI recommendation models [3] [4] successfully extracted time-related user and POI features to improve the performance of POI recommendation; however, they struggle to mine users' time-based high-order connectivity, i.e., the relationship among time slots, even adopting state-of-the-art GCN models. Our idea to tackle the above problem is to adopt subgraph models [2] [52] [53] but to modify the

subgraph construction method to propagate users' time-based preferences to multiple time slots, where preference is reflected as check-ins.

Our proposed time-aware subgraph mining GCN (SGM-GCN) in Section 6.3.2 enables the extraction of user time-based high-order connectivity by partitioning users' edges (check-ins) into multiple time slots and constructing multiple subgraphs. Then, time-aware collaborative signals are propagated in different subgraphs and contribute to learning the embeddings of users and POIs in that time slot. Following SGM-GCN, we propose the edge propagation-based time-aware GCN (EPT-GCN) to propagate the edges (check-ins) and reconstruct subgraphs in Section 6.3.3; thereby we could reduce the loss of information, such as implicit feedback, including clicks and check-ins. Finally, we integrate geographical distance into the EPT-GCN to construct GCT-GCN+Geo in Section 6.3.4. After learning the embeddings of users and POIs, we adopt inner product similarity to calculate the prediction score to recommend the top *k* items to a target user.

6.3.2 Time-aware Subgraph Mining Graph Convolution Network (SGM-GCN)

The basic idea of time-aware subgraph mining GCN (SGM-GCN) is to learn the disentangled embeddings of users and POIs in different time slots through a graph convolution operation on the corresponding subgraph. Our proposed SGM-GCN enables the extraction of user time-based high-order connectivity by partitioning user check-ins into multiple time slots. When constructing the subgraph, the user's check-in time is used hourly. We partition 24 hours into a set of time slots with equal intervals, e.g., 6 hours, with a one-to-one correspondence between the time slot and subgraph. The subgraph of time slot t ($\mathcal{G}_t$) consists of a set of user nodes (U), POI nodes (P), and edges in time slot t ($\mathcal{E}_t$), written as $\mathcal{G}_t = (U, P, \mathcal{E}_t)$ Unlike previously proposed node-based subgraph construction techniques [2], we adopt an edge-based subgraph technique, enabling an edge to appear in multiple subgraphs to generalize users' time-based preferences to multiple time slots. Finally, we obtain a set of subgraphs, each with a different set of edges.

Figure 6-2 shows the overall architecture of the SGM-GCN model, including the graph convolution layer, attention layer, and prediction-making step. First, the SGM-GCN model initializes the embeddings of users and POIs ($e_{u_i}^{(init)}$ and $e_{p_j}^{(init)}$) using a Gaussian distribution. Because we assign an initialized embedding ($e_{u_i}^{(init)}$ and $e_{p_j}^{(init)}$) to each user and POI, i is in the range of 0 to $M-1$, while j in the range of 0 to $N-1$, where M and N are user number and POI number in dataset, respectively. Second, user (POI) nodes aggregate information from check-ins and output time slot embeddings $e_{u_i}^t$ ($e_{p_j}^t$) for the subgraph $\mathcal{G}_t$. The attention layer mines the importance of time slots to represent user preferences and finally integrates users' (POIs') time embeddings into the final embedding representations e_{u_i} (e_{p_j}).

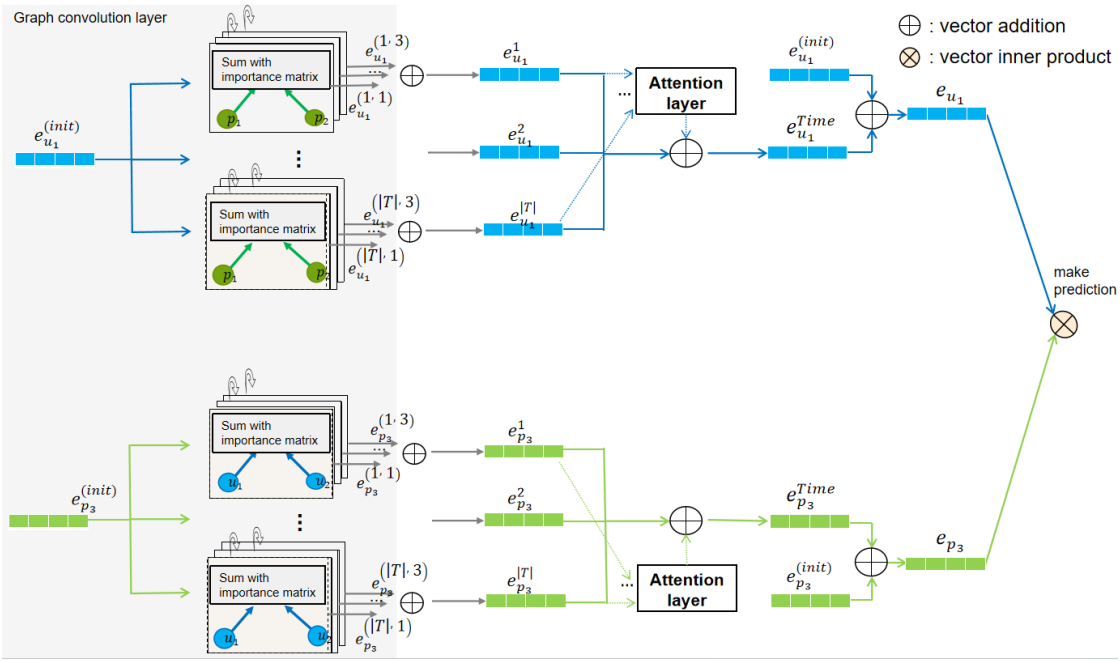


Figure 6-2: Architecture of SGM-GCN model.

For a subgraph $\mathcal{G}_t$ with edges $\mathcal{E}_t$ at each layer k, information is aggregated from neighboring nodes and is output as learned embeddings $e_{u_i}^{(t,k+1)}$ and $e_{p_j}^{(t,k+1)}$. Our subgraph mining enables us to mine the importance of embedding dimensions over different time slots by proposing a diagonal importance matrix $\boldsymbol{I^t}$ for time slot t, where

only the diagonal elements are nonzero, shown in Eq. (6.1).

$$\boldsymbol{I}^t = \begin{pmatrix} \theta_t^1 & 0 & \cdots & 0 \\ 0 & \theta_t^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \theta_t^D \end{pmatrix} \tag{6.1}$$

where θ_t^d is the $d-th$ trainable element in the matrix $\boldsymbol{I}^t \in \mathbb{R}^{D\times D}$. As shown in Figure 6-3, a diagonal importance matrix is prepared for each time slot. In a subgraph, the importance matrix is shared between users and POIs and between multiple layers. Following the embedding size setting of related GCN models [5] [11] [33], user and POI embeddings have the same embedding size D. Therefore, sharing the importance matrix can be applied to generate disentangled embeddings by time slot units.

The propagation rule between the graph convolution layers is expressed by Eq. (6.2). Note that $e_{u_i}^{(t,1)}$ ($e_{p_j}^{(t,1)}$) is calculated from $e_{p_j}^{(init)}$ ($e_{u_i}^{(init)}$).

$$\boldsymbol{e}_{u_i}^{(t,k+1)} = \sum_{p_j \in CN_{u_i}^t} \frac{1}{\sqrt{\left|CN_{u_i}\right| * \left|CN_{p_j}\right|}} \boldsymbol{e}_{p_j}^{(t,k)} \boldsymbol{I}^t \tag{6.2}$$

$$\boldsymbol{e}_{p_j}^{(t,k+1)} = \sum_{u_i \in CN_{p_j}^t} \frac{1}{\sqrt{\left|CN_{p_j}\right| * \left|CN_{u_i}\right|}} \boldsymbol{e}_{u_i}^{(t,k)} \boldsymbol{I}^t$$

where $1/\sqrt{\left|CN_{u_i}\right| * \left|CN_{p_j}\right|}$ is a normalized discount factor controlling the amount of aggregated information from a neighbor node, $CN_{u_i}^t$ is defined as user u_i's checked POIs set at time slot t, $\left|CN_{u_i}\right|$ shows the number of POIs checked by user u_i in the entire check-in set C_{u_i}, and $\boldsymbol{I}^t$ is an importance matrix with trainable diagonal elements and 0 in the remaining positions.

After obtaining the output of each layer, we adopt a weighted summation operation to

calculate the user and POI time slot embeddings $\boldsymbol{e}_{u_i}^t$ and $\boldsymbol{e}_{p_j}^t$, shown in Eq. (6.3).

$$\boldsymbol{e}_{u_i}^t = \sum_{k=1}^{K} \alpha_k \, \boldsymbol{e}_{u_i}^{(t,k)}, \quad \boldsymbol{e}_{p_j}^t = \sum_{k=1}^{K} \alpha_k \, \boldsymbol{e}_{p_j}^{(t,k)} \tag{6.3}$$

where $\alpha_k = 1/(k+1)$, indicating that the importance of layer-outputted embeddings decreases as the number of layers increases.

As users may have multiple active time slots [55], each time slot has different degrees of importance to the user. Here, an attention layer is used to obtain the time embeddings of users, with the architecture shown in Figure 6-4. The attention layer of POIs has the same architecture as that of the users. After passing through multiple MLP layers, important factors are generated as Eq. (6.4).

$$\beta_{u_i}^{t\,\prime} = \boldsymbol{V}_u^{2\,T}(\tanh(\boldsymbol{W}_u^1 \boldsymbol{e}_{u_i}^{t\;T} + \boldsymbol{b}_u)) \tag{6.4}$$

$$\beta_{p_j}^{t\,\prime} = \boldsymbol{V}_p^{2\,T}(\tanh(\boldsymbol{W}_p^1 \boldsymbol{e}_{p_j}^{t\;T} + \boldsymbol{b}_p))$$

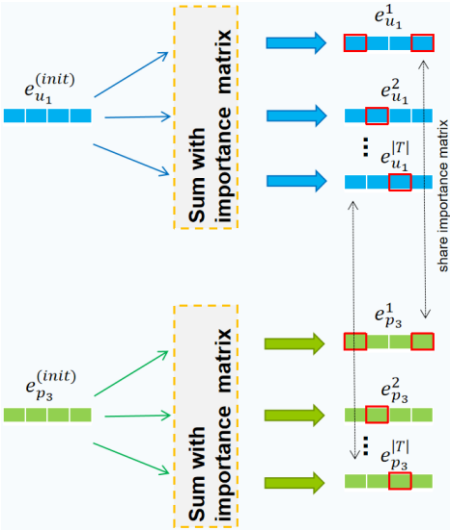


Figure 6-3: Leaning disentangled embeddings with importance matrix.

where $\boldsymbol{W}_u^1$, $\boldsymbol{W}_p^1 \in \mathbb{R}^{D\times D}$; $\boldsymbol{V}_u^2$, $\boldsymbol{V}_p^2 \in \mathbb{R}^{D\times 1}$ are trainable matrices and vectors for dimensional transformation; $b_u, b_p \in \mathbb{R}^{D\times 1}$ are trainable bias vectors; and D is the embedding size, which is set as a hyperparameter. In the first layer of the transformation, we adopt the widely used hyperbolic tangent activation function $\tanh(x) = \frac{\exp(x)-\exp(-x)}{\exp(x)+\exp(-x)}$, the same as Jiang et al.'s work [56].

The proposed attention mechanism requires normalization to generate the final importance factors, formulated as Eq. (6.5).

$$\beta_{u_i}^t = \frac{\exp(\beta_{u_i}^{t\prime})}{\sum_{t'=1}^{t'=|T|}\exp(\beta_{u_i}^{t'\prime})} \quad ,\beta_{p_j}^t = \frac{\exp(\beta_{p_j}^{t\prime})}{\sum_{t'=1}^{t'=|T|}\exp(\beta_{p_j}^{t'\prime})} \tag{6.5}$$

The output time embeddings ($\boldsymbol{e}_{u_i}^{Time}$ and $\boldsymbol{e}_{p_j}^{Time}$) are calculated by multiplying the learned importance factors and time slot embeddings, as shown in Eq. (6.6). After exploiting high-order time-aware connectivity, we adopt a simple summation to generate the final embeddings of nodes (users and POIs), formulated as Eq. (6.7).

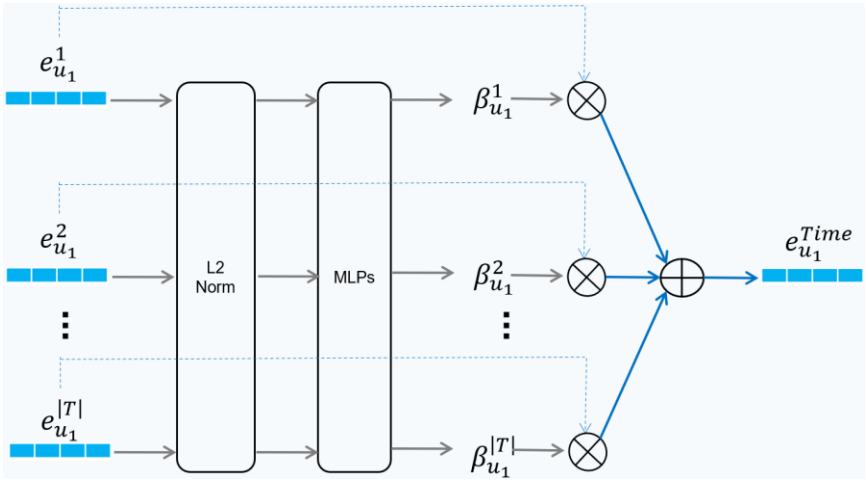


Figure 6-4: Architecture of user's attention layer

$$\boldsymbol{e}_{u_i}^{Time} = \sum_{t=1}^{|T|}\beta_{u_i}^t\,\boldsymbol{e}_{u_i}^t \quad ,\boldsymbol{e}_{p_j}^{Time} = \sum_{t=1}^{|T|}\beta_{p_j}^t\,\boldsymbol{e}_{p_j}^t \tag{6.6}$$

$$\boldsymbol{e}_{u_i} = \boldsymbol{e}_{u_i}^{Init} + \boldsymbol{e}_{u_i}^{Time} \quad , \boldsymbol{e}_{p_j} = \boldsymbol{e}_{p_j}^{Init} + \boldsymbol{e}_{p_j}^{Time}. \tag{6.7}$$

To predict the user preference score for a candidate unchecked POI, inner product similarity is a suitable metric, as used in related works [23] [5] [2]. In our work, we also used inner product similarity as the final output of the model. The inner product similarity causes each dimension of the two embeddings bitwise multiply and accumulate, as shown in Eq. (6.8).

$$\widehat{r_{u_i,p_j}} = \boldsymbol{e}_{u_i}{}^{T} \boldsymbol{e}_{p_j}. \tag{6.8}$$

Based on the preference scores, we rank the candidate POIs and recommend the top k POIs as results for the target user.

6.3.3 Edge Propagation-based Time-aware Graph Convolution Network (EPT-GCN)

Edge (check-in) propagation-based time-aware GCN (EPT-GCN) adds a propagation module based on SGM-GCN with the architecture as shown in Figure 6-5. The basic idea of EPT-GCN is to propagate a user's check-ins to multiple time slots and reconstruct subgraphs to reduce the loss of information. For a checked POI, we aim to find multiple suitable time slots for the target user. To achieve this, for a checked POI p_j, we calculate the target user's time-based preference score for POI p_j in each time slot, followed by calculating the average preference score. We extract the time slots with above-average preference scores and set these time slots as "suitable" for the target user to check POI p_j.

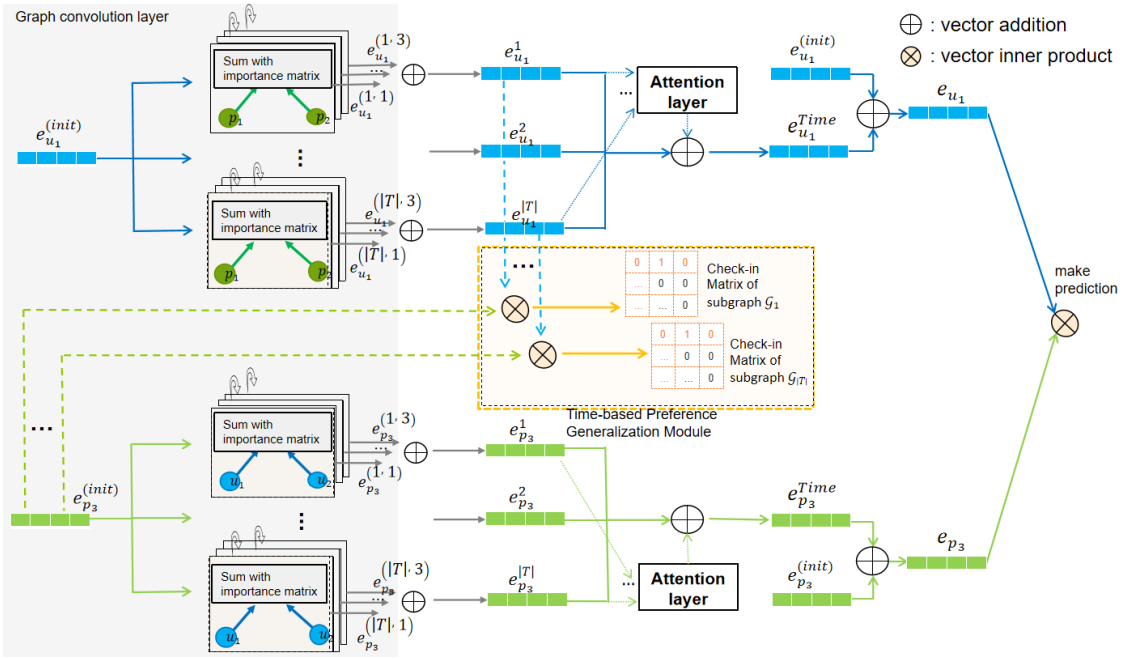


Figure 6-5: Architecture of EPT-GCN model.

Our proposed edge propagation module is executed in each training epoch to reduce the risk of over-smoothing. The variance of the final learned user (item) embeddings increased from 0.545 (0.423) to 0.752 (0.500) on the New York dataset and increased from 0.217 (0.104) to 0.356 (0.178) on the Gowalla dataset, compared with LightGCN [5]. In edge propagation, we sample s edges (e.g., $\frac{1}{25}$ of all edges in $\mathcal{G}$) and cluster them into subgraphs based on the similarity between users' high-order time slot embeddings and POIs' initial embeddings. A higher similarity indicates a higher preference score.

Because the two ends of an edge (check-in) are a user and POI, we can simply estimate the preference by calculating the similarity of these two ends ($\boldsymbol{e}_{u_i}^{t}$ and $\boldsymbol{e}_{p_j}^{Init}$). Edges (check-ins) are propagated to subgraphs (time slots) with a similarity higher than the average score, as shown in Eq. (6.9), based on which we reconstruct subgraph $\mathcal{G}_t$. For example, assume that there are four time slots. The similarity between a target user's high-order time slot embeddings and a supermarket's initial embedding is 0.1, 0.5, 0.4, and 0.2 for the four time slots, respectively. Therefore, the average score is 0.3. Then, the target user's check-in for the supermarket will be propagated to time slot-2 and time slot-3,

which means that the supermarket is suitable for the target user to check in both time slot-2 and time slot-3.

$$\mathcal{E}_t = \left\{ \begin{matrix} \varepsilon_{u_i, p_j} | \, sim^t_{u_i, p_j} > avg_{u_i, p_j} \\ , \varepsilon_{u_i, p_j} \text{ is sampled from edges in } \mathcal{G} \end{matrix} \right\} \tag{6.9}$$

where $\mathcal{E}_t$ denotes all edges of subgraph $\mathcal{G}_t$, used to reconstruct the subgraph. ε_{u_i, p_j} denotes that an edge in $\mathcal{G}$ connecting user u_i and POI p_j, not related to time slot t; $sim^t_{u_i, p_j}$ means the inner product similarity between user u_i's high-order time slot embedding and POI p_j's initial embedding, calculated using Eq. (6.10); and avg_{u_i, p_j} is obtained by calculating the average similarity, as shown in Eq. (6.11).

$$sim^t_{u_i, p_j} = \boldsymbol{e}^{t\ T}_{u_i} \boldsymbol{e}^{Init}_{p_j} \tag{6.10}$$

$$avg_{u_i, p_j} = \frac{\sum_{t'=1}^{|T|} sim^{t'}_{u_i, p_j}}{|T|} \tag{6.11}$$

Note that the proposed technique is an unsupervised module, so it does not require additional ground-truth label assistance. After training the first epoch to generate the underlying time slot embeddings, the propagation module is activated. Simultaneously, the sampled edges with similar intent are propagated to the appropriate time slots, affecting the aggregation path of information in the next epoch.

6.3.4 Combination of EPT-GCN with Geographical Information (EPT-GCN+ Geo)

We combine time and geographical information to jointly assist the GCN in filtering high-order collaborative signals; thereby, we can improve the performance of POI recommendation, where such a combination has been experimentally confirmed to improve performance [3] [4].

To integrate the geographical information, we simplify our technique proposed in work [57] of mining geographical information, which calculates the active area neighbors of users and POIs. We first adopt the DBSCAN algorithm [34] to cluster user u_i's visited POIs based on the latitude and longitude of the POIs, followed by calculating the user's active areas A_{u_i} in the same city. It should be noted that a user may have multiple active areas. Compared with the technique introduced in Chapter 5, we simplify the geographical distance similarity between user and POI as the shortest distance between the user's active areas and the POI, as Eq. (6.12).

$$sim\_geo_{u_i,p_j} = \frac{1}{\min\left(dis(A_{u_i},\ p_j)\right)} \tag{6.12}$$

$$dis\left(A_{u_i},\ p_j\right) = \left\{geodis\left(a^c_{u_i,m},\ p_j\right) | a_{u_i,m} \in A_{u_i}\right\}$$

where $geodis()$ is a function which receives two real-value geographical coordinates (latitude: [-90, 90], longitude: [-180, 180]) and outputs a real-value number ([0, 40075], 40075 means the longest distance between two points on Earth in km), representing the distance between two points on Earth in radians; $a^c_{u_i,m}$ indicates the center of latitude and longitude of POIs located in user u_i's active area $a_{u_i,m}$, as shown in Eq. (6.13).

$$a^c_{u_i,m} = \left(avg_{\forall p_j \epsilon a_{u_i,m}}\left(lat_{p_j}\right), avg_{\forall p_j \epsilon a_{u_i,m}}\left(lon_{p_j}\right)\right) \tag{6.13}$$

where $avg()$ is the function used to calculate the average.

After calculating the geographical distance similarity, we rewrite the aggregation function shown in Eq. (6.2) to apply it to the integration of geographical information, formulated as Eq. (6.14).

$$\boldsymbol{e}_{u_i}^{(t,k+1)} = \sum_{p_j \in CN_{u_i}^t} \left(\gamma * \frac{1}{\sqrt{\left|CN_{u_i}\right| * \left|CN_{p_j}\right|}} \boldsymbol{e}_{p_j}^{(t,k)} + (1-\gamma) * \frac{s_{u_i,p_j}}{\sqrt{\left|CN_{u_i}\right|}} \boldsymbol{e}_{p_j}^{(t,k)} \right) \boldsymbol{I}^t \tag{6.14}$$

$$= \sum_{p_j \in CN_{u_i}^t} \frac{1}{\sqrt{\left|CN_{u_i}\right|}} \left(\frac{\gamma}{\sqrt{\left|CN_{p_j}\right|}} + (1-\gamma)\, s_{u_i,p_j} \right) \boldsymbol{e}_{p_j}^{(t,k)} \boldsymbol{I}^t$$

$$\boldsymbol{e}_{p_j}^{(t,k+1)} = \sum_{u_i \in CN_{p_j}^t} \left(\gamma * \frac{1}{\sqrt{\left|CN_{p_j}\right| * \left|CN_{u_i}\right|}} \boldsymbol{e}_{u_i}^{(t,k)} + (1-\gamma) * \frac{s_{u_i,p_j}}{\sqrt{\left|CN_{p_j}\right|}} \boldsymbol{e}_{u_i}^{(t,k)} \right) \boldsymbol{I}^t$$

$$= \sum_{u_i \in CN_{p_j}^t} \frac{1}{\sqrt{\left|CN_{p_j}\right|}} \left(\frac{\gamma}{\sqrt{\left|CN_{u_i}\right|}} + (1-\gamma) s_{u_i,p_j} \right) \boldsymbol{e}_{u_i}^{(t,k)} \boldsymbol{I}^t$$

where γ is a hyperparameter which controls the balance of the weight of check-in information and geographical information, ranging from 0 to 1. Following previous work [57], s_{u_i,p_j} is calculated from a min-max normalization operation on the similarity sim_geo, as shown in Eq. (6.15).

$$s_{u_i,p_j} = \frac{sim\_geo_{u_i,p_j} - \min(sim\_geo_{u_i})}{\max(sim\_geo_{u_i}) - \min(sim\_geo_{u_i})} \tag{6.15}$$

where $max(sim\_geo_{u_i})$ and $min(sim\_geo_{u_i})$ indicate the maximum and minimum values of the geographical distance similarity between user u_i and the checked POIs in C_{u_i}, respectively.

6.3.5 Model Training

We adopt Bayesian personalized ranking (BPR) loss, formulated as Eq. (6.16), as same as related rank-oriented recommendation studies [23] [5] [58] [59], to optimize the proposed model. BPR loss with positive and negative sampling mechanisms ranks positive cases higher than negative ones. A positive case means an observed user check-in to a POI, and a negative case means unobserved counterparts. Thus, POIs that have more similar embeddings to the target user's embedding will be recommended to the target user.

$$L_{BPR} = -\sum_{u_i=1}^{M} \sum_{p_j \epsilon CN_{u_i}} \sum_{p_k \epsilon P-CN_{u_i}} ln\sigma(\widehat{r_{u_i,p_j}} - \widehat{r_{u_i,p_k}}) + \mu\|\omega\|_2^2 \tag{6.16}$$

where $\widehat{r_{u_i,p_j}}$ indicates the predicted preference score for the positive case (user u_i to POI p_j), $\widehat{r_{u_i,p_k}}$ is the predicted preference score for a negative case (user u_i to POI p_k), $P - CN_{u_i}$ denotes a residual set after removing check-in neighbors CN_{u_i} from the global POI set P, M is the number of users in the dataset. ω denotes all trainable parameters in the model, and μ controls the strength of ℓ_2 regularization to avoid overfitting.

6.3.6 Time Complexity Analysis

The time complexity for training our model and state-of-the-art baselines is analyzed in this section. As lightweight GCN models, the execution time of LightGCN and LR-

GCCF mainly consists of gathering information from neighboring nodes, where the time complexity of LightGCN and LR-GCCF can be analyzed as $3|R^+|D^2$, with $|R^+|$ as the number of non-zero entities in the check-in matrix R and D indicates the embedding size, because the model adopts 3 graph convolution layers and the time complexity for each layer is $|R^+|D^2$. IMP-GCN [2] has an additional subgraph construction module based on the structure of lightweight models (LightGCN and LR-GCCF). Therefore, the time complexity is calculated as $3|R^+|D^2 + 2(M+N)D * D^2 = 3|R^+|D^2 + 2(M+N)D^3$. M and N are the user and POI numbers, respectively.

Table 6-2: Time complexity of proposed EPT-GCN and baselines

Algorithms	Time Complexity for Pre-process	Time Complexity for Model Training
LR-GCCF [23], LightGCN [5]		$3\|R^+\|D^2$
IMP-GCN [2]		$3\|R^+\|D^2 + 2(M+N)D^3$
SGM-GCN (proposed)	$\|T\|\|R^+\|$	$3\|R^+\|D^2 + (M+N)D^3$
EPT-GCN+Geo (proposed)	$\|T\|\|R^+\| + MN$	$4\|R^+\|D^2 + (M+N)D^3$

In SGM-GCN, a pre-process is adopted to partition the edges into multiple time slots, which slightly increases the time complexity by $|T||R^+|$, where $|T|$ indicates the number of time slots. At the model training step, because SGM-GCN adopts an attention mechanism, we calculate the time complexity to generate node embeddings as $3|R^+|D^2 + (M+N)D^3$. EPT-GCN also adds an edge-based propagation module, needing another $|R^+|D^2$, so the overall time complexity for model training is estimated as $3|R^+|D^2 + (M+N)D^3 + |R^+|D^2 = 4|R^+|D^2 + (M+N)D^3$. Using geographical information does not increase the training time complexity. However, the adoption of geographical information leads to unavoidably pre-calculate the distance between users and POIs, introducing the time complexity by MN. Thus, the time complexity for pre-process increases to $|T||R^+| + MN$, which is much smaller than the model training time

complexity. The results of the time complexity analysis are summarized in Table 6-2.

6.3.7 Model Size Analysis

This section analyzes the sizes of our proposed model and other state-of-the-art baselines. For LightGCN and LR-GCCF, the models use $(M+N)D$ trainable parameters, where M and N represent the numbers of users and POIs, respectively, and D indicates the embedding size. The node clustering module in IMP-GCN requires additional transformation weight matrices and bias vectors, which constitute an additional $2D^2 + 2D$ parameters. Therefore, the model size of IMP-GCN is $(M+N)D + 2D^2 + 2D$. Same to IMP-GCN, in our SGM-GCN model, the attention mechanism increases the parameters by $2D^2 + 2D$. The overall model size of SGM-GCN is $(M+N)D + 2D^2 + 2D$. The adoption of geographical information does not cause any increase in the number of trainable parameters. The model size analysis results are summarized in Table 6-3.

Table 6-3: Model sizes of proposed EPT-GCN and baselines

Algorithms	Model Size
LR-GCCF [23], LightGCN [5]	$(M+N)D$
IMP-GCN [2]	$(M+N)D + 2D^2 + 2D$
EPT-GCN+Geo (proposed)	$(M+N)D + 2D^2 + 2D$

6.4 Experimental Evaluation

This section introduces the experimental evaluations and results on two real POI datasets. We compared the three proposed models, SGM-GCN, EPT-GCN, and EPT-GCN+Geo, with baselines.

6.4.1 Datasets

We chose two public datasets consisting of geographical and time information: the Gowalla dataset collected by Liu et al. [36] and the New York dataset collected by Liu et

al. [60]. The Gowalla dataset covers worldwide check-in information, while the New York dataset consists of check-in information in New York City retrieved from Weeplace. Note that different from Chapter 5, we omitted the use of the famous Yelp dataset [36], which has been used in many POI recommendation studies, because the Yelp dataset does not have exact hourly information for each check-in, making it impossible to apply the time slot technique.

The datasets were preprocessed using the same approach as Liu et al. [36] and Section 5.4.1, filtering out users with fewer than 15 check-in POIs and POIs with fewer than 10 user check-ins. After preprocessing, the Gowalla dataset contained 18,737 users, 32,510 POIs, and 1,278,274 check-ins, with a sparsity of 99.87% for the user-item check-in matrix. The New York dataset contained 654,054 check-ins by 3,286 users and 6,369 POIs, with a sparsity of 96.87%. Table 6-4 summarizes the statistics of the datasets. Same as Liu et al.'s work, we set the earliest 70% of check-ins as a training set to predict the latest 20% of check-ins (testing set), and the remaining 10% was used as a tuning set.

Table 6-4: Dataset statistics

Dataset	Number of users	Number of items	Number of check-ins	Sparsity
Gowalla	18,737	32,510	1,278,274	99.87%
New York	3,286	6,369	654,054	96.87%

6.4.2 Baselines

We compared our proposed methods[15] with the six state-of-the-art baselines described below.

1) RankGeoFM [33]: RankGeoFM [33] is based on factorization machines, a state-of-the-art MF-based model. The detail has been described in Section 0, which is not repeated here to avoid redundancy.

[15] https://github.com/bakubonmo/Rec

2) LR-GCCF [23]: LR-GCCF is a linear model used to form a residual network, described in Section 0.

3) LightGCN [5]: LightGCN removes the self-connection to simplify the model, described in Section 0.

4) GPR [3]: GPR integrates geographical information into GCN, where each POI is assigned two trainable embeddings to represent, same as Section 0.

5) GNN-POI [4]: GNN-POI integrates various side information into POI recommendations. A complex nonlinear network structure with consecutive check-ins, social information, and geographical information is used to learn the unique embeddings of nodes (users and POIs). Same as GPR, the time information is reflected in consecutive check-ins. In this experiment, we omitted the social part of GNN-POI to keep the side information used consistently because our model only uses time and geographical information.

6) IMP-GCN [2]: IMP-GCN is a state-of-the-art subgraph-based GCN model for recommendation systems. High-order neighboring users with no common interests are filtered through a node clustering technique composed of three MLP layers. Because our method also adopts the subgraph technique, compared with IMP-GCN, we can verify whether the performance of our time-aware GCN can be improved.

Note that we did not compare with works [61] [62] [63] [65] [64] because of the different research purposes. They studied the next POI recommendation, meaning that the ground truth contained only the target user's last POI in chronological order. Thus, the next POI recommendation algorithm focuses more on mining check-in sequences. In addition, the evaluation metrics also make a significant difference. Influenced by the number of POIs contained in the ground truth of the target user, the next POI recommendation algorithm is evaluated only by recall and cannot calculate precision because the maximum precision is $\frac{1}{|\text{top } k|}$.

In addition, we did not compare with works [66] [67] represented by Zhong et al. [67] due to the use of different side information. They used social information, but we focused on time and geographical information. Different side information may have different degrees of influence on the recommendation performance and is difficult to compare directly. We used Python and PyTorch[16] to implement the GCN baselines while using the source code of Liu et al. [36] for the RankGeoFM model.

6.4.3 Metrics

Same as Section 5.4.3, the experiment adopts three widely adopted ranking metrics: $Precision@k$, $Recall@k$, and $NDCG@k$ for the top$k$ recommendation list, where larger values indicate better performance. The Equations are listed as Eq. (5.9) to Eq. (5.12).

On the Gowalla dataset, the theoretical maximums were P@5=0.949, R@5=0.736, P@10=0.673, and R@10=0.909. On the New York dataset, the theoretical maximums were P@5=0.841, R@5=0.581, P@10=0.702, and R@10=0.744.

6.4.4 Hyperparameter Settings

This section describes the hyperparameter settings for the proposed model, except for the number of time slots, which is discussed in Section 6.4.6.

Hyperparameters for DBSCAN Algorithm

The DBSCAN algorithm was adopted to mine the active areas of users. Same as Section 5.4.4, after searching eps from 0.25 to 2 with the interval 0.25 and $minPts$ from 2 to 5 with the interval 1. We set eps to 1 and $minPts$ to 2 as the optimal parameters.

Hyperparameters for Proposed Models

Same as previous works [3] [23] [5], we fixed the embedding size to 64 and the number

[16] https://pytorch.org/

of hidden layers K to 3 for both the proposed models and all GCN-based baselines. Embeddings were initialized using a Gaussian distribution with mean 0 and standard deviation $1e^{-2}$. The learning rate was searched from $\{1e^{-4}, 1e^{-3}, 1e^{-2}, 1e^{-1}\}$ to tune the best performance, finally set to $1e^{-2}$. The regularization coefficient μ was searched in the range $\{1e^{-6}, 1e^{-5}, \ldots, 1e^{-1}\}$, finally set μ to $1e^{-4}$. γ and δ are two hyperparameters that balance the importance of check-ins and geographical information.

Table 6-5: Summary of hyperparameter settings

Algorithms	Hyperparameter settings		
	Hyperparameter	Search range	Optimal value
DBSCAN	eps	$\{0.25, 0.5, 0.75, \ldots, 2\}$	1 km
	$minPts$	$\{2, 3, 4, 5\}$	2
LR-GCCF [23], LightGCN [5], GPR [3], GNN-POI [4], IMP-GCN [2], ECN-GCN+Geo (proposed)	embedding size	same as Chen et al. [23] and He et al. [5]	64
	embedding initialization	same as Chen et al. [23] and He et al. [5]	Gaussian dist. (Mean:0, SD:$1e^{-2}$)
	learning rate	$\{1e^{-4}, 1e^{-3}, 1e^{-2}, 1e^{-1}\}$	$1e^{-2}$
	regularization coefficient	$\{1e^{-6}, 1e^{-5}, \ldots, 1e^{-1}\}$	$1e^{-4}$
ECN-GCN+Geo (proposed)	edge sampling ratio	$\{1\|\mathcal{E}\|, \frac{1}{25}\|\mathcal{E}\|, \frac{1}{50}\|\mathcal{E}\|, \frac{1}{75}\|\mathcal{E}\|, \frac{1}{100}\|\mathcal{E}\|\}$	$\frac{1}{75}\|\mathcal{E}\|$ on Gowalla $\frac{1}{100}\|\mathcal{E}\|$ on New York
	number of time slots	{2, 3, 4, 6, 8}	4
	check-in coefficient γ	$\{0, 0.25, 0.5, 0.75, 1\}$	0.25

We searched γ and δ from 0 to 1 with the interval 0.25. Finally, we set α to 0.5 and β to 1 as the best parameters. Hyperparameters for the GCN-based baselines were set using the same strategy as for our model. For RankGeoFM, the same parameters as those of Liu et al. [36] were set for a fair comparison. The settings of the hyperparameters are summarized in Table 6-5.

6.4.5 Experimental Results

Table 6-6 and Table 6-7 present our experimental results on the two real datasets. The maximum values other than those of our proposed models are underlined. The maximum value of each metric is bolded.

Table 6-6: Experimental results on New York dataset

Algorithms		$P@5$	$R@5$	$N@5$	$P@10$	$R@10$	$N@10$
Baselines	RankGeoFM [33]	0.0219	0.0244	0.0220	0.0184	0.0260	0.0197
	LR-GCCF [23]	0.0634	0.0248	0.0681	0.0553	0.0410	0.0410
	LightGCN [5]	0.1093	0.0338	0.1214	0.0831	0.0528	0.1055
	GPR [3]	0.1041	0.0307	0.1171	0.0836	0.0507	0.1047
	GNN-POI [4]	<u>0.1114</u>	<u>0.0360</u>	<u>0.1262</u>	0.0864	<u>0.0546</u>	<u>0.1111</u>
	IMP-GCN [2]	0.1100	0.0328	0.1236	<u>0.0865</u>	0.0534	0.1096
Proposed	SGM-GCN	0.1181*	0.0357	0.1333*	0.0921*	0.0567*	0.1171*
	EPT-GCN	0.1233*	0.0382*	0.1414*	0.0935*	0.0597*	0.1224*
	EPT-GCN+Geo	**0.1248***	**0.0388***	**0.1420***	**0.0951***	**0.0601***	**0.1240***
	EPT-GCN+Geo's improvement percentage compared with underlined value (absolute improvement difference from underlined value)	12.03% (0.0134)	7.78% (0.0028)	12.52% (0.0158)	9.94% (0.0086)	10.07% (0.0055)	11.61% (0.0129)

* Statistically significant for $p<0.01$ when comparing with any baselines.

Table 6-7: Experimental results on Gowalla dataset

Algorithms		$P@5$	$R@5$	$N@5$	$P@10$	$R@10$	$N@10$
Baselines	RankGeoFM [33]	0.0684	0.0479	0.0719	0.0559	0.0755	0.0622
	LR-GCCF [23]	0.0640	0.0669	0.0721	0.0504	0.0762	0.0749
	LightGCN [5]	0.0760	0.0788	0.0859	0.0597	0.0865	0.0870
	GPR [3]	<u>0.0775</u>	0.0580	0.0671	<u>0.0640</u>	0.0881	0.0613
	GNN-POI [4]	0.0764	0.0792	0.0860	0.0607	0.0880	0.0877
	IMP-GCN [2]	0.0774	<u>0.0803</u>	<u>0.0874</u>	0.0613	<u>0.0890</u>	<u>0.0889</u>
Proposed	SGM-GCN	0.0803*	0.0834*	0.0914*	0.0632	0.0919*	0.0927*
	EPT-GCN	0.0827*	0.0859*	0.0940*	0.0636	0.0925*	0.0939*
	EPT-GCN+Geo	**0.0843***	**0.0874***	**0.0953***	**0.0648+**	**0.0940***	**0.0953***
	EPT-GCN+Geo's improvement percentage compared with underlined value (Absolute improvement difference from underlined value)	8.77% (0.0068)	8.84% (0.0071)	9.04% (0.0079)	1.25% (0.0008)	5.62% (0.0050)	7.20% (0.0064)

* Statistically significant for $p<0.01$ compared to any baseline.

+ Statistically significant for $p<0.05$ compared to any baseline.

Comparison among Models without Side-Information

Among all the GCN models without time or geographical information (i.e., LR-GCCF, LightGCN, and IMP-GCN), IMP-GCN achieved the best performance on both datasets. This indicates that the subgraph construction technique based on high-order neighbor similarity suits POI recommendation systems to prevent information aggregation from high-order dissimilar nodes.

Comparison between IMP-GCN and GNN-POI

On the New York dataset, GNN-POI outperformed IMP-GCN because of the

integration of time and geographical information. However, on the Gowalla dataset, GNN-POI performed slightly worse than IMP-GCN. A possible reason is that GNN-POI uses complex models and nonlinear representations to train the embeddings of users and POIs, which tends to cause model overrepresentation, that is, complex yet ineffective representation and problems on extremely sparse datasets, such Gowalla with a sparsity of 99.87%, which is higher than that of the New York dataset.

Comparison between GPR (heavy reliance on geographical information) and Other Baselines

The GPR model achieved a higher P@10 compared to the other baselines on the Gowalla dataset. The main reason for this is that the exponential function is adopted for aggregating information over a geographical distance, i.e., closer POIs gather exponentially more information in the GPR model, thereby improving the impact of geographical information. Because the Gowalla dataset is worldwide, GPR works well. The exponential geographic information makes the model recommend nearby POIs and thus filters distant POIs, improving the model performance in a large geographical range dataset. However, in the New York dataset, within the city range, the importance of geographical information decreases, preventing further improvement in model performance.

Comparison between SGM-GCN and IMP-GCN

On both the New York and Gowalla datasets, the SGM-GCN outperformed the IMP-GCN. The IMP-GCN adopts a node-oriented clustering module, whereas the SGM-GCN uses time information to partition edges (check-ins) into multiple time slots. The experimental results show that edge-oriented subgraph partitioning performs better than node-oriented subgraph partitioning in POI-recommendation tasks.

Comparison between EPT-GCN and SGM-GCN

Comparing EPT-GCN with SGM-GCN, we can verify that the proposed propagation module can further improve the model performance based on high-order time slot

embeddings by 7.00% on the New York dataset and 3.00% on the Gowalla dataset in terms of R@5. The results confirm that not only the attention mechanism of SGM-GCN, the propagation module is effective in improving accuracy. Unlike IMP-GCN, EPT-GCN is a check-in-oriented method that enables an edge to be clustered into multiple time slots. For example, the learned characteristics that users prefer to check restaurants during lunchtime can be mapped and propagated to the dinner time slot by the propagation module.

Comparison between EPT-GCN+Geo and Other Baselines

On both datasets, ECN-GCN+Geo exhibited the best performance. ECN-GCN+Geo successfully improved R@5 from 0.0360 to 0.0388 (7.78%) and R@10 from 0.0546 to 0.0601 (10.07%) on the New York dataset while improving R@5 from 0.0803 to 0.0874 (8.84%) and R@10 from 0.0890 to 0.0940 (5.62%) on the Gowalla dataset, compared to state-of-the-art baselines with underlined values. Comparing EPT-GCN+Geo with EPT-GCN, the adoption of geographical information significantly improved the performance on the Gowalla dataset. In contrast, the improvement on the New York dataset was not as high as that of the former. The main reason for this is that the importance of geographical information decreases as the geographical range of the dataset reduces. Note that the Gowalla dataset is worldwide, while the New York dataset is a city-range dataset.

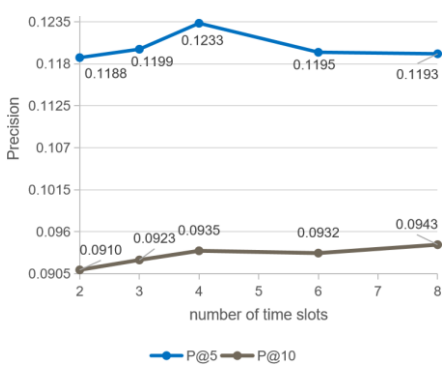
6.4.6 Number of Time Slots

This section investigates the effect of the number of time slots on the proposed EPT-GCN and EPT-GCN+Geo.

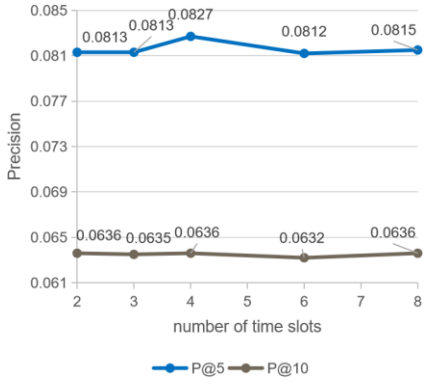
Effect of the number of time slots on EPT-GCN

The number of time slots may affect the performance of our proposed EPT-GCN; therefore, we confirmed the effects by varying the number of time slots from one to eight. For example, if the number of time slots is four, we have the following four time slots:12 AM to 6 AM, 6 AM to 12 PM, 12 PM to 6 PM, and 6 PM to 12 AM local time. For

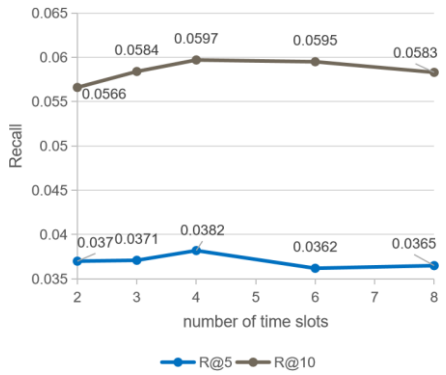
convenience, we start from midnight and equally divide 24 hours into four time slots. We use latitude and longitude coordinates to convert the timestamp to local time.



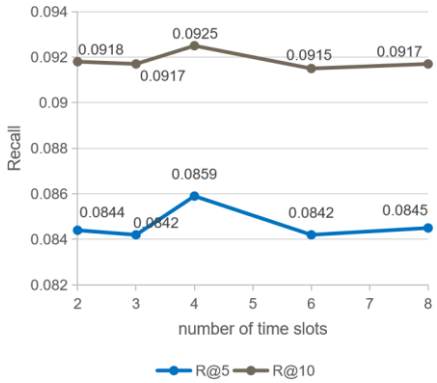
Precision ratio on New York dataset



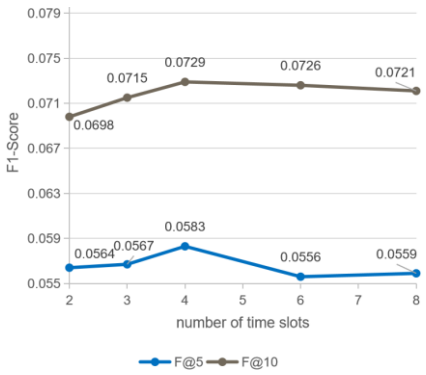
Precision ratio on Gowalla dataset



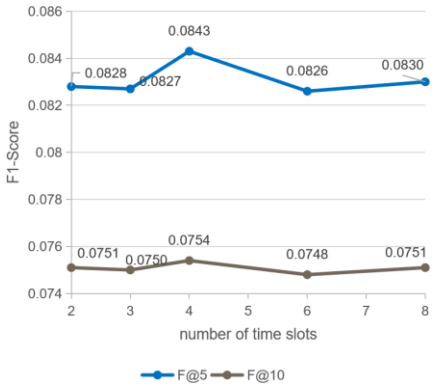
Recall ratio on New York dataset



Recall ratio on Gowalla dataset



F1-Score on New York dataset



F1-Score on Gowalla dataset

Figure 6-6: Influence of the number of time slots on EPT-GCN (edge sampling ratio is set as 1/75 |E| on Gowalla dataset and 1/100 |E| on New York dataset).

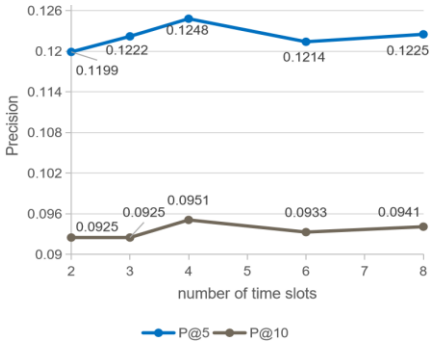
Figure 6-6 shows the precision, recall, and F1-score of EPT-GCN on the two datasets by varying the number of time slots from 2 to 8, where the F1-score is the harmonic mean of precision and recall, calculated as $F1@k = \frac{2*P@k*R@k}{P@k+R@k}$. Similar trends in both datasets were confirmed, i.e., that EPT-GCN performs best when the number of time slots is four, which may conform to user behavior. We also tested the performance when the number of time slots was set to one. The algorithm degenerates to LightGCN, with P@5=0.0752, R@5 =0.0781, P@10=0.0599, and R@10 =0.0871 on the Gowalla dataset and P@5=0.1099, R@5=0.0341, P@10=0.0848, and R@10=0.0531 on the New York dataset.

Effects of Combining Geographical Information on EPT-GCN with Different Number of Time Slots

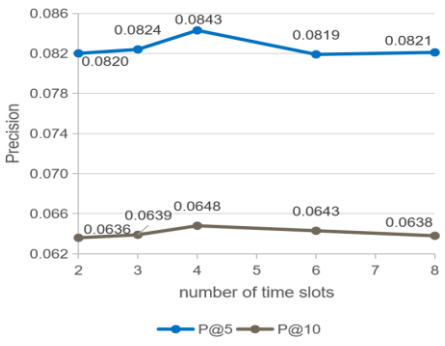
Figure 6-7 illustrates the influence of the number of time slots on EPT-GCN+Geo. Comparing Figure 6-7 with Figure 6-6, we can observe a similar trend in performance with hyperparameter (number of time slots) variation, where the model performs best when the number of time slots is set to four. We also tested the performance when the number of time slots was set to one. The algorithm degenerates to only adopt geographical information and omit time information with P@5=0.0780, R@5=0.0810, P@10=0.0610, and R@10 =0.0881 on the Gowalla dataset and P@5=0.1114, R@5=0.0358, P@10=0.0858, and R@10=0.0529 on the New York dataset.

6.5 Conclusion

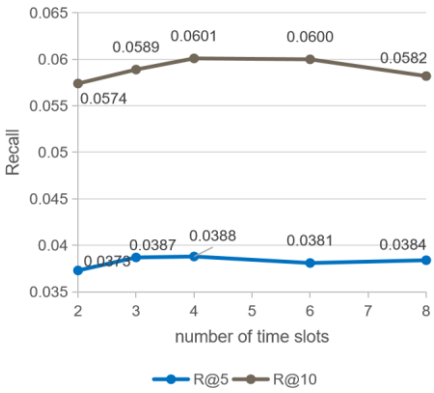
In this chapter, we proposed an edge propagation-based time-aware GCN for POI recommendation constituting the following: 1) a subgraph mining GCN model to divide 24 hours into equal interval time slots and learning users' and POIs' disentangled time-



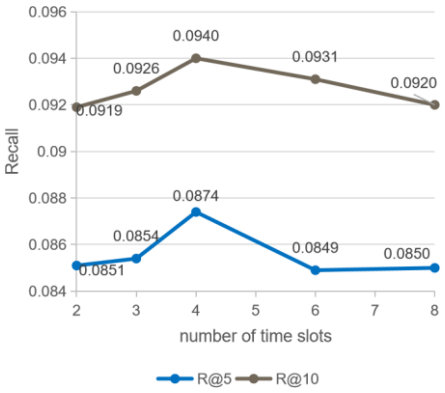
Precision ratio on New York dataset



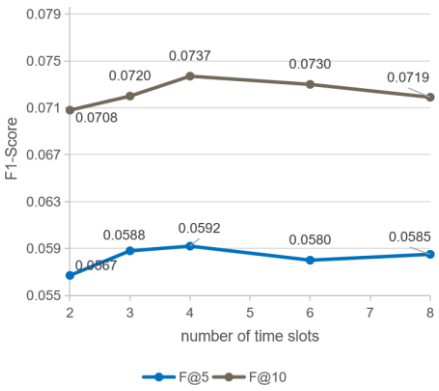
Precision ratio on Gowalla dataset



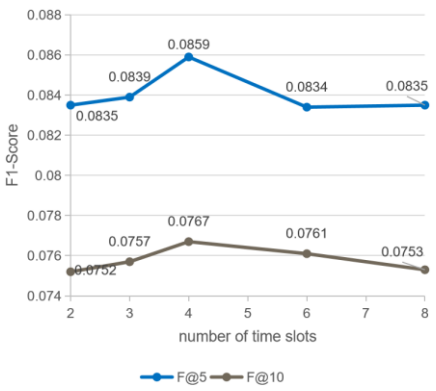
Recall ratio on New York dataset



Recall ratio on Gowalla dataset



F1-Score on New York dataset



F1-Score on Gowalla dataset

Figure 6-7: Influence of the number of time slots on EPT-GCN+Geo (edge sampling ratio is set as 1/75 |E| on Gowalla dataset and 1/100 |E| on New York dataset).

aware embeddings; 2) an edge propagation module to reconstruct subgraphs by calculating the similarity between users' higher-order time slot embeddings and POIs' initial embeddings; and 3) a modified aggregation function to combine check-in information with geographical information. Experimental evaluation on two real datasets (Gowalla and New York) confirms that our proposed method outperforms state-of-the-art baselines. On the Gowalla dataset, Recall @5 improved from 0.0803 to 0.0874 (8.84%), while on the New York dataset, Recall@5 improved from 0.0360 to 0.0388 (7.78%). The proposed subgraph mining technique and novel edge-based propagation module have high scalability and can be applied to other subgraph construction models.

7 Conclusion and Future Work

7.1 Conclusion

In this section, we summarize our work and contributions to combine side information, i.e., time and geographical information, with recommendation models into three-fold.

In Contribution 1, the time information is modeled as training speed for ad recommendation. By using DA to accelerate recommendation optimization to achieve real-time periodic recommendation, the proposed technique can capture changes in user interest over time effectively while satisfying the delivery constraints. Experimental results on the real Geniee dataset confirmed that our proposed method outperforms the baselines by 35.56% with prediction algorithm Logistic regression while shortening the execution time from 525s to 108s and 35.86% with XGBoost while shortening the execution time from 526s to 108s, introduced in Chapter 3.

In Contribution 2, we first introduced some basic knowledge of POI recommendation and previous works related to our contributions, i.e., graph convolution network and side information for POI recommendation (Chapter 4). Then, we divided two Chapters on how to adopt geographical (Chapter 5) and time (Chapter 6) information, named GN-GCN and EPT-GCN, respectively. For the geographical information, we modeled users' multiple active areas. Further, we proposed the concept of active area neighbors, making the GCN model not only aggregate information from check-in but also from active area neighbors. Experiments on real Gowalla and Yelp datasets indicated that the proposed technique successfully improved $Recall@5$ from 0.0788 to 0.0815 on the Gowalla dataset and from 0.0453 to 0.0469 on the Yelp dataset compared with state-of-the-art LightGCN model. The technique and results were introduced in Chapter 5.

In Contribution 3, we modeled users' time-based high-order connectivity for the time information, defined as the relationship between indirect neighbors with similar preferences in the same time slot. For a POI recommendation, combining both time and

geographical information gives better results, which inspired us to modify the aggregation function to further combine the proposed EPT-GCN with geographical information. To verify the performance of the proposed technique, we conducted experiments on real Gowalla and New York datasets, which contain time information and can apply our proposed models. On the Gowalla dataset, Recall @5 improved from 0.0803 to 0.0874. On the New York dataset, Recall@5 improved from 0.0360 to 0.0388, compared with state-of-the-art GCN-based models. We introduced the technique and results in Chapter 6.

7.2 Discussion and Future Work

In this section, we discuss the future research directions on side information.

In the time dimension, we recognize that DA has high speed and can be adapted to various models. We advocate the DA application for other time-sensitive tasks, like the training of deep learning models. The training of deep learning models is slow. Converting the deep learning model into a form of QUBO, which can be executed on DA to increase the speed of training, will be a promising technique. Besides, as a quantum-inspired computer, DA has a finite number of units. How to reduce the problem size and, thus, adapt the DA is also a future research direction.

For side information technique with deep learning model to improve accuracy, the first thing worth mentioning is the integration of more information, such as categorical information, for recommendation accuracy. More information always means that performance can be further improved. Besides, we refocus our attention on geographical and time information. For geographical information, the proposed technique of modeling user active areas does not distinguish between urban and suburban areas. Typically, users behave differently in urban and suburban areas, leading to new thinking and research directions. For time information, a more fine-grained distinction between time-based high-order connectivity may yield a new research direction. For example, dividing the weekends and workdays to make more specific time slots division. Besides, subgraph

reconstruction techniques are also worth exploring. We proposed a technique to partition subgraphs based on edges. Subgraph partitioning that considers both edges and nodes to improve the time-based high-order mining may achieve better performance.

7.3 Discussion of Recommendation Beyond Accuracy

The above discussion is for the accuracy of recommendation systems. In addition, we would like to discuss other research goals of recommendation systems from a larger perspective.

In addition to recommendation accuracy, the diversity of recommendation systems has received increasing attention from researchers in recent years. Recommendation diversity is primarily defined as categorical diversity [70] [71]. Let us consider the restaurant recommendation. Categorical diversity recommends a wide variety of restaurant categories, such as Chinese, Japanese, and French restaurants, that may interest the target user, not just Chinese restaurants, even if the target user likes Chinese food. Recommendation system diversity keeps the results fresh in users' minds, thus increasing users' satisfaction. In the research area, improving the accuracy and beyond-accuracy aspects-diversity are conflicting tasks, which complicate the diversity improvement and accuracy maintenance tasks. Previous research [85] [86] attempted to adopt a re-ranking technique to alleviate the conflicts. i.e., first, use a base algorithm to generate a candidate recommendation list, followed by an optimization step for improving diversity. However, optimizing the diversity by reranking is independent of the basic candidate-item generation model, resulting in a suboptimal system. In recent years, designing the model of de-reranking [87] has still been in the exploratory stage. A worthwhile research direction is how to effectively represent users' needs for diversity in machine learning models.

Another research goal that is still in the exploratory stage is recommendation proportionality. Proportionality in recommendation results [72] is a further requirement beyond accuracy. More than diversity, recommendation proportionality ensures that the

past interests of the target user are proportionally reflected in a recommendation list. In several papers [73] [74], the technical term “calibration” is used to replace “proportionality.” Let us consider the previous example of the restaurant to explain the recommendation calibration (proportionality). Assume a target user checks 80, 10, and 10% Chinese, Japanese, and French restaurants, respectively; the target user may be highly interested in receiving a recommendation list with the same distribution. Synthesizing the multiple interests of users will be a future research trend. Similar to recommendation diversity, reranking techniques are widely used in recommendation calibration [88] [89]. As a result, the future research direction is similar to that of diversity, which tends to remove reranking and directly represent the users' needs for calibration in the machine learning models.

Recommendation explainability [75] [76] is also emphasized by researchers. Recommendation explainability is required when the list of recommendations is generated; the reasons for the recommendation items are also generated to make the results more acceptable to the target user. A simple method to provide explainability is using keywords [75]. For example, recommending the item i because a similar user “purchased” the item. Another example is because the item “is described by” feature f. However, the keyword-based technique requires pre-designed templates for the explanation, making the explanation unnatural. In recent years, with the development of natural language processing (NLP) techniques like Transformer [82], combining recommendation systems and NLP techniques has made it possible to create recommendation reasons automatically and naturally, and it will be a promising research direction.

Recommendation fairness [90] [91] is described as fair exposure to different items. i.e., no popularity bias among different items. The research aims to mitigate the long-tail effect and recommends unpopular items. While simultaneous optimization of accuracy and fairness in deep learning models [84] was proposed to alleviate the conflicting two goals (high fairness and high accuracy), recommending unpopular items still carries the risk of

reduced accuracy. How to improve fairness with minimal loss of accuracy will be a focus of research in the area.

Recommendation acceleration [77] has also injected new challenges for recommendation systems. Current computing resources are expensive, including GPUs and high electricity costs. Slow training speed means high overhead. Therefore, working on improving the training speed of models is also an essential task in the field of recommendation systems. In addition to the possibility of using DA to accelerate model training, as mentioned in Chapter 7.2, techniques to accelerate model convergence [92] were proposed to reduce training time. However, the methods do not reduce the time complexity of the models, which leads to a less generalized approach. A generalized algorithm for reducing time complexity will be a promising research direction.

Finally, we introduce the recommendation systems in incremental environments [78] [83]. Retaining the model is time-consuming. Therefore, we tend to update the already-trained model after new data arrives instead of retraining it. In an incremental environment, selective retention of learned knowledge is challenging. Prior studies [78] [83] introduced a sampling-based technique to sample the learned knowledge but couldn't achieve the same performance as retaining. i.e., reduce performance. An approach suitable for model training in an incremental environment is worth considering, such as a mechanism to selectively update only part of the model's parameters.

Reference

[1] Aramon M., Rosenberg G., Valiante E., Miyazawa T., Tamura H., and Katzgrabeer H. 2019. Physics-inspired optimization for quadratic unconstrained problems using a digital annealer. Frontiers in Physics, 7(48), pp.1-14.

[2] Liu, F., Cheng, Z., Zhu, L., Gao, Z., and Nie, L. 2021. Interest-aware message-passing GCN for recommendation. In Proceedings of the Web Conference 2021, pp. 1296-1305.

[3] Chang, B., Jang, G., Kim, S., and Kang, J. 2020. Learning graph-based geographical latent representation for point-of-interest recommendation. In Proceedings of the 29th ACM International Conference on Information & Knowledge Management, pp. 135-144.

[4] Zhang, J., Liu, X., Zhou, X., and Chu, X. 2021. Leveraging graph neural networks for point-of-interest recommendations. Neurocomputing, 462, 1-13.

[5] He, X., Deng, K., Wang, X., Li, Y., Zhang, Y., and Wang, M. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In Proceedings of the 43rd ACM SIGIR, pp. 639-648.

[6] Abrams, Z., Mendelevitch, O., and Tomlin, J. 2007. Optimal delivery of sponsored search advertisements subject to budget constraints. In Proceedings of the 8th ACM conference on Electronic commerce, pp. 272-278.

[7] Wu, D., Chen, X., Yang, X., Wang, H., Tan, Q., Zhang, X., and Gai, K. 2018. Budget constrained bidding by model-free reinforcement learning in display advertising. In Proceedings of the 27th ACM International Conference on Information and Knowledge Management, pp. 1443-1451.

[8] Yang, X., Deng, T., Tan, W., Tao, X., Zhang, J., Qin, S., and Ding, Z. 2019. Learning Compositional, Visual and Relational Representations for CTR Prediction in Sponsored Search. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management, pp. 2851-2859.

[9] Aramon, M., Rosenberg, G., Valiante, E., Miyazawa, T., Tamura, H., and Katzgrabeer, H., 2019. Physics-inspired optimization for quadratic unconstrained problems using a digital annealer. Frontiers in Physics, 7(48), pp.1-14.

[10] Mo, F., Jiao, H., Morisawa, S., Nakamura, M., Kimura, K., Fujisawa, H., Ohtsuka, M., and Yamana, H. 2020. Real-Time Periodic Advertisement

Recommendation Optimization using Ising Machine. In Proceedings of 2020 IEEE International Conference on Big Data, pp.5783-5785.

[11] Shan, L., Lin, L., and Sun, C. 2018. Combined Regression and Tripletwise Learning for Conversion Rate Prediction in Real-Time Bidding Advertising. In Proceedings of the 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, pp. 115-123.

[12] Su, Y., Jin, Z., Chen, Y., Sun, X., Yang, Y., Qiao, F., and Xu, W. 2017. Improving click-through rate prediction accuracy in online advertising by transfer learning. In Proceedings of the International Conference on Web Intelligence, pp. 1018-1025.

[13] Agarwal, D., Chen, B., and Elango, P. 2009. Spatio-temporal models for estimating click-through rate. In Proceedings of the 18th international conference on World wide web, pp. 21-30.

[14] Juan, Y., Lefortier, D., and Chapelle, O. 2017. Field-aware factorization machines in a real-world online advertising system. In Proceedings of the 26th International Conference on World Wide Web Conference, pp. 680-688.

[15] Pan, J., Xu, J., Ruiz, A., Zhao, W., Pan, S., Sun, Y., and Lu, Q. 2018. Field-weighted factorization machines for click-through rate prediction in display advertising. In Proceedings of the 2018 World Wide Web Conference, pp. 1349-1357.

[16] Wang, R., Fu, B., Fu, G., and Wang, M. 2017. Deep & cross network for ad click predictions. In Proceedings of the ADKDD'17, pp. 1-7.

[17] Yang, X., Li, Y., Wang, H., Wu, D., Tan, Q., Xu, J., and Gai, K. 2019. Bid optimization by multivariable control in display advertising. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 1966-1974.

[18] Huang, Z., Pan, Z., Liu, Q., Long, B., Ma, H., and Chen, E. 2017. An Ad CTR Prediction Method Based on Feature Learning of Deep and Shallow Layers. In Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, pp. 2119-2122.

[19] Kang, S., Jeong, C., and Chung, K. 2020. Advertisement Recommendation System Based on User Preference in Online Broadcasting. In Proceedings of 2020 International Conference on Information Networking, pp. 702-706.

[20] Grigas, P., Lobos, A., Wen, Z., and Lee, K. C. 2017. Profit maximization for online advertising demand-side platforms. In Proceedings of the ADKDD'17, pp. 1-7.

[21] Chen, T., and Guestrin, C. 2016. XGBoost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 785-794.

[22] Goldberg, D. E. 1989. Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Longman Publishing Company.

[23] Chen, L., Wu, L., Hong, R., Zhang, K., and Wang, M. 2020. Revisiting graph based collaborative filtering: A linear residual graph convolutional network approach. In: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 27-34.

[24] Fan, W., Ma, Y., Li, Q., He, Y., Zhao, E., Tang, J., and Yin, D. 2019. Graph neural networks for social recommendation. In Proceedings of the WWW, pp. 417-426 .

[25] Wang, X., He, X., Wang, M., Feng, F., and Chua, T. 2019. Neural graph collaborative filtering. In Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval, pp. 165-174 .

[26] Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W., and Leskovec, J. 2018. Graph convolutional neural networks for web-scale recommender systems. In Proceedings of the 24th ACM SIGKDD, pp. 974-983.

[27] Baral, R., and Li, T. 2016. Maps: A multi aspect personalized poi recommender system. In Proceedings of the 10th ACM RecSys, pp. 281-284.

[28] Liu, W., Wang, Z. J., Yao, B., and Yin, J. 2019. Geo-ALM: POI Recommendation by Fusing Geographical Information and Adversarial Learning Mechanism. In Proceedings of the 28th International Joint Conference on Artificial Intelligence, pp. 1807-1813.

[29] Zhang, and Chow, C. 2015. Geosoca: Exploiting geographical, social and categorical correlations for point-of-interest recommendations. In Proceedings of the 38th SIGIR, pp. 443-452.

[30] Ye, M., Yin, P., Lee, W. C., and Lee, D. 2011. Exploiting geographical influence for collaborative point-of-interest recommendation. In Proceedings of the 34th ACM SIGIR, pp. 325-334.

[31] Ference, G., Ye, M., and Lee, W. C. 2013. Location recommendation for out-of-town users in location-based social networks. In Proceedings of the 22nd ACM CIKM, pp. 721-726.

[32] Han, P., Shang, S., Sun, A., Zhao, P., Zheng, K., and Zhang, X. 2021. Point-of-interest recommendation with global and local context. IEEE Transactions on Knowledge and Data Engineering, 34(11), pp. 5484-5495.

[33] Li, X., Cong, G., Li, X., Pham, T., and Krishnaswamy. 2015. Rank-geofm: A ranking based geographical factorization method for point of interest recommendation. In Proceedings of the 38th ACM SIGIR, pp. 433-442.

[34] Ester, M., Kriegel, H. P., Sander, J., and Xu, X. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In Proceedings of the 2nd ACM SIGKDD, pp. 226-231.

[35] Rendle, S., Freudenthaler, C., Gantner, Z., and Schmidt-Thieme, L. 2012. BPR: Bayesian personalized ranking from implicit feedback. In arXiv preprint arXiv:1205.2618, 10 pages.

[36] Liu, Y., Pham, T. A. N., Cong, G., and Yuan, Q. 2017. An experimental evaluation of point-of-interest recommendation in location-based social networks. In Proceedings of the VLDB Endowment, vol.10, no.10, pp. 1010-1021.

[37] Elmi, S., Benouaret, K., and Tan, K. L. 2021. Social and Spatio-Temporal Learning for Contextualized Next Points-of-Interest Prediction. In Proceedings of 2021 IEEE 33rd International Conference on Tools with Artificial Intelligence, pp.322-329.

[38] Cui, Q., Tang, Y., Wu, S., and Wang, L. 2019. Distance2Pre: Personalized spatial preference for next point-of-interest prediction. In Proceedings of Pacific-Asia Conference on Knowledge Discovery and Data Mining pp. 289-301.

[39] Yuan, Q., Cong, G., Ma, Z., Sun, A., and Thalmann, N. 2013. Time-aware point-of-interest recommendation. In Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval, pp. 363-372.

[40] Gao, H., Tang, J., Hu, X., and Liu, H. 2013. Exploring temporal effects for location recommendation on location-based social networks. In Proceedings of the 7th ACM conference on Recommender systems, pp. 93-100.

[41] Ying, Y., Chen, L., and Chen, G. 2017. A temporal-aware POI recommendation system using context-aware tensor decomposition and weighted HITS. Neurocomputing, 242, pp. 195-205.

[42] Zhao, S., Zhao, T., King, I., and Lyu, M. R. 2017. Geo-teaser: Geo-temporal sequential embedding rank for point-of-interest recommendation. In Proceedings of the 26th international conference on world wide web companion, pp. 153-162.

[43] Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. Advances in neural information processing systems, 26, pp. 1-9.

[44] Chen, C., Ma, W., Zhang, M., Wang, Z., He, X., Wang, C., Liu, Y, and Ma, S. 2021. Graph heterogeneous multi-relational recommendation. In Proceedings of the AAAI Conference on Artificial Intelligence, pp. 3958-3966.

[45] Shi, C., Hu, B., Zhao, W. X., and Philip, S. Y. 2018. Heterogeneous Information Network Embedding for Recommendation. IEEE Transactions on Knowledge and Data Engineering, 31(2), pp. 357-370.

[46] Yang, Y., Guan, Z., Li, J., Zhao, W., Cui, J., and Wang, Q. 2023. Interpretable and Efficient Heterogeneous Graph Convolutional Network. IEEE Transactions on Knowledge and Data Engineering, 35(2), pp. 1637-1650.

[47] Mao, K., Zhu, J., Xiao, X., Lu, B., Wang, Z., and He, X. 2021. UltraGCN: ultra simplification of graph convolutional networks for recommendation. In Proceedings of the 30th ACM International Conference on Information & Knowledge Management, pp. 1253-1262.

[48] Wu, J., Wang, X., Feng, F., He, X., Chen, L., Lian, J., and Xie, X. 2021. Self-supervised graph learning for recommendation. In Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval, pp. 726-735.

[49] Yang, Z., Ding, M., Xu, B., Yang, H., and Tang, J. 2022. STAM: A Spatiotemporal Aggregation Method for Graph Neural Network-based Recommendation. In Proceedings of the ACM Web Conference 2022, pp. 3217-3228.

[50] Zhang, Y., Wang, P., Zhao, X., Qi, H., He, J., Jin, J., Lin, Z. and Shao, J. 2022. IA-GCN: Interactive Graph Convolutional Network for Recommendation. arXiv preprint arXiv:2204.03827, 11 pages.

[51] Liu, F., Cheng, Z., Zhu, L., Gao, Z., and Nie, L. 2021. Interest-aware message-passing GCN for recommendation. In Proceedings of the Web Conference 2021, pp. 1296-1305.

[52] Liu, G., Wang, J., and Wu, J. 2021. Multi-Aspect Heterogeneous Graph Convolutional Network for Recommendation. In Proceedings of 2021 IEEE 33rd International Conference on Tools with Artificial Intelligence, pp. 1192-1196.

[53] Peng, S., Sugiyama, K., and Mine, T. 2022. Less is More: Reweighting Important Spectral Graph Features for Recommendation. In Proceedings of the 45th International ACM SIGIR conference on research and development in Information Retrieval, pp. 1273-1282.

[54] Sun, T., Luo, M., Chen, R., Xia, Y., and Jiang, N. 2021. Rec-clusterGCN: An Efficient Graph Convolution Network for Recommendation. In Proceedings of 2021 IEEE International Conference on Systems, Man, and Cybernetics, pp. 244-250.

[55] Mo, F., Jiao, H., and Yamana, H. 2020. Time distribution based diversified point of interest recommendation. In Proceedings of 2020 IEEE 5th International Conference on Cloud Computing and Big Data Analytics, pp. 37-44.

[56] Jiang, Y., Ma, H., Liu, Y., Li, Z., and Chang, L. 2021. Enhancing social recommendation via two-level graph attentional networks. Neurocomputing, 449, pp. 71-84.

[57] Mo, F. and Yamana, H. 2022. GN-GCN: Combining Geographical Neighbor Concept with Graph Convolution Network for POI Recommendation. In Proceedings of the 24th International Conference on Information Integration and Web Intelligence, pp. 153-165.

[58] Wu B., Zhong L., Yao L., and Ye Y. 2022. EAGCN: An Efficient Adaptive Graph Convolutional Network for Item Recommendation in Social Internet of Things. in IEEE Internet of Things Journal, vol. 9, no. 17, pp. 16386-16401.

[59] Yi, Z., Wang, X., Ounis, I., and Macdonald, C. 2022. Multi-modal Graph Contrastive Learning for Micro-video Recommendation. In Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 1807-1811.

[60] Liu, X., Liu, Y., Aberer, K., and Miao, C. 2013. Personalized point-of-interest recommendation by mining users' preference transition. In Proceedings of the 22nd ACM international conference on Information & Knowledge Management, pp. 733-738.

[61] Elmi, S., Benouaret, K., and Tan, K. L. 2021. Social and Spatio-Temporal Learning for Contextualized Next Points-of-Interest Prediction. In Proceedings of 2021 IEEE 33rd International Conference on Tools with Artificial Intelligence, pp.322-329.

[62] Li, M., Zheng, W., Xiao, Y., Zhu, K., and Huang, W. 2021. Exploring Temporal and Spatial Features for Next POI Recommendation in LBSNs. IEEE Access, vol. 9, pp. 35997-36007.

[63] Li, Y., Chen, T., Yin, H., and Huang, Z. 2021. Discovering collaborative signals for next POI recommendation with iterative Seq2Graph augmentation. arXiv preprint arXiv:2106.15814, 7 pages.

[64] Wang, Z., Zhu, Y., Liu, H., and Wang, C. 2022. Learning Graph-based Disentangled Representations for Next POI Recommendation. In Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 1154-1163.

[65] Luo, Y., Liu, Q., and Liu, Z. 2021. Stan: Spatio-temporal attention network for next location recommendation. In Proceedings of the Web Conference 2021, pp. 2177-2185.

[66] Cai, Z., Yuan, G., Qiao, S., Qu, S., Zhang, Y., and Bing, R. 2022. FG-CF: Friends-aware graph collaborative filtering for POI recommendation. Neurocomputing, 488, pp. 107-119.

[67] Zhong, T., Zhang, S., Zhou, F., Zhang, K., Trajcevski, G., and Wu, J. 2020. Hybrid graph convolutional networks with multi-head attention for location recommendation. In World Wide Web, pp. 3125-3151.

[68] Mo, F., Jiao, H., Morisawa, S., Ohtsuka, M., Nakamura, M., Kimura, K., Fujssawa, H., and Yamana, H. 2021. Real-time Periodic Advertisement Recommendation Optimization under Delivery Constraint using Quantum-inspired Computer. In Proceedings of 2021 International Conference on Enterprise Information Systems, pp. 431-441.

[69] Mo, F., and Yamana, H. (2023). EPT-GCN: Edge propagation-based time-aware graph convolution network for POI recommendation. Neurocomputing, 543, 126272, pp. 1-15.

[70] Ye, R., Hou, Y., Lei, T., Zhang, Y., Zhang, Q., Guo, J., Wu, H., Zhang, Q., and Luo, H. 2021. Dynamic graph construction for improving diversity of recommendation. In Proceedings of the 15th ACM Conference on Recommender Systems, pp. 651-655.

[71] Zheng, Y., Gao, C., Chen, L., Jin, D., and Li, Y. 2021. Dgcn: Diversified recommendation with graph convolutional networks. In Proceedings of the Web Conference 2021, pp. 401-412.

[72] Dang, V., and Croft, W. B. 2012. Diversity by proportionality: an election-based approach to search result diversification. In Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval, pp. 65-74.

[73] Steck, H. 2018. Calibrated recommendations. In Proceedings of the 12th ACM conference on recommender systems, pp. 154-162.

[74] Liu, F., Cheng, Z., Zhu, L., Gao, Z., and Nie, L. 2021. Interest-aware message-passing GCN for recommendation. In Proceedings of the Web Conference 2021, pp. 1296-1305.

[75] Xian, Y., Fu, Z., Muthukrishnan, S., De Melo, G., and Zhang, Y. 2019. Reinforcement knowledge graph reasoning for explainable recommendation. In Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval, pp. 285-294.

[76] Wang, X., He, X., Cao, Y., Liu, M., and Chua, T. S. 2019. Kgat: Knowledge graph attention network for recommendation. In Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining, pp. 950-958.

[77] Ke, L., Gupta, U., Cho, B. Y., Brooks, D., Chandra, V., Diril, U., ... and Zhang, X. 2020. Recnmp: Accelerating personalized recommendation with near-memory processing. In Proceedings of 2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture, pp. 790-803.

[78] Wu, Guile, Shaogang Gong, and Pan Li. 2021. Striking a balance between stability and plasticity for class-incremental learning. In Proceedings of the IEEE/CVF Int'l Conf. on Computer Vision, pp. 1124-1133.

[79] Li, Z., Chen, Q., and Koltun, V. 2018. Combinatorial optimization with graph convolutional networks and guided tree search. Advances in neural information processing systems, 31.

[80] Khalil, E., Dai, H., Zhang, Y., Dilkina, B., and Song, L. 2017. Learning combinatorial optimization algorithms over graphs. Advances in neural information processing systems, 30.

[81] DasGupta, B., and Muthukrishnan, S. 2013. Stochastic budget optimization in internet advertising. Algorithmica, 65, 634-661.

[82] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... and Polosukhin, I. 2017. Attention is all you need. Advances in neural information processing systems, 30, pp. 1-11.

[83] Fan, X., Mo, F., Chen, C., Bai, C., Yamana, H., Connectivity-aware Experience Replay for Graph Convolution Network-based Collaborative Filtering in Incremental Setting. In Proceedings of 2024 IEEE 9th International Conference on Big Data Analytics. (Forthcoming)

[84] Dai, E., and Wang, S. 2021. Say no to the discrimination: Learning fair graph neural networks with limited sensitive attribute information. In Proceedings of the 14th ACM International Conference on Web Search and Data Mining, pp. 680-688.

[85] Qin, L., and Zhu, X. 2013. Promoting diversity in recommendation by entropy regularizer. In Twenty-Third International Joint Conference on Artificial Intelligence, pp. 2698-2704.

[86] Sha, C., Wu, X., and Niu, J. 2016. A framework for recommending relevant and diverse items. In IJCAI'16, pp. 3868-3874.

[87] Zheng, Y., Gao, C., Chen, L., Jin, D., and Li, Y. 2021. Dgcn: Diversified recommendation with graph convolutional networks. In Proceedings of the Web Conference 2021, pp. 401-412.

[88] Steck, H. 2018. Calibrated recommendations. In Proceedings of the 12th ACM conference on recommender systems, pp. 154-162.

[89] Seymen, S., Abdollahpouri, H., and Malthouse, E. C. 2021. A constrained optimization approach for calibrated recommendations. In Proceedings of the 15th ACM Conference on Recommender Systems, pp. 607-612.

[90] Li, Y., Ge, Y., and Zhang, Y. 2021. Tutorial on fairness of machine learning in recommender systems. In Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval, pp. 2654-2657.

[91] Mehrotra, R., McInerney, J., Bouchard, H., Lalmas, M., and Diaz, F. 2018. Towards a fair marketplace: Counterfactual evaluation of the trade-off between relevance, fairness & satisfaction in recommendation systems. In Proceedings of the 27th acm international conference on information and knowledge management, pp. 2243-2251.

[92] Wang, Y., Zhao, Y., Zhang, Y., and Derr, T. 2023. Collaboration-Aware Graph Convolutional Network for Recommender Systems. In Proceedings of the ACM Web Conference 2023, pp. 91-101.

[93] Ma, Q., Ge, S., He, D., Thaker, D., and Drori, I. 2019. Combinatorial optimization by graph pointer networks and hierarchical reinforcement learning. arXiv preprint arXiv:1911.04936, 8pages.

[94] Brasó, G., and Leal-Taixé, L. 2020. Learning a neural solver for multiple object tracking. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 6247-6257.

[95] Resnick, P., and Varian, H. R. 1997. Recommender systems. Communications of the ACM, 40(3), pp. 56-58.

[96] Liu, J., Chen, Y., Huang, X., Li, J., and Min, G. 2023. GNN-based long and short term preference modeling for next-location prediction. Information Sciences, 629, pp.1-14.

List of My Publications (Including co-authors)

List of Research Achievements

Journal	(1) (Web of science) (Scopus) (Peer-reviewed) Mo, F., and Yamana, H. (2023). EPT-GCN: Edge propagation-based time-aware graph convolution network for POI recommendation. *Neurocomputing*, *543*, 126272, pp.1-15.
International conference papers	(1) (Scopus) (Full paper) (Peer-reviewed) Feng, J., Mo, F., Yada, Y., Matsumoto, T., Fukushima, N., Kido, F., and Yamana, H. (2023). Analysis of Dark Pattern-related Tweets from 2010. In 2023 IEEE 8th International Conference on Big Data Analytics (ICBDA), pp. 100-106. (2) (Scopus) (Full paper) (Peer-reviewed) Chen, C., Mo, F., Fan, X., Bai, C., and Yamana, H. (2023). MOBARec-GCNFP: Champion Recommendation for Multi-Player Online Battle Arena Games Using Graph Convolution Network with Fewer Parameters. In Proceedings of 2023 IEEE 8th International Conference on Big Data Analytics (ICBDA), pp. 147-153. (3) (Web of science) (Scopus) (Full paper) (Peer-reviewed) Mo, F., and Yamana, H. (2022). GN-GCN: Combining Geographical Neighbor Concept with Graph Convolution Network for POI Recommendation. In Proceedings of Information Integration and Web Intelligence: 24th International Conference, iiWAS 2022, pp. 153-165.

	(4) (Web of science) (Scopus) (Full paper) (Peer-reviewed) Mo, F., Jiao, H., Morisawa, S., Ohtsuka, M., Nakamura, M., Kimura, K., Fujssawa, H., and Yamana, H. (2021). Real-time Periodic Advertisement Recommendation Optimization under Delivery Constraint using Quantum-inspired Computer. in Proceedings of 2021 International Conference on Enterprise Information Systems (ICEIS 2021), pp. 431-441. (5) (Scopus) (Full paper) (Peer-reviewed) Jiao, H., Mo, F., and Yamana, H. (2021). Point of Interest Recommendation Acceleration using Clustering. in Proceedings of 2021 IEEE International Conference on Big Data Analystics (ICBDA 2021), pp. 175-180. (6) (Scopus) (Full paper) (Peer-reviewed) Mo, F., Jiao, H., and Yamana, H. (2020). Time Distribution based Diversified Point of Interest Recommendation. in Proceedings of IEEE 5th International Conference on Cloud Computing and Big Data Analytics (ICCCBDA 2020), pp. 37-44.
Workshop and Poster papers	(1) (Scopus) (Workshop) (Peer-reviewed) Mo, F., Matsumoto, T., Fukushima, N., Kido, F., and Yamana, H. (2022). Decoy Effect of Recommendation Systems on Real E-commerce Websites. in Proceedings of The Joint Workshop on Interfaces and Human Decision Making for Recommender Systems (IntRS@Recsys 2022), pp.151-163. (2) (Scopus) (Poster) (Peer-reviewed) Mo, F., Jiao, H., Morisawa, S., Ohtsuka, M., Nakamura, M., Kimura, K., Fujssawa, H., and Yamana, H. (2020) Real-Time Periodic

	Advertisement Recommendation Optimization using Ising Machine. in Proceedings of 2020 IEEE International Conference on Big Data (BigData 2020), pp. 5783-5785. (3) (Scopus) (Workshop) (Peer-reviewed) Mo, F., and Yamana, H. (2019) Point of Interest Recommendation by Exploiting Geographical Weigh Center and Categorical Preference. in Proceedings of 2019 International Conference on Data Mining Workshops (ICDMW 2019), pp. 73-76.
Domestic conference papers (non-referred)	(1) (Non-reviewed) Ma, S., Mo, F., and Yamana, H. (2021) Review-aware Explainable Recommendation System with Aspect Matching. in Proceedings of 19th Forum on Data Engineering and Information Management (DEIM 2021), pp. 1-6. (2) (Non-reviewed) Jiao, H., and Mo, F., and Yamana, H. (2020) Evaluation of POI Recommendation System Beyond Accuracy: Diversity, Explainability and Computation Cost. in Proceedings of 18th Forum on Data Engineering and Information Management (DEIM 2020), March, pp. 1-5.
Awards	(1) Excellent Presentation Award on 2020 IEEE 5th International Conference on Cloud Computing and Big Data Analytics (ICCCBDA 2020).
International Conference	(1) Session Chair of 2021 International Conference on Enterprise Information Systems. Session name: Industrial

Related Activities	Applications of Artificial Intelligence (2) Session Chair of 2021 International Conference on Enterprise Information Systems. Session name: Enterprise Architecture (3) Review of Paper of the 6th international conference on Computer science and Application Engineering (CSAE 2022).