Ứng dụng học đồ thị trong hệ gợi ý du lịch

TÓM TẮT

Trong bối cảnh ngành du lịch Việt Nam đang phát triển mạnh mẽ, việc hỗ trợ du khách lựa chọn các điểm tham quan phù hợp với sở thích cá nhân trở nên rất quan trọng. Bài báo này đề xuất một hệ thống gợi ý du lịch ứng dụng học đồ thị, nhằm đưa ra các gợi ý về điểm tham quan dựa trên hồ sơ người dùng. Hệ thống gợi ý được xây dựng dựa trên Graph Neural Network, một kỹ thuật học máy tiên tiến cho phép học các đặc trưng và mối quan hệ từ dữ liệu có cấu trúc đồ thị. Hệ thống đang được triển khai và thử nghiệm trên mạng Internet.

Từ khóa: *Hệ gợi ý, gợi ý, học đồ thị, mạng Nơ-ron đồ thị, du lịch.*

Application of graph learning in travel recommendation system

ABSTRACT

In the context of Vietnam's rapidly growing tourism industry, supporting tourists in choosing attractions that suit their personal interests has become very important. This paper proposes a travel recommendation system using Graph learning to provide suggestions for attractions based on user profiles. The recommendation system is built on Graph Neural Network, an advanced machine learning technique that allows learning features and relationships from graph-structured data. The system is being deployed and tested on the Internet.

Keywords: *Recommender systems, recommendations, graph learning, graph neural networks, travel.*

1. INTRODUCTION

With the advancement of information technology, an increasing number of products and services are being introduced to the digital space. As the volume of available products and services grows, users face significant challenges in selecting the most suitable ones. On the other hand, personalized recommendations, those based on user profile information, can greatly enhance the likelihood of selecting appropriate products. Therefore, the application of recommender systems (RSs) to provide appropriate suggestions is very important.[1-3]

To build recommender systems, one of the emerging approaches being adopted is graph learning. Graph learning is a technique that applies machine learning to graph-structured data[4]. This is an emerging technique in the field of artificial intelligence and has been developing rapidly in recent years. In practice, the data collected and provided to recommender systems can naturally be represented in a graph structure. Specifically, entities within the recommender system, including users, items, attributes, and contexts, are often tightly interconnected, either explicitly or implicitly, forming either homogeneous or heterogeneous graphs. Moreover, graph learning has the capability to learn complex relationships. In fact, many graph learning techniques have been developed to understand relationships modeled by graphs[5-7]. Consequently, applying graph learning in recommender systems is one of the research directions that is of interest and urgency today.

In Vietnam, the tourism industry has seen significant investment and robust growth in recent years. Vietnam, with its numerous renowned attractions (such as the ancient town of Hoi An, Sapa, Ha Long Bay, Fansipan Mountain, and more), is recognized internationally as an ideal destination. Travel magazines consistently rank Vietnam among the top attractive destinations, showing the appeal of Vietnamese tourism. Alongside the growth of the tourism industry, choosing sightseeing spots to plan a trip has become increasingly challenging. Travelers often spend considerable time and effort reading reviews on various attractions or consulting friends for travel recommendations. This necessitates the development of systems that can recommend attractions to tourists based on their preferences.

Based on these issues, we have conducted research to develop a tourism recommender system using graph learning. The system is designed to provide recommendations for tourist attractions based on user preference profiles. This system is currently being tested online at the following address: https://travel.fansis.vn.

2. GRAPH LEARNING IN RS

2.1. Problem Definition

The recommendation system allows making predictions about user preferences for products/services, thereby suggesting items that users might be interested in. With a graph learning approach, the recommendation system can be modeled as a bipartite graph[6]. Here, each vertex represents either a user or an item, meaning there are two types of vertices: user vertices and item vertices. Each edge connecting a user vertex to an item vertex represents the interaction between the user and the corresponding item. These interactions can be views, purchases, ratings, and more.

Formally, let U denote the set of users and I denote the set of items. We construct a graph $G = (V, E)$ where V is the set of vertices and E is the set of edges representing recorded interactions in history. In other words, $E = \{(u, i) |\, u \in U, i \in I, r(u, i) \neq 0\}$ where $r(u, i)$ denotes the interaction of user $u \in U$ with item $i \in I$. If the interaction is a rating given by user u on item i, we can define u interacting with i when u's rating on i is greater than a certain threshold θ (where $\theta \geq 1$).

The input to the RS is the set of interactions between each user and item. The task of the RS is to predict items that users may be interested in the future. This can be viewed as a link prediction problem in the bipartite graph G (as illustrated in *Figure 1*). In other words, we need to find a function f, such that $f(u, i)$ represents the likelihood of interaction between user u and item i.

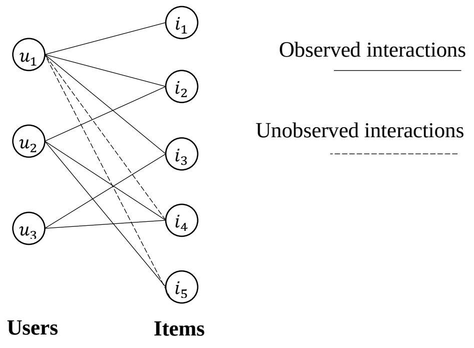


Figure 1. Bipartite graphs represent interactions between users and items.

2.2. The Graph Neural Network Approach

The Graph Neural Network (GNN) approach is a method that applies neural networks to data structured as graphs. GNNs leverages this structure to learn features and relationships from the data, offering better efficiency compared to traditional methods. GNNs enables RS to exploit the complex structure of the user-item graph to uncover deeper hidden relationships. For instance, a RS using GNNs can analyze not only the interaction history of users but also the relationships between items, this means capturing collaborative signals from higher-order neighbors (multi-hop connections), thereby providing more accurate recommendations to each user[7].

GNNs focus on learning representations of nodes by propagating information from neighboring nodes through multiple iterations on the graph structure and aggregating it. This is similar to the Convolutional Neural Network (CNN) technique, with the key difference being that CNNs operate on grid-like structures where the number and order of neighbors are fixed, whereas in GNNs, the number of neighbors is determined by the historical interaction data.

2.3 Graph Neural Network Architecture

The architecture of the GNN model consists of four components[8,9] (illustrated in *Figure 2*):

- Embeddings
- Graph Convolution
- Layer Combination
- Prediction

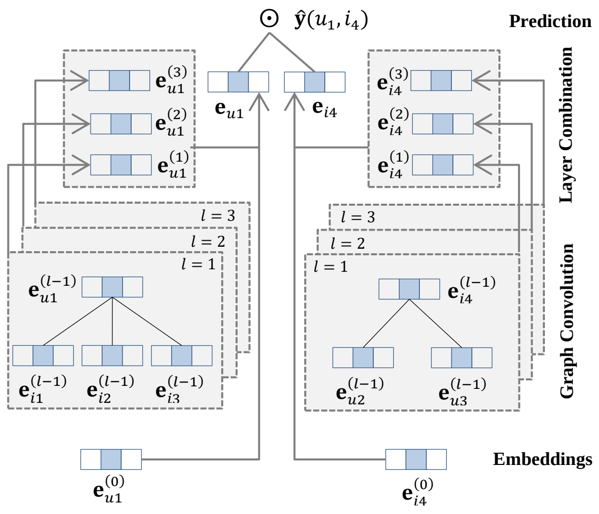


Figure 2. The process of learning representations and predicting links (between node u_1 and i_4) through the GNN model architecture.

First, in the *Embeddings* component, each user and item is represented by an initial embedding vector $\mathbf{e}_u^{(0)}$, $\mathbf{e}_i^{(0)}$, respectively. Here, $\mathbf{e}_u^{(0)} \in \mathbb{R}^d$, $\mathbf{e}_i^{(0)} \in \mathbb{R}^d$, where d is the dimension of the initial embedding vector. These embedding vectors are propagated through the interaction graph to learn and generate the final representation for each corresponding user and item ($\mathbf{e}_u$ and $\mathbf{e}_i$).

Next, in the *Graph Convolution* component, features from neighboring nodes are transmitted along edges to the active node to aggregate and create a new representation for that node. For example, consider the interaction data illustrated in *Figure 1*, the corresponding graph for the root node u_1 with an interaction depth of 3, as shown in *Figure 3*. In the first layer, the neighboring nodes of u_1 are i_1, i_2, i_3, which transmit their information and contributing to the representation of node u_1; similarly, this process continues for the second and third layers. Specifically, let $\mathbf{e}_u^{(k+1)}$ be the embedding vector of node u at the $(k+1)$-th layer. Then, $\mathbf{e}_u^{(k+1)}$ is constructed based on the following formula:

$$\mathbf{e}_u^{(k+1)} = \mathrm{AGG}(\mathbf{e}_u^{(k)}, \quad \{\mathbf{e}_i^{(k)} : \mathrm{i} \in \mathrm{N}_u\}) \qquad (1)$$

Where,

- AGG is the aggregation function (for instance, using a summation function).
- $\mathbf{e}_u^{(0)}$ and $\mathbf{e}_i^{(0)}$ represent the initial embedding vectors of user u and item i.
- $\mathbf{e}_u^{(k)}$ and $\mathbf{e}_i^{(k)}$ denote the embedding vectors of user u and item i after k layers of propagation.
- N_u represents the set of items interacted with by user u.

Each different specific algorithm in the GNN approach will have a different implementation. For example, in the NGCF[8] algorithm, $\mathbf{e}_u^{(k+1)}$ is specifically calculated as follows:

$$\mathbf{e}_u^{(k+1)} = \sigma\left(\mathbf{W}_1\mathbf{e}_u^{(k)} + \sum_{i \in \mathrm{N}_u} \frac{1}{\sqrt{|N_u||N_i|}}\left(\mathbf{W}_1\mathbf{e}_i^{(k)} + \mathbf{W}_2\left(\mathbf{e}_i^{(k)} \odot \mathbf{e}_u^{(k)}\right)\right)\right) \qquad (2)$$

$$\mathbf{e}_i^{(k+1)} = \sigma\left(\mathbf{W}_1\mathbf{e}_i^{(k)} + \sum_{u \in \mathrm{N}_i} \frac{1}{\sqrt{|N_u||N_i|}}\left(\mathbf{W}_1\mathbf{e}_u^{(k)} + \mathbf{W}_2\left(\mathbf{e}_u^{(k)} \odot \mathbf{e}_i^{(k)}\right)\right)\right) \qquad (3)$$

Where,

- $\mathbf{W}_1$, $\mathbf{W}_2$: are weight matrices.
- $\boldsymbol{\sigma}$: is the nonlinear activation function.
- $\odot$: denotes the element-wise product.

Meanwhile, the LightGCN[9] algorithm simplifies this propagation function by removing the nonlinear activation function $\boldsymbol{\sigma}$ and weight matrices $\mathbf{W}_1$, $\mathbf{W}_2$. The formula for calculating $\mathbf{e}_u^{(k+1)}$ and $\mathbf{e}_i^{(k+1)}$ in LightGCN is as follows:

$$\mathbf{e}_u^{(k+1)} = \sum_{i \in \mathrm{N}_u} \frac{1}{\sqrt{|N_u||N_i|}} \mathbf{e}_i^{(k)} \qquad (4)$$

$$\mathbf{e}_i^{(k+1)} = \sum_{u \in \mathrm{N}_i} \frac{1}{\sqrt{|N_u||N_i|}} \mathbf{e}_u^{(k)} \qquad (5)$$

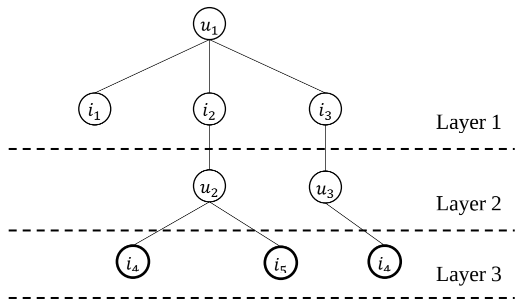


Figure 3. Information propagation from neighboring nodes to node u_1 with 3 layers of connections.

The next part of the GNN is *Layer Integration*. After k layers, we combine the embedding vectors from all layers to form a single representation for the node under consideration. Specifically:

$$\mathbf{e}_u = COMBINE(\{\mathbf{e}_u^{(k)}, k = [0..K]\} \qquad (6)$$

Here, $COMBINE$ can be a concatenation of vectors as used in the NGCF algorithm:

$$\mathbf{e}_u = \mathbf{e}_u^{(0)} \parallel \mathbf{e}_u^{(1)} \parallel . . . \parallel \mathbf{e}_u^{(K)} ; \quad \mathbf{e}_i = \mathbf{e}_i^{(0)} \parallel \mathbf{e}_i^{(1)} \parallel . . . \parallel \mathbf{e}_i^{(K)} \qquad (7)$$

Or it can be a summation, as in the LightGCN algorithm:

$$\mathbf{e}_u = \textstyle\sum_{k=0}^{K} \alpha_k \mathbf{e}_u^{(k)} \; ; \; \mathbf{e}_i = \textstyle\sum_{k=0}^{K} \alpha_k \mathbf{e}_i^{(k)} \qquad (8)$$

With $\alpha_k \geq 0$ as a hyperparameter, it represents the importance of the embedding vector at the k-th layer in contributing to the final embedding vector. In the LightGCN algorithm, the authors set $\alpha_k = \frac{1}{K+1}$.

Finally, we have the *Link Prediction* component. Once the embeddings for each node have been established, the predicted link $\hat{y}_{ui}$ between two nodes u and i, can be simply computed using the inner product:

$$\hat{y}_{ui} = \mathbf{e}_u^T \mathbf{e}_i \qquad (9)$$

The higher this result, the stronger the link between u and i. Recommending items to user u is done by calculating the link values between u and various candidate items i. These links are then sorted, and the *top-K* strongest links are selected for recommendation.

2.4. Model Training

Using a loss function to optimize parameters in the model. One popular loss function is BPR (*Bayesian Personalized Ranking*)[10] designed to optimize the ranking of recommendations:

$$L = -\sum_{(u,i,j)\in D} ln\sigma\left(\hat{y}_{ui} - \hat{y}_{uj}\right) + \lambda\|\Theta\|^2 \qquad (10)$$

In which,

- D: is the set of triples (u,i,j), where (u,i) is the observed interaction of user u with item i, while (u,j) is the interaction of user u with item j that is not observed.
- $\hat{y}_{ui}$ and $\hat{y}_{uj}$: are the predicted values of the strength of the connections between the pairs (user, item): (u,i) and (u,j), computed by the model.
- σ: is the *sigmoid* function.
- Θ: are the parameters of the model. In the LightGCN algorithm, these parameters are the embedding vectors representing each user and item at layer $k = 0$.
- λ: is the regularization coefficient.

The BPR loss function encourages the model to predict links so that items the users have interacted with (e.g., viewed or rated highly) are ranked higher than items they have not interacted or disliked. Through the above calculation steps, the model will adjust the parameters to minimize the BPR loss function, thereby improving the personalized ranking.

3. TRAVEL RECOMMENDATION APPLICATION

3.1 System Architecture

Our system is built using a layered architecture, specifically consisting of four layers: the *Database layer*, the *RS (Recommendation System) layer*, the *Service layer*, and the *Application layer*, as illustrated in *Figure 4.*

The *Database layer* provides data for the system, comprising two types of data: (i) Rating data for tourist attractions and (ii) Side information, which includes detailed information about the attractions (such as category, address, images). In particular, the Rating data shows the interest and satisfaction level (from 1-5 stars) of tourists with the attractions. This is the basis for building a prediction model, which helps in generating suitable recommendations. These data will be periodically extracted (e.g., every 3 days) from the application database, then they will be stored (using *Google Cloud Storage* service) to serve the recommendation system.

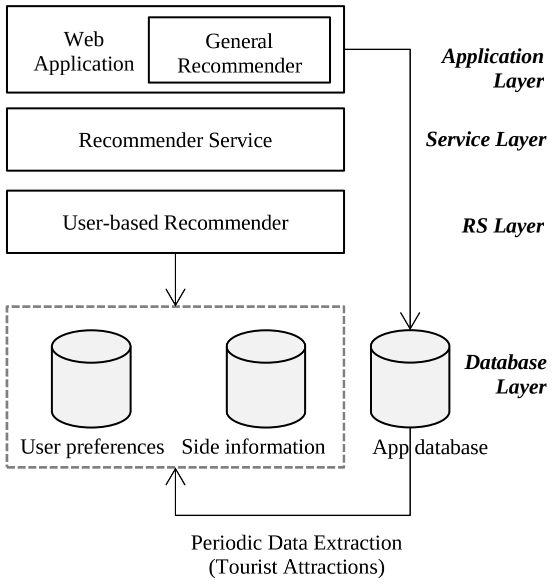


Figure 4. Architecture for the Travel Recommendation Application.

The *RS layer* contains the *Tourist Attraction Recommendation Module*. This layer is responsible for building the prediction model based on the collected rating data and generating a personalized list of recommendations for each user based on their rating history. We will provide detailed information on the implementation of this module in the following section.

The *Service layer* creates Web APIs to expose the functionalities of the recommendation system to various applications, allowing them to call these APIs and present the results. The primary purpose of this layer is to enable different types of applications across various platforms and technologies to utilize the recommendation system. We implement this module using *Flask*[i] and deploy it on *Google Cloud*[ii].

Finally, the *Application layer* includes applications that interact with end-users and present the results. Since this layer is built upon the *Service layer*, these applications can be deployed on various platforms such as Mobile, Desktop, and Web. In this study, we implement a Web application for testing, using *Wordpress*—a powerful and widely-used Content Management

[i] Flask, một framework phát triển ứng dụng web trên Python, https://flask.palletsprojects.com/en/3.0.x/
[ii] Dịch vụ điện toán đám mây của Google, https://cloud.google.com/

System (CMS). The *General Recommender* module within the application provides recommendations for users who are not logged into the system or do not have profiles on the system. This module recommends highly-rated attractions to users based on simple statistical methods. In the future, this component could be enhanced to offer recommendations based on seasonality, regions, and other factors.

3.2 Recommendation Process & Experiments

The *User-based Recommender* module in the *RS layer* provides personalized attraction recommendations for a specific user based on their past rating history. In which:

- Input: User ID.
- Output: *Top-K* tourist attractions the user may be interested in.

We propose a general process for generating recommendations, as illustrated in *Figure 5*. Specifically, it includes main components such as: *Data, Filter, Model Training, Links Prediction & Ranking*.

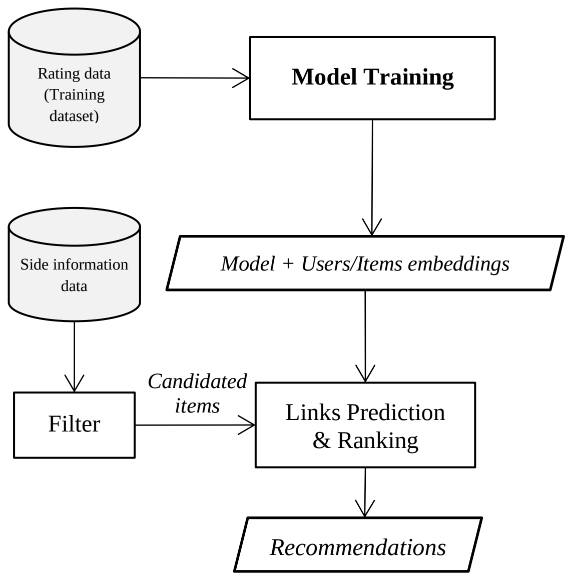
The *Data* component includes two types: Rating Data and Side Information Data. The Rating Data includes user ratings for various attractions and is used to train the prediction model. Meanwhile, the Side Information Data is used to filter candidate attractions.

The *Filter* module is designed to select candidate attractions from all available attractions in the system based on user-specific criteria. For example, it can filter attractions that belong to categories such as spiritual tourism or adventure tourism. However, within the scope of this study, the module is implemented in a simplified manner by including all attractions in the system, except for those the user has already interacted with, as input for the recommendation process.

The *Model Training* module is responsible for training the prediction model. The output includes the trained model and the final embeddings for each user and attraction. In this study, we utilize the LightGCN[9] algorithm for model training and prediction.

Links Prediction & Ranking: This module predicts the links between users and attractions, followed by ranking them. Specifically, for each attraction, the system predicts the strength of the connection between the user and that attraction. The attractions are then sorted in descending order of connection strength, and the *top-K* attractions with the highest connection strengths are returned as recommendations.

One noteworthy point is that we do not store the trained model. Instead, for link prediction, we only save the final embeddings of each user and attraction (Users/Items embeddings) and compute the inner product between these vectors. This approach simplifies server processing since it eliminates the need to reference libraries for loading the model.



Rating data (Training dataset)
Model Training
Side information data
Model + Users/Items embeddings
Candidated items
Filter
Links Prediction & Ranking
Recommendations

Figure 5. Recommendation Process.

To evaluate the effectiveness of the prediction model, we conducted experiments on three datasets: MovieLens[iii], Yelp2018[9], and TripAdvisor[11]. Both Yelp2018 and TripAdvisor are datasets about tourism domain. Statistics on the size of these datasets are described *Table 1*. Specifically, we used the MovieLens dataset with 100,000 ratings, the Yelp2018 dataset with 500,000 rating[iv], and the TripAdvisor dataset with 13,697 ratings for our experiments. The metrics recall@20, ndcg@20, and precision@20 were used to assess the model's performance[12,13]. The graph learning algorithm used in these experiments is LightGCN. The initial embedding size is set to 64. The model was optimized using the *Adam*[14] optimizer with a learning rate of 0.005. The number of propagation layers in the graph was set to 3. Each batch size was 1,024, and the model was trained for 100 epochs. We

[iii] https://grouplens.org/datasets/movielens/
[iv] We reduce the data size to fit computational resources.

utilized *Google Colab*[v] to train the model and to learn the embeddings for users and attractions.

Table 1. Experimental Datasets.

Datasets	Ratings #	Users #	Items #
MovieLens	100.000	943	1.682
Yelp2018	500.000	8.154	36.837
TripAdvisor	13.697	2.371	2.269

The training process to optimize the model is illustrated through the *training-loss* curve in *Figure 6-8*, corresponding to the MovieLens, Yelp2018 and TripAdvisor data sets.

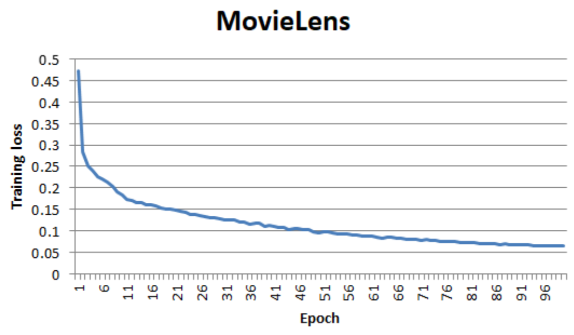


Figure 6. Training-loss curve on the MovieLens dataset.

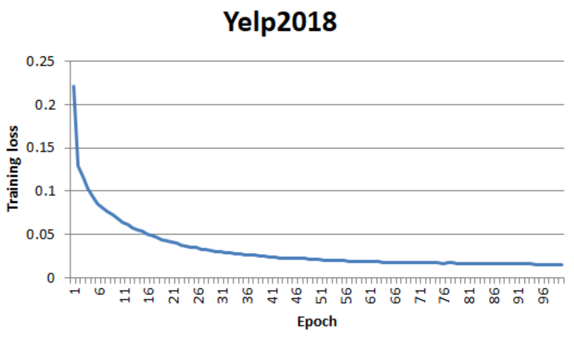


Figure 7. Training-loss curve on the Yelp2018 dataset.

[v] A cloud service provided by Google, allowing writing and execution of Python code through a web browser, https://colab.research.google.com/.

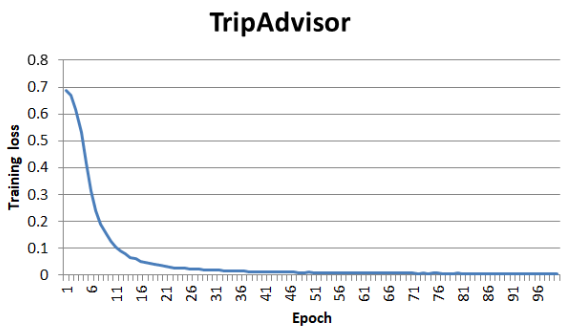


Figure 8. Training-loss curve on the TripAdvisor dataset.

Experimental results of the LightGCN model on the datasets are shown in *Table 2*.

Table 2. Experimental results of the LightGCN

Datasets	Metrics		
	Recall	**Precision**	**NDCG**
MovieLens	0,212395	0,386744	0,439850
Yelp2018	0,082281	0,052545	0,083219
TripAdvisor	0,088338	0,005332	0,039634

Experimental results show that the LightGCN algorithm works effectively, shown through Recall, Precision, and NDCG metrics. In particular, with the MovieLens data set, the best results were obtained with Recall, Precision and NDCG values of 21.2%, 38.7% and 44%, respectively. For the other two datasets, the results were similar: for the Yelp2018 dataset, the results were 8.2%, 5.3%, and 8.3%; and for the TripAdvisor dataset, the results were 8.8%, 0.5%, and 4%. LightGCN was evaluated to be more effective compared to the NGCF algorithm and other algorithms[8,9]. In this study, we do not aim to compare the LightGCN algorithm with other algorithms, instead, we test the algorithm with different data sets, to ensure that it performs well before integrating it into the travel recommendation application.

3.3 The Web Application

The web application is built on the *WordPress* platform, providing a useful and convenient tool for users to explore attractions. The application serves two main types of users: administrators and tourists. In particular, the administrator has main functions such as: Managing attractions (posting, editing articles...) and Managing users; Meanwhile, tourists can search for attractions by province/city, leave reviews about their experiences with attractions, and view recommendations based on their rating history. If the user is not logged in, the system suggests attractions with high average ratings.

As described in the previous section, this Web application will connect to the Web API to fetch recommended attractions for users. Some images of the application are depicted in *Figure 9-11*.

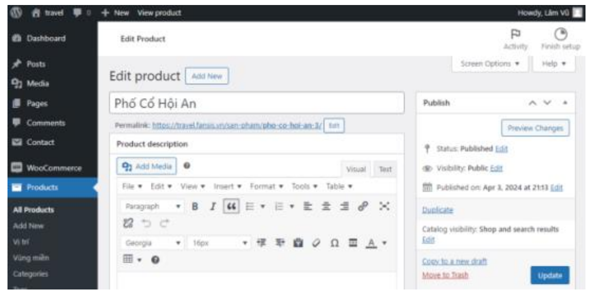


Figure 9. The function of Posting attractions.

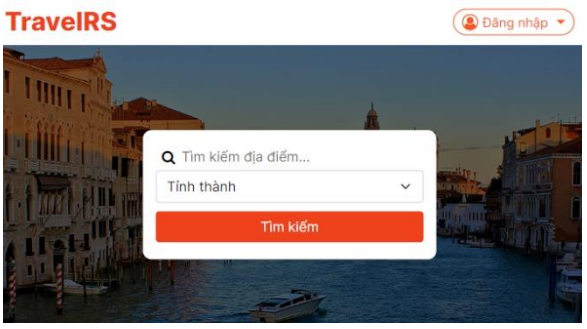


Figure 10. The function of Searching attractions.

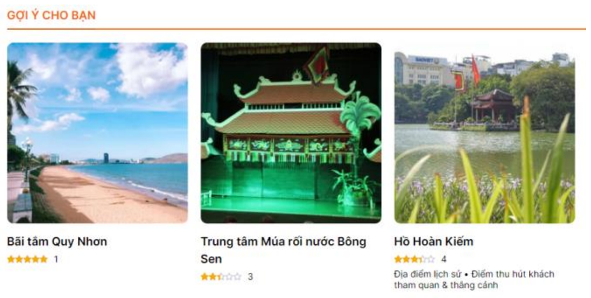

GỢI Ý CHO BẠN
Bãi tắm Quy Nhơn
1
Trung tâm Múa rối nước Bông Sen
3
Hồ Hoàn Kiếm
4
Địa điểm lịch sử • Điểm thu hút khách tham quan & thắng cảnh

Figure 11. The function of Suggesting attractions.

To validate the recommendation functionality of the web application, we simulated creating 3 users: u_1, u_2 and u_3.. These users rated 5 tourist attractions (encoded as i_1, $i_2 \ldots i_5$) out of a total of 50 attractions in the system. The rating data is illustrated in *Figure 1*. We plotted graphs (*Figures 3, 12*, and *13*) with the root nodes corresponding to each user to find attractions the user might be interested in through the connections in the graph.

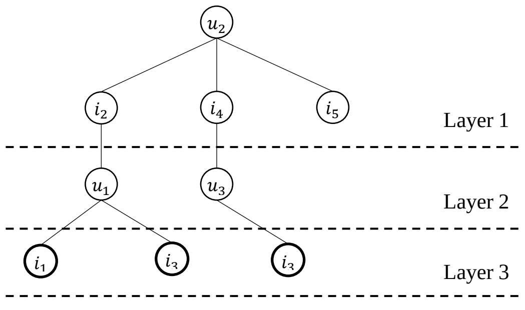


Figure 12. Graph with 3 connection layers for root vertex u_2.

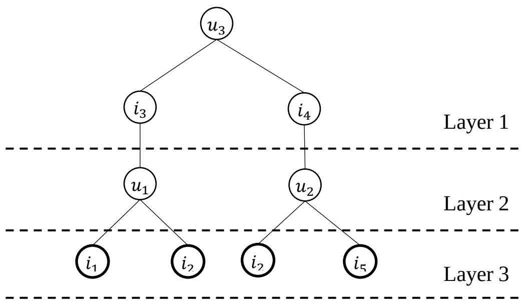


Figure 13. Graph with 3 connection layers for root vertex u_3.

By visually analyzing these small graphs, the attractions that should be suggested for each user are identified and listed in *Table 3* below. These are the leaf nodes on the graph, connected to the root nodes (users) through intermediary nodes.

Table 3. Attractions that should be suggested to each user, derived from the graph analysis.

Users	**Attractions should be suggested**
u_1	i_4, i_5
u_2	i_3, i_1
u_3	i_2, i_1, i_5

Following the recommendation process described earlier, we learned the representations for each user and attraction to calculate the strength of the connections between them. The predicted connection strengths are provided in *Table 4* below. Based on these results, we observe that the constructed model accurately predicts the attractions that users might be interested in. Specifically:

- For u_1, the two attractions with the highest positive connection strengths are i_4 and i_5.
- For u_2, the two attractions with the highest positive connection strengths are i_3 and i_1.

- For u_3, the two attractions with the highest positive connection strengths are i_2, i_1 and i_5.

Table 4. Predicted Connection Strengths between Users and Attractions**.**

User	Attractions	Predicted Links
u_1	i_4	4,399
u_1	i_5	3,109
u_1	$i_{17}, i_{12}, i_7, i_{34}, i_{36}, i_{39}, i_{46}, i_{10}$	-1,567, -1,694, -1,699, -1,705, -1.713, -1,718, -1,725, -1,726
u_2	i_3	4,378
u_2	i_1	3,184
u_2	$i_{17}, i_7, i_{12}, i_{39}, i_{33}, i_{36}, i_{14}, i_{41}$	-1,640, -1,747, -1,752, -1,764, -1,773, -1,776, -1,794, -1,797
u_3	i_2	3,669
u_3	i_1	2,735
u_3	i_5	2,697
u_3	$i_{17}, i_{34}, i_7, i_{40}, i_{41}, i_{46}, i_{11}$	-1,369, -1,415, -1,441, -1,442, -1,459, -1,466, -1,471

Thus, the web application has demonstrated a travel recommendation system according to the proposed process described earlier. Accordingly, for users who already have a profile on the system, the application will suggest attractions that they may be interested in based on their review history. Here, we only created a very limited number of users and attractions to illustrate the system's workflow. Building a larger dataset can be undertaken once the system is deployed in a real-world operational environment.

4. RELATED WORKS

Recommendation systems are widely applied across various domains, including online shopping, news reading, travel, music, movie, and social networking. These systems act as information filters, suggesting products or services that users might be interested in, based on their behavior history or the matching characteristics between products/services and user profiles.

Among the many approaches to building recommendation systems, Graph Learning is currently attracting significant attention. Graph learning applies machine learning techniques to graph-structured data and is rapidly evolving[4]. In recent years, several graph-based recommendation models have been proposed and studied[5,6]. The application of graph learning to recommendation systems can be categorized into three main approaches: (i) Random Walk Based Approaches[15,16]. These methods explore the graph by simulating random walks to generate sequences of nodes, which are then used for making recommendations. While effective in capturing the connectivity within the graph, these approaches are heuristic-based and lack model parameters for optimization. (ii) Graph Embedding Based Approaches[17-20]. This method is efficient for analyzing complex relationships embedded in the graph. (iii) Graph Neural Network Based Approaches[21-25]. GNNs apply neural network techniques to graph data. They iteratively aggregate information from a node's neighbors to learn representations. GNN-based models have become increasingly popular due to their ability to effectively model the intricate dependencies in graph data, leading to significant advancements in recommendation performance. In Vietnam, a group of authors proposed using GNN to build session-based recommendation models[26]. Their model was evaluated using common metrics in recommendation systems, such as Recall@20 and Mean Reciprocal Rank (MRR)@20, demonstrating the effectiveness of GNNs in capturing user preferences and making accurate recommendations.

5. CONCLUSION

In this paper, we have presented the application of graph learning in a travel recommendation system. Specifically, we have illustrated the use of the LightGCN algorithm in the Graph Neural Network approach to model and process data in the form of a bipartite graph. This method allows the recommendation system to exploit the complex structure and hidden relationships between users and attractions, thereby providing accurate suggestions to users. At the same time, we also tested the algorithm on three different data sets to evaluate their effectiveness. Besides, we have proposed the travel recommendation system application architecture. This architecture is general, based on that, we can build systems with different recommendation algorithms, with different technologies.

In the future, we plan to continue to improve and expand the system. Specifically, some future research directions include: (i) Iintegrating contextual recommendation. We aim to enhance the recommendation system by incorporating contextual information. For example, the system could suggest tourist attractions based on additional context such as the time to visit or the type of companions (family, friends, solo travel). (ii) Exploration of advanced machine learning techniques. We plan to investigate and apply more advanced machine learning techniques to optimize the system. (iii) Enhancement of the web application. We intend to further develop the web application by adding more diverse information and functionalities. This could include features like tour booking, itinerary planning, recommendations for accommodations (hotels, hostels) and dining options (restaurants, cafes).

In short, building a travel recommendation system is important and necessary, especially in the context of the rapid growth of the tourism industry in Vietnam. We hope that these research results will be an important stepping stone to develop more intelligent travel recommendation systems in the future.

REFERENCES

1. L. Quang-Hung, V. Son-Lam, L. Anh-Cuong. A Comparative Analysis of Various Approaches for Incorporating Contextual Information into Recommender Systems, *Journal of Computer Science*, **2022**, *18(3)*, 187-203.
2. V. Son-Lam, L. Quang-Hung. A Deep Learning Based Approach for Context-Aware Multi-Criteria Recommender Systems, *Computer Systems Science and Engineering*, **2022**, *44(1)*, 471–483.
3. L. Quang-Hung, V. Son-Lam, N. Thi-Kim-Phuong, L. Thi-Xinh. A state-of-the-art survey on context-aware recommender systems and applications, *International Journal of Knowledge and Systems Science (IJKSS)*, **2021**, *12(3)*, 1–20.
4. Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, S. Y. Philip. A comprehensive survey on graph neural networks, *IEEE transactions on neural networks and learning systems*, **2020**, *32(1)*, 4–24.
5. Q. Guo, F. Zhuang, C. Qin, H. Zhu, X. Xie, H. Xiong, Q. He. A survey on knowledge graph-based recommender systems, *IEEE Transactions on Knowledge and Data Engineering*, **2020**, *34(8)*, 3549-3568.
6. S. Wang, L. Hu, Y. Wang, X. He, Q. Z. Sheng, M. A. Orgun, P. S. Yu. *Graph learning based recommender systems: A review*, Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI-21), Montreal, 2021.
7. S. Wu, F. Sun, W. Zhang, X. Xie, B. Cui. Graph neural networks in recommender systems: a survey, *ACM Computing Surveys*, **2022**, 55(5), 1-37.
8. X. Wang, X. He, M. Wang, F. Feng, T. S. Chua. *Neural Graph Collaborative Filtering*, In Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval, 2019.
9. X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, M. Wang. *LightGCN: Simplifying and powering graph convolution network for recommendation*, In Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, 2020.
10. Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, Lars Schmidt-Thieme. *BPR: Bayesian Personalized Ranking from Implicit Feedback*, Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, 2009.
11. Y. Zheng, B. Mobasher, R. Burke. *Context recommendation using multi-label classification*, In 2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT), 2014.
12. Y. M. Tamm, R. Damdinov, A. Vasilev. *Quality metrics in recommender systems: Do we calculate metrics consistently?*, In Proceedings of the 15th ACM Conference on Recommender Systems, 2021.
13. Vu Son Lam, Le Quang Hung, Nguyen Van Vinh. *Đánh giá hệ gợi ý: Khảo sát và thực nghiệm*, Kỷ yếu hội thảo Quốc gia lần thứ XXIII "Một số vấn đề chọn lọc của Công nghệ thông tin và Truyền thông", 2020.
14. Diederik P. Kingma, Jimmy Ba. *Adam: A Method for Stochastic Optimization*, International Conference on Learning Representations, 2014.
15. Z. Jiang, H. Liu, B. Fu, Z. Wu, T. Zhang. *Recommendation in heterogeneous information networks based on generalized random walk model and bayesian personalized ranking*, In Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, 2018.
16. H. Bagci, P. Karagoz. *Context-aware friend recommendation for location based social networks using random walk*, In Proceedings of the 25th international conference companion on world wide web, 2016.
17. Z. Wang, H. Liu, Y. Du, Z. Wu, X. Zhang. *Unified embedding model over heterogeneous information network for personalized recommendation*, In Proceedings of the 28th international joint conference on artificial intelligence, 2019.
18. Y. Cen, X. Zou, J. Zhang, H. Yang, J. Zhou, J. Tang. *Representation learning for attributed multiplex heterogeneous network*, In Proceedings

of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019.

19. B. Hu, C. Shi, W. X. Zhao, P. S. Yu. *Leveraging meta-path based context for top-n recommendation with a neural co-attention model*, In Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining, 2018.

20. Y. Deng. Recommender Systems Based on Graph Embedding Techniques: A Review, *IEEE Access*, **2022**, 10, 51587-51633.

21. R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, J. Leskovec. *Graph convolutional neural networks for web-scale recommender systems*, In Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining, 2018.

22. W. Fan, Y. Ma, Q. Li, Y. He, E. Zhao, J. Tang, D. Yin. *Graph neural networks for social recommendation*, In The world wide web conference, 2019.

23. Z. Cui, Z. Li, S. Wu, X. Y. Zhang, L. Wang. *Dressing as a whole: Outfit compatibility learning based on node-wise graph neural networks*, In The World Wide Web Conference, 2019.

24. C. Gao, X. Wang, X. He, Y. Li. *Graph neural networks for recommender system*, In Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining, 2022.

25. D.H. Tran, Q.Z. Sheng, W.E. Zhang, et al. HeteGraph: graph learning in recommender systems via graph convolutional networks, *Neural Comput & Applic*, **2023**, 35, 13047–13063.

26. T. K. Nguyen, T. A. Nguyen, T. N. Mai, H. A. Nguyen, V. A. Nguyen. Hệ gợi ý mua sắm dựa theo phiên làm việc với mô hình mạng học sâu đồ thị, *Các công trình nghiên cứu, phát triển và ứng dụng Công nghệ Thông tin và Truyền thông*, **2022**, 2022(2), 72-81.