Collaborative filtering by graph convolution network in location-based recommendation system

Tin T. Tran[1*], Vaclav Snasel[2], and Thuan Q. Nguyen[3]
[1] Artificial Intelligence Laboratory, Faculty of Information Technology,
Ton Duc Thang University, Ho Chi Minh city, Vietnam
[e-mail: trantrungtin@tdtu.edu.vn]
[2] VSB-Technical University of Ostrava, 708 00 Ostrava, Czech Republic
[e-mail: vaclav.snasel@vsb.cz]
[3] Ho Chi Minh City Open University, Ho Chi Minh city, Vietnam
[e-mail: thuan.nq@ou.edu.vn]
[*]Corresponding author: Tin T. Tran

Received February 18, 2024; revised May 22, 2024; accepted June 12, 2024; published July 31, 2024

Abstract

Recommendation systems research is a subfield of information retrieval, as these systems recommend appropriate items to users during their visits. Appropriate recommendation results will help users save time searching while increasing productivity at work, travel, or shopping. The problem becomes more difficult when the items are geographical locations on the ground, as they are associated with a wealth of contextual information, such as geographical location, opening time, and sequence of related locations. Furthermore, on social networking platforms that allow users to check in or express interest when visiting a specific location, their friends receive this signal by spreading the word on that online social network. Consideration should be given to relationship data extracted from online social networking platforms, as well as their impact on the geolocation recommendation process. In this study, we compare the similarity of geographic locations based on their distance on the ground and their correlation with users who have checked in at those locations. When calculating feature embeddings for users and locations, social relationships are also considered as attention signals. The similarity value between location and correlation between users will be exploited in the overall architecture of the recommendation model, which will employ graph convolution networks to generate recommendations with high precision and recall. The proposed model is implemented and executed on popular datasets, then compared to baseline models to assess its overall effectiveness.

Keywords: Location-based recommendation, point of interest, social recommender system, collaborative filtering, graph convolution network.

1. Introduction

Recommendation systems suggest items to users within that systems. Items can include products on an online shopping platform, movies in a theater, books in a library, or tourist attractions. Recommendation systems must look for similarities between users as well as among items to archive good results. Based on the results, the system will select a set of items with high similarity scores and recommend them to the intended users. In a world of explosive information nowadays, both the number of users and the number of items grow rapidly. Classical recommendation models that use matrix factorization (MF) [1] and singular value decomposition (SVD) techniques [2] are quickly overloaded because the size of the user and product interaction matrix grows rapidly, and the value is constantly updated over time. Furthermore, the system's calculation speed must be fast enough to make timely recommendations. These factors make it difficult for information mining models to process data from a variety of sources quickly.

Graph neural network-based recommendation models calculate feature embeddings for both users and items, as well as the level of interaction between a specific user and a candidate item. The collaborative filtering mechanism is used during the propagation process, which has the outstanding advantages of directly recognizing similarities between users when they share many interacted items, as well as indirectly when the process of iterative propagation creates a chain of interactions that can be extended from user to item to user. The size of the embeddings is chosen so that data loss after the learning process stays within an acceptable threshold. Not only is the interaction between users and items investigated, but contextual information also influences the overall recommendation results. The first significant source of information is the user's real-life relationships or friend information obtained from online social networking platforms. According to [3, 4, 5], advice from friends gave more weight than other sources of information, so friendship data must be incorporated into the overall data retrieval process. Items will be correlated if they belong to the same catalogue or a closely related product group. When items are geographical locations within a city or area, their actual distance will reflect the correlation observed during a visit by a specific user. When a user is staying in a specific location, nearby locations are obviously more likely to be visited than farther locations.

In this article, we propose a model for transforming input data into interaction information matrices for users and items, correlation between users, and item similarities. These information matrices will be interacted with to spread information, with the goal of recording the characteristics of each user and item and perfecting the model-based recommendation system. In the following chapters, we also experiment on popular datasets, compare the results to various measures, and discuss the results.

2. Related work

In this chapter, we present important foundations for the proposed model, including publications on collaborative filtering (CF), graph convolution networks (GCN), social recommendation system models, location-based applications, and related data processing techniques.

2.1 Social-aware recommendation system

When online social networks first appeared and were integrated into e-commerce platforms, social recommendation systems were developed. These systems provide users with personalized titles based on their actions, such as clicks, sharing, and commenting. Additional data from social networks is also analyzed to determine the influence of their friends [3]. The vast amount of information on social networks has influenced users, causing them to share the same concerns as their friends [6, 7].

ContextMF is an MF-based algorithm that uses contextual information in the recommendation process. This method combines social and rating data using probabilistic matrix factorization [8]. TrustSVD [4], built on SVD++ [9], includes the impact of friends on social relationships as additional implicit feedback for the observed users. GraphRec is a graph neural network-based algorithm developed for social recommendation tasks. GraphRec employs a novel graph neural network framework to simultaneously capture interactions and opinions in the user-item graph. It coherently models two graphs with different strengths [10]. SocialLGN is a graph neural network-based algorithm specifically designed for social recommendation tasks. SocialLGN employs a graph to learn user and item embeddings by propagating information from the user's social network [11]. It has been shown to outperform other cutting-edge social recommendation methods on several benchmark datasets. The complex relationship between social relationship data and user influence in shopping is difficult to understand because it changes over time and depends on the type of item. The Social Cognitive Self-Monitoring Tri-training (SEPT) model [12] uses three graph encoders to derive temporal signals from user social relationships, user correlations when interested in similar items, and information-sharing to improve recommendation predictions. The self-monitoring learning process in that model increases the effectiveness of the recommendations.

2.2 Graph convolution network

Graph neural network (GNN) is a type of neural network that operates on datasets as graphs [13]. Graphs are data structures that consist of nodes and edges. A node can be either a user or an item, and edges define the relationships between nodes. Inheriting the advantages of GNN, GCN continues to explore the correlation between nodes through more hops by repeating the CF process with the propagation process on the graph. GCNs apply filters to the graph, inspecting nodes and edges that can be used to classify other nodes in the data. GraphSAGE is a type of graph convolutional network that learns node embeddings by gathering data from the node's immediate surroundings [14]. It works by selecting a fixed number of neighbors for each node and aggregating their features to calculate the node embedding. GraphSAGE can be applied to a variety of tasks, including node classification, link prediction, and graph classification. NGCF is a type of graph convolutional network intended for CF tasks [15]. NGCF employs a neural network to learn node embeddings by combining data from the node's local neighborhood and the global graph structure. To compute the node embedding, it uses a weighted sum of the embeddings of its neighbors. LightGCN is a simplified version of NGCF that is intended to be more efficient and scalable [16]. This model learns node embeddings through a simple graph convolution operation that propagates information from the node's neighbors. It computes the node embedding in a similar fashion to NGCF. LightGCN has demonstrated state-of-the-art performance on several benchmark datasets while being significantly faster than other methods. In a recent publication [17], we used the GCN model to combine user friendship signals and interaction data. In that model, we calculated users' influence weights and used them as attention signals, accelerating convergence during the collaborative filtering process.

2.3 Location-based recommendation system

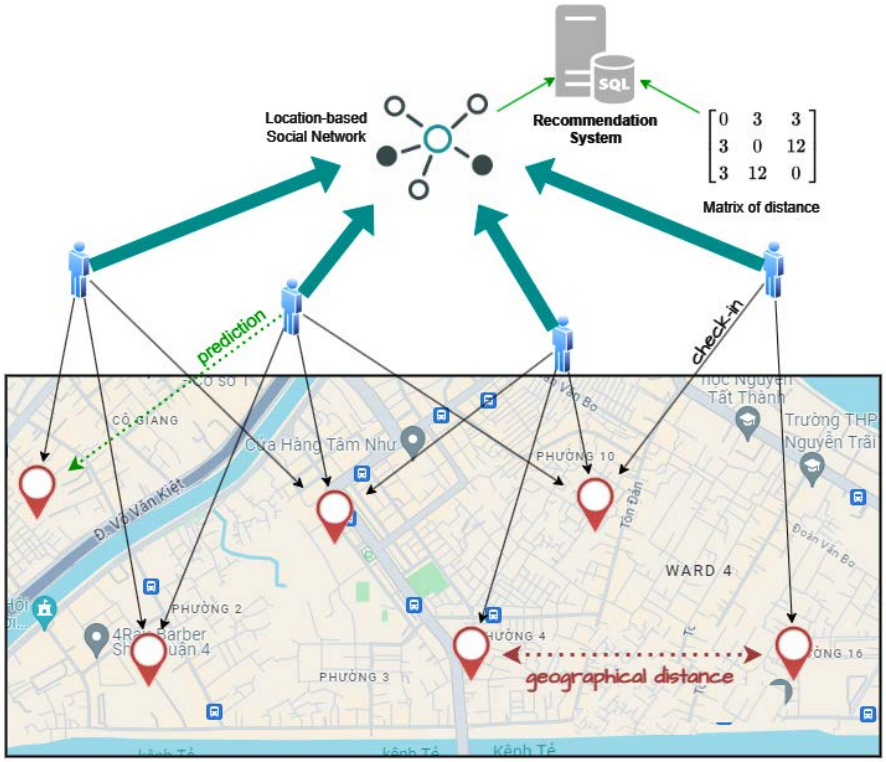

Fig. 1. An overview of a location-based recommendation system that aggregates signals from collaborative filtering of common locations and geographical distance using the Haversine equation.

As shown in **Fig. 1**, the location-based recommendation system will look at the user's check-in history and recommend other suitable locations. We can refer to famous or popular places as points of interest (POI). Traditional POI recommendation methods treat POIs as items and use techniques such as CF and MF [18] on the interaction matrix between users and POIs. However, a POI is more than just an item; it is also associated with geographical information and the user's appearance time at that location, which is why models should consider including those pieces of information in the signal collection process. The PACE model in publication [19] proposed a semi-supervised deep learning model based on CF to mitigate the effects of data sparsity by anticipating users' preferences and contexts. Furthermore, since trust between users influences recommendation results; the model in [20] incorporated a trust impact factor among users into their model. MF was also used in Rank-geofm [21], which combined time and location data into user check-in sequences and calculated location scores. Geo-Teaser in [22] predicts locations by combining a temporal POI embedding model with a geographical preference ranking. GeoIE [23] applied geographical model influence on POIs using the inner product of geo-susceptibility vectors and geography influence. When a user visits multiple POIs over time, they become clustered and are accounted for by Tobler's first law of geography [24]. These clusters also have multi-center characteristics, indicating separate visited locations. A two-dimensional KDE was used to learn the spatial clustering phenomenon [25, 26] and estimate the number of clusters. GeoSAN [27] proposed a self-attention-based POI encoder that integrates into the self-attention network to survey the user's travel history.

3. Proposed model

In this chapter, we propose a POI recommendation model that includes data preprocessing operations, input matrix construction, GCN model propagation processes, loss functions, and prediction methods for recommendations. In **Table 1**, we define the notation used in this publication.

Table 1. Notation in equations and model.

Notation	Explanation
u_i	user i in the recommendation system
i_j	location j in the recommendation system
e_u	output features embedding of users
e_i	output features embedding of locations
e_u^k	embedding of users in layer k^{th}
e_i^k	embedding of items in layer k^{th}
e_s	embedding of social friendship between users
e_d	embedding of the Jaccard index of locations
e_{loc}	embedding of geographical distance between locations
R_{nxm}	check in matrix of n users at m locations
S_{nxn}	matrix contains social relationships between n users
J_{mxm}	matrix contains the Jaccard index between m locations

3.1 Data preprocessing

We propose **Algorithm 1** for removing users with fewer than k check-ins from the dataset. The datasets extracted from social networking platforms contain information about the user's interaction with the item as well as the characteristics of the items in the system. Location-based recommendation systems provide a longitude and latitude value for a given location. The interaction between the user and the item is limited because the user only interacts with a few items. Furthermore, only users with a few check-ins at k locations are retained to ensure that outliers do not skew the recommendation results.

Algorithm 1. Remove outlier items in the original dataset

Input: U × I {Check-ins of users on locations in original dataset}
Output: R ⊆ U × I
original_ratio = |U| / |I|
for each location i ∈ I **do**
    if location i has at least *k* check-ins **then**
        I ← location i
    end if
end for
p ← size of set **I**
q ← p ÷ *original_ratio*
for each user u ∈ U **do**
    set_u ← list of location checked-in by user u
    correlation_u ← Jaccard index between I and set_u
end for
U ← q users with highest correlation_u
R = U × I
return R

This method is known as *k-core* and is commonly used in publications [15, 16]. In our experiment, we will use a *10-core* setting, as is common in similar publications. On the other hand, using location data, we can calculate the correlation between each pair of locations based on their geographical distance. Geographic distance is one of the most important factors influencing users' decisions to move. **Algorithm 1**'s constant *selected_ratio* can be used to adjust the ratio of users to locations in datasets after processing and removing outliers. We can also select a scale whose value is equal to the ratio of the original dataset.

3.1.1 Locations correlation by distance

Geographic location refers to a position on Earth that can be defined by two coordinates: longitude and latitude. POIs are well-known public geographic locations, such as attractions, airports, bus stations, museums, and restaurants. When a user visits a point of interest (POI), he or she can check in using mobile apps and online social networking platforms. The geographic coordinates of this POI are recorded using the mobile device's built-in GPS. The geographical distance between two POIs influences users' decisions about future POIs they want to visit. The Haversine formula can be used to calculate the geographical distance between two locations where information is provided: longitude and latitude, as represented in (1).

$$d = 2r \sin^{-1}\left(\sqrt{\sin^2\left(\frac{\varphi_2-\varphi_1}{2}\right) + \cos\varphi_1 . \cos\varphi_2 . \sin^2\left(\frac{\lambda_2-\lambda_1}{2}\right)}\right) \quad (1)$$

where:

- d is the distance between the two locations,
- r is the radius of the earth, 6371 km,
- φ_1, φ_2 are the latitude of location 1 and 2, respectively,
- λ_1, λ_2 are the longitude of location 1 and 2, respectively.

With a dataset of m locations, we create a matrix D with dimensions $m \times m$ in which each element $D_{i,j}$ represents the distance d between POI i and POI j using (1). That matrix will be used to calculate the geographical distance embedding e_{loc} in our proposed GCN model.

3.1.2 Locations correlation by collaborative filtering

In addition to their geographic location, POIs have other features that entice users to visit them. The number of users who have checked in at both locations can be used to determine their similarity. According to the publication [28], a weight matrix representing similarity can be calculated using the equation $W = R.R^T$, where R is the matrix representing interaction between users and items. The value of each element $W_{i,j}$ is the number of users checking in at locations i and j. However, the frequency with which a user checks in must also be considered, as the more they interact with the system, the more reliable their recommendations become. This data was not recorded in matrix W.

To take advantage of information in both, the number of users who have checked into the same location and how frequently those users interact with the system, we propose using the Jaccard index to determine the logical correlation between locations. If many users visit the two locations at the same time, we anticipate a high degree of similarity between them. As a result, the Jaccard index, shown in **Fig. 2**, is appropriate for measuring this similarity and can be calculated as (2).

$$Jaccard(POI_i, POI_j) = \frac{U_j \cap U_j}{U_j \cup U_j} \quad (2)$$

where U_i and U_j represent the sets of users who have interacted with POI_i and POI_j, respectively.

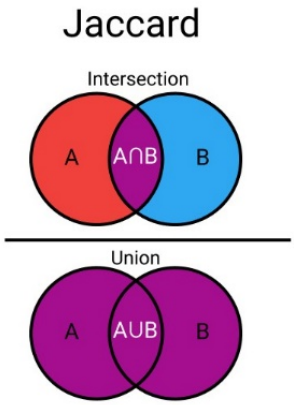

Fig. 2. Calculating Jaccard index from two sets of locations checked-in by user A and user B.

However, it is worth noting that propagation in the GCN model captures collaboration signals from the high-order connectivity graph. That is, the indirect influence of the user on the item is also considered, such as user - item - user - item - and so on [3, 15, 16, 17, 28]. In the process, weighting matrices were added to speed up signal collection. This technique has the side effect of overlearning, in which the system focuses on a small number of users or items while eliminating the possibility of low-interaction users or items. To address this issue, we proposed two approaches: calculate the matrices' normalized values or cluster them into different weighted categories. In this publication, we will look at clustering into quartiles in **Table 2**. After all, the matrix containing convert values will be used to compute the embedding of the Jaccard index in our proposed GCN model.

Table 2. Clustering the correlation between locations by Jaccard index.

Jaccard index quartile	Q1	Q2	Q3	Q4
Denotation	Completely unrelated	Low level correlation	High level correlation	Strongly related
Convert values	0.0	0.05	.5	1.0

3.1.3 Social relation

In the evolution of social networking platforms, users not only post or check in to places but also interact with one another and can become friends on the same or other social networking platforms. Once users become friends, they can receive reviews, advice, and interactions from others in their friend group. These interactions are consistently more powerful than those with strangers [3, 4, 5, 29]. Furthermore, social networking platforms tend to expand a user's network of friends to attract users and increase their time spent on the platform.

In our proposed model, the friendship relationship is extracted from the social networking platform and graphically represented by an undirected graph, then stored as a matrix S which defined by (3). That matrix will be used to compute the embedding of online platform social friendship relationships e_s in our proposed GCN model.

$$S_{i,j} = \begin{cases} 1, & if\ user_i\ and\ user_i\ are\ friends \\ 0, & otherwise \end{cases} \tag{3}$$

3.2 Our proposed GCN model

In this section, we propose a model for deconstructing and analyzing information from the inputs, resulting in a prediction matrix as the output. An overview of our proposed model is shown in **Fig. 3**.

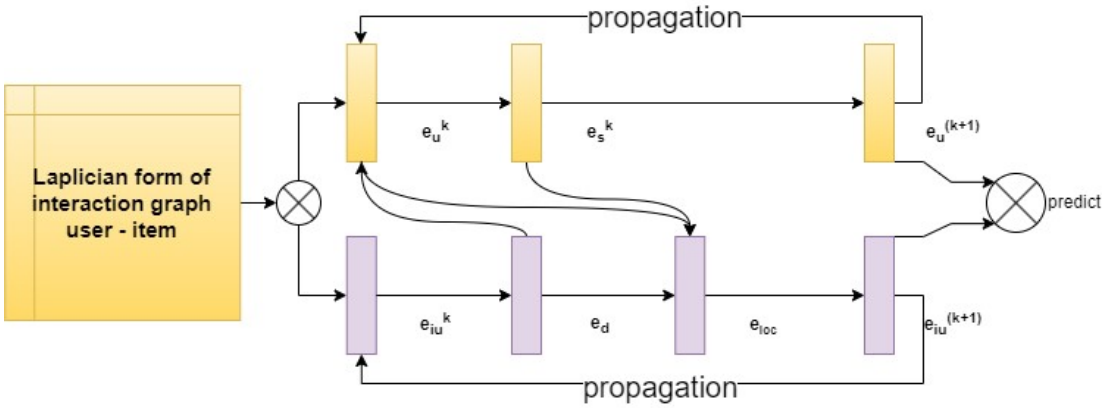

Fig. 3. Overview on our proposed GCN model.

Important components of the model include:

- Matrix R for recording users checking in to locations: In recommendation systems, the interaction between users and items can be represented in two ways: implicitly and explicitly. Implicit datasets store the interaction in binary form, whereas explicit datasets store the user's rating of the item during the interaction. E-commerce systems frequently allow users to rate or provide feedback on items they have purchased or viewed.
- The information enrichment matrices D, J, and S were introduced in the previous section: The first one is matrix D that contains the geographic distance matrix between locations. The second matrix J contains correlation data between locations using the Jaccard index measure. And the last matrix S is the user-extracted friendship data from online social networking platforms. It is important to note that the user friend matrix is optional, as not all e-commerce platforms or recommendation systems collect this information.
- Embeddings in GCN layers: Embeddings are useful for reducing the dimensionality of input information matrices while minimizing information loss. The size of the embedding is passed to the model as a parameter. Initially, the embeddings are

initialized using the Xaivier method [30]. We will discuss embeddings in detail in the following sections.

Algorithm 2. describes in detail the calculation steps for the GCN model with K embedding layers. To capture signals, the user and item embeddings are splitted at each iteration. After that, they are combined and stacked to form the final embedding, which returns for the next iteration as an ego embedding.

Algorithm 2. Propagation by Graph Convolution Network	
	Input: Matrices **A, S, J, D**
	Output: feature embedding of users and items
1	ego_emb, e_s, e_d, e_{loc} = embedding.initialized()
2	$e_s = S.e_s$;
3	$e_d = J.e_d$;
4	$e_{loc} = D.e_{loc}$;
5	**for k** $\in \overline{\mathbf{0}, \boldsymbol{K-1}}$ **do**
6	$embed^k$ = A.ego_emb
7	split $embed^k$ into e_u^k and e_i^k
8	$e_u^{k+1} = f_{combination}(e_u^k, e_s)$
9	$e_i^{k+1} = f_{combination}(e_i^{k-1}, e_d, e_{loc})$
10	$embed^{k+1}$ = concatenate(e_u^{k+1}, e_i^{k+1})
11	ego_emb = $embed^{k+1}$
12	**end for**
13	embed = mean($embed^k$) with k $\in \overline{1, K}$
14	e_u, e_s = embed.split()
15	**return** embedding of users $\mathbf{e_u}$ and items $\mathbf{e_i}$

3.2.1 The check-in matrix A

To realize the process of spreading high-order connectivity, the check-in matrix *R* must be designed in Laplacian form [15]. Then, *A* is a square matrix of size *(n+m)*, where *n* and *m* represent the number of users and locations in the dataset, respectively. *0* is a square matrix in which all elements have the value zero. We illustrated the structure of matrix A in **Fig. 4**.

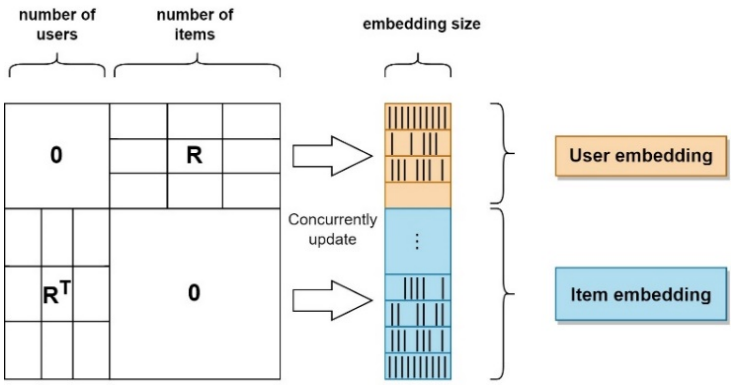

Fig. 4. The check-in matrix layout enables concurrent propagation updates for embeddings of users and items.

The Laplacian form of R can be calculated as (4).

$$A = \begin{bmatrix} 0 & R \\ R^T & 0 \end{bmatrix}$$
$$\tilde{A} = D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \tag{4}$$

where D is the diagonal degree matrix and 0 is the all-zero matrix.

3.2.2 Propagation process for user embedding

The feature signals are captured from input matrices as in (5), our model propagates the user embeddings e_u^k and item embedding e_i^k on the light graph convolution model [16] at every k^{th} layer. After each calculating of each layer, we obtains embeddings $e_{ua}^{(k+1)}$ and $e_s^{(k+1)}$, that contains signals of high-order propagation. The embedding $e_{ua}^{(k+1)}$ keeps user-item interaction signals in the matrix A, while the embedding $e_s^{(k+1)}$ keeps social signals between users in matrix S. The matrix S is optional because some e-commerce platforms do not store user friendship information.

$$e_{ua}^{(k+1)} = \sum_{i \in \mathrm{N}_u^A} \frac{1}{\sqrt{\left|\mathrm{N}_u^A\right|\left|\mathrm{N}_i^A\right|}} e_i^{(k)}$$
$$e_s^{(k+1)} = \sum_{s \in \mathrm{N}_u^S} \frac{1}{\sqrt{\left|\mathrm{N}_u^S\right|\left|\mathrm{N}_s^S\right|}} e_u^{(k)} \tag{5}$$

where $\left|N_q^X\right|$ denotes the number of nearby items of user q in the matrix X with X = [A, S].

The foundation of (5) is based on the symmetric normalization element, which was used in most GCN models [15, 16, 17, 29]. Then the $(k+1)^{th}$ user embeddings are combined by (6).

$$e_u^{(k+1)} = COMBINATION_K\left(e_{ua}^{(k+1)}, e_s^{(k+1)}\right) \tag{6}$$

After K iterations of the propagation process, we receive (K) users' embeddings. The final embedding of users will be calculated as (7).

$$e_u = \frac{1}{K}\sum_{k=1}^{K} e_u^{(k)} \tag{7}$$

The result embedding is the mean value of all embeddings at all layers calculated with (7), and this function can be replaced with others such as sum, maximum, and median.

3.2.3 Propagation process for item embedding

Item embedding is also calculated by (8) alongside the propagation process in user embedding.

$$\begin{aligned} e_{loc}^{(k+1)} &= \sum_{loc \in \mathrm{N}_i^C} \frac{1}{\sqrt{\left|\mathrm{N}_i^C\right|\left|\mathrm{N}_c^C\right|}} e_i^{(k)} \\ e_d^{(k+1)} &= \sum_{d \in \mathrm{N}_i^D} \frac{1}{\sqrt{\left|\mathrm{N}_i^D\right|\left|\mathrm{N}_d^D\right|}} e_i^{(k)} \\ e_{iu}^{(k+1)} &= \sum_{u \in \mathrm{N}_i^A} \frac{1}{\sqrt{\left|\mathrm{N}_i^A\right|\left|\mathrm{N}_i^A\right|}} e_u^{(k)} \end{aligned} \tag{8}$$

At the end of iteration, the item embedding is combined by (9), which can be a sum function or the application of weights to component embeddings.

$$e_i^{(k+1)} = COMBINATION_K\left(e_{loc}^{(k+1)}, e_d^{(k+1)}, e_{iu}^{(k+1)}\right) \tag{9}$$

The result embedding of items will be obtained by (10).

$$e_i = \frac{1}{K}\sum_{k=1}^{K} e_i^{(k)} \tag{10}$$

3.2.4 Convolution on GNN

GNN models implemented the message-passing technique by extracting signals from inputs and aggregating them into output embeddings [31, 32]. With GCN, output embedding continues to propagate through the graph's structure, resulting in collaborative filtering of both users and items.

- Extract signals: the signals propagated from users to users by (11) and items to users by (12).

$$m_{u \leftarrow u} = e_u + e_s \tag{11}$$

$$m_{u \leftarrow i} = \frac{1}{\sqrt{|N_u||N_i|}} * (e_i + e_{loc} + e_d) \tag{12}$$

- Combine signals: all received messages from nearby users of user u are combined to refine the user embedding, as in (13). The activation function LeakyReLU adds messages encoded with positive and also slightly negative signals [33]. The correlation of users is obtained from both collaborative filtering and addition information matrices J, D, and S.

$$e_u = LeakyReLU\left(m_{u \leftarrow u} + \sum_{i \in N_u} m_{u \leftarrow i}\right) \tag{13}$$

3.2.5 Optimization on the prediction score

After several propagation iterations, the output embedding vector E^* will be converged, and the prediction score between user u_i and location i_j can be calculated by (14).

$$\widehat{y_{ui}} = e_u{}^{\top} e_i \tag{14}$$

We proposed a sampling function and prediction method to evaluate precision and recall score of our model. The sampling process must include both positive and negative samplers. The positive sampler will select locations where the user has previously interacted, whereas the negative sampler will select locations where the user has never interacted. In the prediction evaluation phase, the set of locations chosen by the positive sampler will be used to evaluate precision, whereas the other set of locations chosen by the negative sampler will affect the recall.

Bayesian personalized ranking (BPR) is the best choice for implementing the loss function because it is the most effective ranking method for datasets with implicit feedback [34]. We use two pooling observable sets: $\Omega^{+}{}_{ui}$ is observed check-ins and $\Omega^{-}{}_{uj}$ is unobserved check-ins. The loss function has been implemented by (15).

$$Loss_{BPR} = \sum_{\Omega^{+}_{ui}} \sum_{\Omega^{-}_{uj}} -ln\sigma\left(\widehat{y_{ui}} - \widehat{y_{uj}}\right) + \lambda \|\Phi\|_2^2 \tag{15}$$

where Φ is embedding E^{*} and $\sigma(.)$ is sigmoid function and λ controls the regularization.

4. Experiments and results

To validate our surveys and proposed models, we conducted experiments on common site datasets. We compared the empirical results to baseline models, such as BPR-MF, NGCF, LightGCN, and WiGCN.

4.1 Datasets description

We conduct experiments on BrightKite, NYC and Gowalla. Each dataset consists of a file of check-in records that include *user_id*, *location_id*, and *check-in_time*; and a friendship relation record file, where each record consists of two *user_id* who are friends on a social platform. Some additional files can provide more details on the user or location. We provide the statistics of all datasets in **Table 3**.

- **BrightKite**: a location-based social networking service provider that allows users to check in and share their locations while also providing ratings and comments. The friendship network has over 50 thousand nodes and over 214 thousand edges. It was collected via their public API. The network was collected using directed relations, but we rebuilt it with undirected edges when there was a friendship in both directions. This dataset is available in the SNAP project [35].
- **NYC**: The NYC Open Data is a collection of data sets that provide information about different aspects of New York City, such as transportation, education, health, the environment, and culture [36]. New York City agencies and other partners publish this data, which is freely available to the public. The data can be used to analyze and understand the city's operations, as well as to create new applications and services for the public.
- **Gowalla**: Gowalla is one of several web-based applications that incorporate location-based social networking [35]. Users can check-in and share public locations with their friends. Gowalla's friendship network is represented as an undirected graph.

Table 3. Statistical of datasets.

Dataset	Users	Locations	Check-ins	Density	Social links
BrightKite	7,946	26,094	351,184	0.00169	77,660
NYC	504	3,312	29,665	0.018	unavailable
Gowalla	23,499	43,043	866,964	0.00086	208,328

4.2 Baseline models

We use the following state-of-the-art models as base lines to compare with our proposed model.

- **BPR-MF** [34]: A personalized ranking model based on a generic optimization criterion, the maximum posterior estimator, derived from a Bayesian problem analysis. The model is equipped with a generic optimization learning algorithm that was created using stochastic gradient descent and bootstrap sampling.
- **NGCF** [15]: This is one of the most effective GCN models. It propagates embeddings with multiple iterations to capture high-order connectivity in the interaction graph before stacking them on the output. The latent vectors contain the collaborative signal, which leads to higher precision.
- **LightGCN** [16]: This model focuses on neighborhood aggregation for collaborative filtering using NGCF. It removes weight matrices and activation functions. The users' and items' embeddings for the interaction graph are learned using linear propagation. The resulting embedding is the sum of all the learned embeddings.
- **WiGCN** [28]: The model uses a weighted matrix in the propagation process to strengthen the signals. This matrix represents users' influence on other users by calculating the common items between them. This results in more data collection propagation and improved recommendation performance.

4.3 Evaluation criteria

It is critical to select the appropriate evaluative measure for each algorithm [37]. A common evaluation method is to divide a dataset into a train set (typically containing 80 percent of the data) and a test set (the remaining 20 percent of the data). The algorithm is then applied to the train set to make predictions, which are evaluated on the test set. The difference between the actual data value and the predicted result indicates the accuracy of the experimental model's algorithm. This error can be represented by Mean Absolute Error (MAE) and Root Mean Square Error (RMSE). In addition to MAE and RMSE, precision, recall, scalability, learning time, memory consumption, and interpretability are important to consider when evaluating the recommended system.

In implicit datasets, user check-ins to locations are recorded in binary format. Algorithms use accuracy measures for classification, with precision and recall being the most used metrics [38, 39]. Precision is the ratio of correct predictions on the test set, whereas recall is the algorithm's sensitivity, or the proportion of relational assertions extracted from the test set. (16) is used to calculate precision and recall measurements.

$$Precision = \frac{True\ positive}{True\ positive + False\ positive}$$
$$Recall = \frac{True\ positive}{True\ positive + False\ negative} \tag{16}$$

where:

- *True positive* is set of hits predicting check-in exists of users on locations,
- *False positive* is set of mispredictions of check-in,
- *False negative* presents the number of predicted check-ins not being presented on the test set.

Furthermore, the discounted cumulative gain score (DCG) is a method that labeled each result [40]. It accumulates a gain function G applied to the label of each result across the result vector, which is scaled by a discount function D based on the result rank. The DCG is then normalized by dividing it by the DCG of an ideal result vector, I. This yields the normalized discounted cumulative gain (NDGC), as in (17)

$$NDCG@K = \frac{DCG@K}{IDCG@K} \tag{17}$$

In our experiments, we use precision and recall measurement in (16) and the NDCG score in (17) to consider 5 and 10 items, respectively.

4.4 Overall result comparison

We have summarized the experimental results in **Table 4**. Because the NYC dataset lacks friendship data, POI-3 returns no results. We divide the table into two parts: Top-5 and Top-10, with 5 and 10 selected locations for testing, respectively. When matrix information is added, the results improve in precision and recall.

Table 4. Overall performance comparisons of LBRS.

	Dataset / Model	BrightKite			NYC			Gowalla		
		recall	**precision**	**ndcg**	**recall**	**precision**	**ndcg**	**recall**	**precision**	**ndcg**
Top-5 locations	BPR-MF	0.0152	0.0205	0.0261	0.0172	0.0196	0.0234	0.0319	0.0420	0.0487
	NGCF	0.0180	0.0247	0.0309	0.0185	0.0375	0.0387	0.0317	0.0422	0.0493
	LightGCN	0.0204	0.0295	0.0368	0.0204	0.0468	**0.0551**	0.0325	0.0431	0.0502
	WiGCN	0.0201	0.0285	0.0374	0.0207	0.0436	0.0512	0.0321	0.0439	0.0487
	POI-1	0.0206	0.0294	0.0366	0.0201	0.0468	0.0551	0.0325	0.0442	0.0493
	POI-2	0.0231	0.0315	0.0407	**0.0210**	**0.0480**	0.0530	0.0367	0.0451	0.0496
	POI-3	**0.0237**	**0.0341**	**0.0424**	n/a	n/a	n/a	**0.0343**	**0.0465**	**0.0502**
Top-10 locations	BPR-MF	0.0217	0.0149	0.0253	0.0110	0.0254	0.0279	0.0414	0.0322	0.0506
	NGCF	0.0218	0.0224	0.0296	0.0210	0.0263	0.0388	0.0411	0.0313	0.0483
	LightGCN	0.0304	0.0218	0.0353	0.0254	0.0294	0.0407	0.0450	0.0338	0.0531
	WiGCN	0.0301	0.0221	0.0363	0.0247	0.2812	0.0401	0.0452	0.0341	0.0544
	POI-1	0.0302	0.0217	0.0394	0.0263	0.0304	0.415	0.0453	0.0344	0.0539
	POI-2	0.0307	0.0221	0.0401	**0.0284**	**0.0334**	**0.0416**	0.0455	0.0348	0.0541
	POI-3	**0.0342**	**0.0245**	**0.0401**	n/a	n/a	n/a	**0.0464**	**0.0354**	**0.0549**

4.5 Ablation studies

We repeat the experiments with different parameters to determine the impact of each component on system results. This also helps us explain the machine learning model more clearly.

4.5.1 The influence of each matrix

To assess the impact of these factors on the model, we created three variations, which are detailed below and summarized in **Table 5**.

- POI-1: Based on the collaborative filtering model using GCN, we add to the model one of the correlation matrices between locations calculated based on the geographical distance information of each pair of locations by Haversine equation (1), or the correlation of locations based on CF with Jaccard index (2).
- POI-2: We incorporate into our proposed model a combination of location correlation based on CF results (information from each pair of locations with the same number of check-in users) and location correlation by geographical distance.
- POI-3: We incorporate the social link matrix derived from online social networking platforms into our POI-2 model.

Table 5. Overall performance comparisons of LBRS.

Embedding	Check-ins R	distance D	Jaccard index J	Social link S
LightGCN	☑			
POI-1	☑	☑ max (D, J)		
POI-2	☑	☑	☑	
POI-3	☑	☑	☑	☑

4.5.2 Computational speed

On e-commerce platforms, the calculation speed of the recommendation model is critical. A model must not only have precision and recall, but also converge after a limited number of epochs. In previous GCN models, precision and recall will increase gradually because iterations take time to propagate through the Laplacian form matrix A. In the model we proposed, additional information matrices aided the embeddings in receiving signals and rapidly reaching states specific to users and locations. We show the increase in precision and recall after the number of epochs of three models: LightGCN, POI-1, and POI-3 (or POI-2 if social links are not available) on the NYC dataset in **Fig. 5** and the BrightKite dataset in **Fig. 6**.

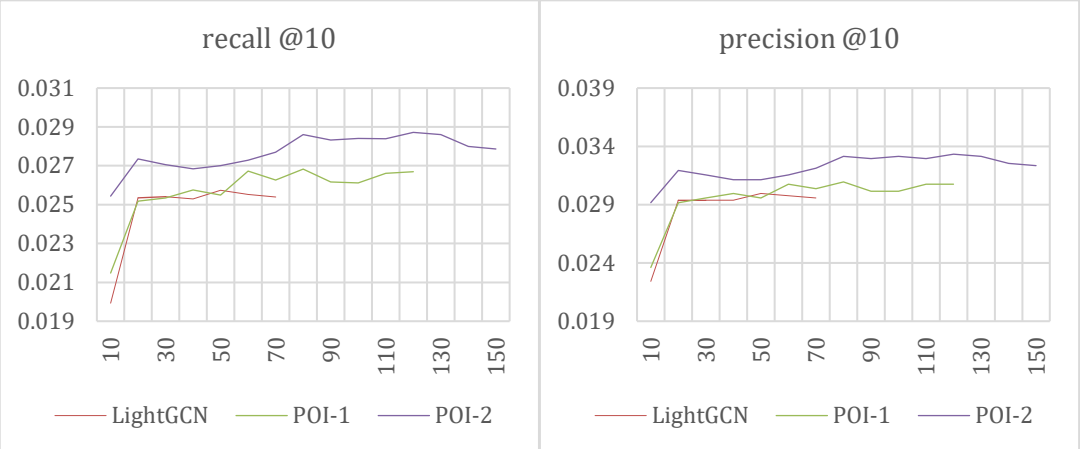

Fig. 5. Values of the recall and precision measurement after number of epochs on NYC dataset.

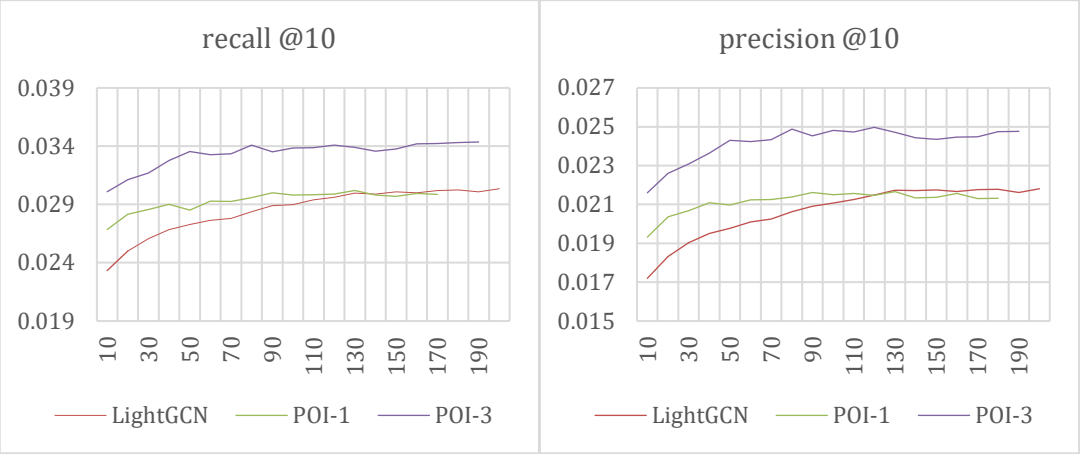

Fig. 6. Values the recall and precision measurement after number of epochs on BrightKite dataset.

4.5.3 The size of embedding

We tested our POI-2 model on the NYC dataset with sizes of 32, 64, 96, 128, and 256. The accuracy trade-off is the size of the memory used to store the matrices during the calculation. We conclude that 64 is an appropriate size for the datasets in this publication. When the original matrices are sparse with implicit binary data, the embedding size has little effect on experiments with other base line models. In **Fig. 7**, we show the difference in recall and precision values for different embedding sizes.

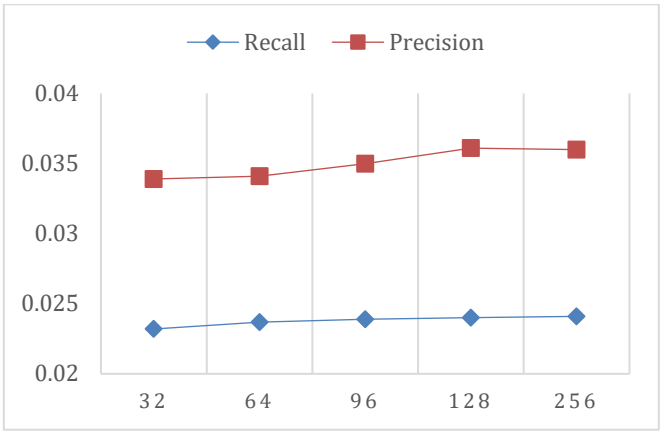

Fig. 7. Value of the recall and precision with several size of embedding on POI-3 model with BrightKite dataset.

4.5.4 Number of layers in GCN

The advantage of GCN models over GNNs is that they repeat the signal propagation process multiple times, which we refer to as the number of layers in the proposed model. The weights and embeddings in the model are reused at each iteration, increasing their accuracy. However, if repetition occurs too frequently, overfitting can occur, and the model's accuracy decreases. We ran tests on the NYC and BrightKite datasets using layer numbers 1, 2, 3, and 4. **Fig. 8** shows the accuracy of each layer for the NYC dataset using three models: NGCF, LightGCN, and POI-2.

With three layers, the model achieves the highest precision. When there are four layers, accuracy tends to decrease. This conclusion is also consistent with the discussion in the publication [16] using the model LightGCN. The majority of GCN models in publications use 3-layer parameters [15, 17, 29].

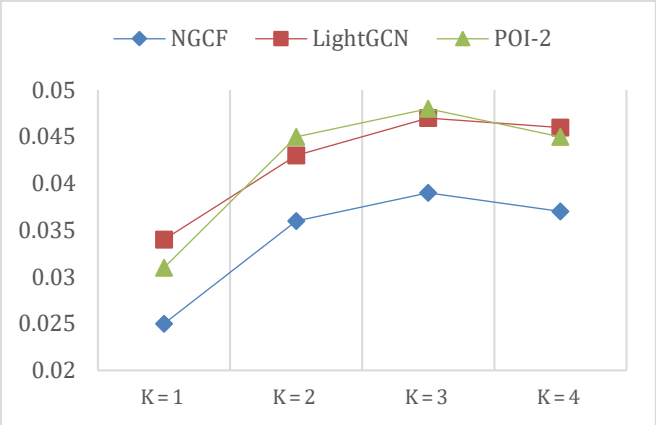

Fig. 8. Comparison the precision measurement after each propagation iteration on NYC dataset.

5. Conclusion

Experimental results show that correlations from both the user and item sides contribute to feature embeddings and communicate with one another during the propagation process. Furthermore, the proposed model's input information blocks are designed as modules, making the model easily customizable and explainable. Mining geographical distances is becoming popular in recommendation systems as users move around and use smartphones equipped with GPS positioning and continuous tracking. However, the issue of location recommendation requires further investigation because, unlike traditional recommendations, the location a user wishes to visit is very closely related to the chain of recently visited locations. Instead of treating all locations equally, the sequential recommendation system will evaluate the model based on the time order in which a user visits each location. The group recommendation problem is also difficult because it will provide recommendations to many users when they visit as a group of friends.

References

[1] R. Mehta and K. Rana, "A review on matrix factorization techniques in recommender systems," in *Proc. of 2017 2nd International Conference on Communication Systems, Computing and IT Applications (CSCITA)*, pp.269-274, 2017. Article (CrossRef Link)

[2] Xun Zhou, Jing He, Guangyan Huang, Yanchun Zhang, "SVD-based incremental approaches for recommender systems," *Journal of Computer and System Sciences*, vol.81, no.4, pp.717-733, 2015. Article (CrossRef Link)

[3] Ma, H., Zhou, D., Liu, C., Lyu, M. R. & King, I., "Recommender systems with social regularization," in *Proc. of WSDM '11: Proceedings of the fourth ACM international conference on Web search and data mining*, pp.287-296, 2011. Article (CrossRef Link)

[4] Guo, G., Zhang, J. & Yorke-Smith, N., "A Novel Recommendation Model Regularized with User Trust and Item Ratings," *IEEE Transactions on Knowledge And Data Engineering*, vol.28, no.7, pp.1607-1620, 2016. Article (CrossRef Link)

[5] Jiang, M., Cui, P., Wang, F., Zhu, W. & Yang, S., "Scalable Recommendation with Social Contextual Information," *IEEE Transactions on Knowledge and Data Engineering*, vol.26, no.11, pp.2789-2802, 2014. Article (CrossRef Link)
[6] Bond, R. M., Fariss, C. J., Jones, J. J., Kramer, A. D. I., Marlow, C., Settle, J. E. & Fowler, J. H., "A 61-million-person experiment in social influence and political mobilization," *Nature*., vol.489, pp.295-298, 2012. Article (CrossRef Link)
[7] Qiu, J., Tang, J., Ma, H., Dong, Y., Wang, K. & Tang, J., "DeepInf: Modeling Influence Locality in Large Social Networks," 2018. Article (CrossRef Link)
[8] Wang, J., Bagul, D. & Srihari, S., "ContextMF : A Fast and Context-aware Embedding Learning Method for Recommendation Systems," 2018. Article (CrossRef Link)
[9] Koren, Y., "Factorization Meets the Neighborhood: A Multifaceted Collaborative Filtering Model," in *Proc. of KDD '08: Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp.426-434, 2008. Article (CrossRef Link)
[10] Fan, W., Ma, Y., Li, Q., He, Y., Zhao, E., Tang, J., & Yin, D., "Graph Neural Networks for Social Recommendation," in *Proc. of WWW '19: The World Wide Web Conference*, pp.417-426, 2019. Article (CrossRef Link)
[11] Liao, J., Zhou, W., Luo, F., Wen, J., Gao, M., Li, X. & Zeng, J., "SocialLGN: Light graph convolution network for social recommendation," *Information Sciences*, vol.589, pp.595-607, 2022. Article (CrossRef Link)
[12] Yu, J., Yin, H., Gao, M., Xia, X., Zhang, X. & Hung, N. Q. V., "Socially-Aware Self-Supervised Tri-Training for Recommendation," in *Proc. of KDD '21: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp.2084-2092, 2021. Article (CrossRef Link)
[13] Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W. L. & Leskovec, J., "Graph Convolutional Neural Networks for Web-Scale Recommender Systems," in *Proc. of KDD '18: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp.974-983, 2018. Article (CrossRef Link)
[14] Hamilton, W. L., Ying, R. & Leskovec, J., "Inductive Representation Learning on Large Graphs," in *Proc. of NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp.1025-1035, 2017. Article (CrossRef Link)
[15] Wang, X., He, X., Wang, M., Feng, F. & Chua, T., "Neural Graph Collaborative Filtering," in *Proc. of SIGIR'19: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp.165-174, 2019. Article (CrossRef Link)
[16] He, X., Deng, K., Wang, X., Li, Y., Zhang, Y. & Wang, M., "LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation," in *Proc. of SIGIR '20: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp.639-648, 2020. Article (CrossRef Link)
[17] T. T. Tran, V. Snasel and L. T. Nguyen, "Combining Social Relations and Interaction Data in Recommender System With Graph Convolution Collaborative Filtering," *IEEE Access*, vol.11, pp.139759-139770, 2023. Article (CrossRef Link)
[18] Su, X. & Khoshgoftaar, T. M., "A Survey of Collaborative Filtering Techniques," *Advances in Artificial Intelligence*, Oct. 2009. Article (CrossRef Link)
[19] Yang, C., Bai, L., Zhang, C., Yuan, Q. & Han, J., "Bridging Collaborative Filtering and Semi-Supervised Learning: A Neural Approach for POI Recommendation," in *Proc. of KDD '17: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp.1245-1254, 2017. Article (CrossRef Link)
[20] Wang, W., Chen, J., Wang, J., Chen, J., Liu, J. & Gong, Z., "Trust-Enhanced Collaborative Filtering for Personalized Point of Interests Recommendation," *IEEE Transactions on Industrial Informatics*, vol.16, no.9, pp.6124-6132, 2020. Article (CrossRef Link)
[21] Chamberlain, B. P., Hardwick, S. R., Wardrope, D. R., Dzogang, F., Daolio, F. & Vargas, S., "Scalable Hyperbolic Recommender Systems," arXiv:1902.08648, 2019. Article (CrossRef Link)

[22] Zhao, S., Zhao, T., King, I. & Lyu, M. R., "Geo-Teaser: Geo-Temporal Sequential Embedding Rank for Point-of-Interest Recommendation," in *Proc. of WWW '17 Companion: Proceedings of the 26th International Conference on World Wide Web Companion*, pp.153-162, 2017. Article (CrossRef Link)
[23] Wang, H., Shen, H., Ouyang, W. & Cheng, X., "Exploiting POI-Specific Geographical Influence for Point-of-Interest Recommendation," in *Proc. of IJCAI'18: Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pp.3877-3883, Jul. 2018. Article (CrossRef Link)
[24] Waters, N., Tobler's First Law of Geography. Dec. 2017. Article (CrossRef Link)
[25] Lian, D., Ge, Y., Zhang, F., Yuan, N. J., Xie, X., Zhou, T. & Rui, Y., "Scalable Content-Aware Collaborative Filtering for Location Recommendation," *IEEE Transactions on Knowledge and Data Engineering*, vol.30, no.6, pp.1122-1135, 2018. Article (CrossRef Link)
[26] Lian, D., Zheng, K., Ge, Y., Cao, L., Chen, E. & Xie, X., "GeoMF++: Scalable Location Recommendation via Joint Geographical Modeling and Matrix Factorization," *ACM Transactions on Information Systems (TOIS)*, vol.36, no.3, Mar. 2018. Article (CrossRef Link)
[27] Lian, D., Wu, Y., Ge, Y., Xie, X. & Chen, E., "Geography-Aware Sequential Location Recommendation," in *Proc. of KDD '20: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp.2009-2019, Aug. 2020. Article (CrossRef Link)
[28] Tran, T. T. & Snasel, V., "Improvement Graph Convolution Collaborative Filtering with Weighted Addition Input," in *Proc. of 14th Asian Conference, Intelligent Information and Database Systems, Springer*, pp.635-647, 2022. Article (CrossRef Link)
[29] Loc Tan Nguyen, Tin T. Tran, "CombiGCN: An effective GCN model for Recommender System," in *Proc. of 12th International Conference on Computational Data and Social Networks*, pp.111-119, 2023. Article (CrossRef Link)
[30] Li, J., Song, Y., Song, X., & Wipf, D., "On the Initialization of Graph Neural Networks," in *Proc. of the 40th International Conference on Machine Learning*, vol.202, pp.19911-19931, 2023. Article (CrossRef Link)
[31] Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K. & Jegelka, S., "Representation Learning on Graphs with Jumping Knowledge Networks," in *Proc. of the 35th International Conference On Machine Learning*, vol.80, pp.5453-5462, 2018. Article (CrossRef Link)
[32] Hamilton, W. L., Ying, R. & Leskovec, J., "Inductive Representation Learning on Large Graphs," in *Proc. of NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp.1025-1035, 2017. Article (CrossRef Link)
[33] Maas, A. L., Hannun, A. Y., Ng, A. Y., "Rectifier Nonlinearities Improve Neural Network Acoustic Models," in *Proc. of 30th International Conference on Machine Learning (ICML)*, vol.28, 2013. Article (CrossRef Link)
[34] Rendle, S., Freudenthaler, C., Gantner, Z. & Schmidt-Thieme, L., "BPR: Bayesian Personalized Ranking from Implicit Feedback," in *Proc. of UAI '09: Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pp.452-461, 2009. Article (CrossRef Link)
[35] Cho, E., Myers, S. A. & Leskovec, J., "Friendship and Mobility: User Movement in Location-Based Social Networks," in *Proc. of KDD '11: Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp.1082-1090, 2011. Article (CrossRef Link)
[36] NYC Open Data. [Dataset]. New York City, 2023. https://opendata.cityofnewyork.us/
[37] Herlocker, J. L., Konstan, J. A., Terveen, L. G. & Riedl, J. T., "Evaluating Collaborative Filtering Recommender Systems," *ACM Transactions on Information Systems (TOIS)*, vol.22, no.1, pp.5-53, 2004. Article (CrossRef Link)
[38] Sarwar, B., Karypis, G., Konstan, J. & Riedl, J., "Application of Dimensionality Reduction in Recommender System -- A Case Study," in *Proc. of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Aug. 2000. Article (CrossRef Link)

[39] Sarwar, B., Karypis, G., Konstan, J. & Riedl, J., “Analysis of Recommendation Algorithms for E-Commerce,” in *Proc. of EC '00: Proceedings of the 2nd ACM conference on Electronic commerce*, pp.158-167, 2000. Article (CrossRef Link)
[40] Najork, M. & McSherry, F, “Computing Information Retrieval Performance Measures Efficiently in the Presence of Tied Scores,” in *Proc. of 30th European Conference on IR Research (ECIR)*, Apr. 2008. Article (CrossRef Link)

Tin T. Tran received the M.S. degree in Computer Science from HCMUT—University of Technology, Ho Chi Minh city, Vietnam in 2012 and is studying a PhD program at VSB—Technical University of Ostrava, Czech Republic. He is working as a lecturer at the Faculty of Information Technology, Ton Duc Thang university, Vietnam. His research interests include artificial intelligence, social network, graph neural network, IOT and location-based application.

Vaclav Snasel research and development experience include over 25 years in the Industry and Academia. He works in a multi-disciplinary environment involving artificial intelligence, multidimensional data indexing, conceptual lattice, information retrieval, semantic web, knowledge management, data compression, machine intelligence, neural network, web intelligence, data mining and applied to various real-world problems. He has authored/co-authored several refereed journal/conference papers and book chapters. He has published more than 400 papers. He has supervised many Ph.D. students from the Czech Republic, Jordan, Yemen, Slovakia, Ukraine, and Vietnam.

Thuan Q. Nguyen is working at the Library of the Open University - Ho Chi Minh city and has extensive experience in managing databases of documents and learner interaction. He received a master's degree in Computer Science in 2012 and is researching in the field of database system.