International Journal of Computing and Artificial Intelligence

E-ISSN: 2707-658X
P-ISSN: 2707-6571
IJCAI 2023; 4(2): 06-18
Received: 06-05-2023
Accepted: 10-06-2023

Sravani Prakki
Department of Computer Science, California State University, East Bay Hayward, California, United States

Moayed Daneshyari
Computer Science Department California State University, East Bay Hayward, California, United States

Corresponding Author:
Sravani Prakki
Computer Science Department California State University, East Bay Hayward, California, United States

Travel recommendation system using graph neural networks

Sravani Prakki and Moayed Daneshyari

DOI: https://doi.org/10.33545/27076571.2023.v4.i2a.66

Abstract
There are many applications and uses of recommendation systems. For any recommendation system, the user-to-item interactions are important which can also be seen as graphs. We are focusing on travel or place recommendations. There are very few works on travel or trip recommendation systems using Graph Neural Networks (GNN) leveraging user-to- item interactions. In this work, we have implemented travel or place recommendations using the LightGCN model and compared it with the implementation of the traditional non-graph-based collaborative filtering Matrix Factorization (MF) approach. We have then shown that the LightGCN model performs better for travel or place recommendations than the Matrix Factorization approach. We achieved very good results [203% increase in precision, a 167% increase in recall, and a 98.6% NDCG increase in metrics] using a graph based LightGCN model for travel or place recommendations compared to the Matrix Factorization approach.

Keywords: Graph neural networks, LightGCN, neural graph collaborative filtering, collaborative filtering, embedding propagation

Introduction

In this section, we discuss the motivation for graph-based travel recommendations, our project goal, and the description of the article outline of the below sections.

Many of us enjoy planning vacations or trips. However, according to Travel Daily News - ”66% of travelers say that planning a Trip is stressful and difficult”. Travel Agent Central says - “89 Percent Say Travel Is Enjoyable but Stressful and Difficult to Plan”. Omnibus survey says “Deciding on a location (50%)”

This could be because, with the advancement of big data, multiple internet resources are available. Finding vacation destinations or travel packages among these huge travel-related online resources is very hard.

Also, sometimes we might not be fully aware of all our choices to make fast decisions. Many travel service agencies are available in the market to guide us with trip recommendations.

However, it is also hard for them to personalize based on individual customer preferences. This forms the primary motivation for any personalized travel recommendation systems. They reduce this huge information overload to only a small subset containing the choices or information the user or customer is most likely interested in.

As recommendation systems are a broader area, we are focusing on travel-based recommendation models. In general, travel-based recommendation models are different from traditional recommendation systems. This is because travel products are not commonly browsed or purchased, unlike general products like music, movies, etc. Moreover, many factors such as price, destinations, etc., are also considered when purchasing travel products or making any travel plans.

There are many travel recommendation systems and techniques available in the market. In general, recommendation systems use explicit or implicit feedback (could be a rating or feedback provided by the user to an item or travel product, or a place) or any information related to users' past experiences or interactions to suggest future recommendations.

The interactions between users and items (travel products or places or destinations) could also be viewed as a graph. Numerous problems in the actual world can be represented using graphs. Any complex or simple data could be represented well, using graphs.

They also consider the underlying relationship between the entities. Though there are many traditional-based travel recommendation models such as collaborative filtering, content-based filtering, convolution neural networks, matrix factorization, etc., they consider either user information or the travel product or item information to suggest recommendations.

Whereas in graph-based recommendation systems, both the user-to-item interactions are considered for recommendations. Also, there are very few works on graph-based recommendation systems for travel or place recommendation. This forms the motivation for our project.

Our objective is to develop a travel or place recommendation solution that leverages the graph structure considering the underlying relationship between users and the items or place data relationships for making recommendations.

In Section II, we have mentioned related work. In Section III the methodology of the proposed solution is described. The performance evaluation and results are described in section IV and section V respectively. The summary of the project and future works are mentioned in sections VI and VII followed by acknowledgments and references.

Related Work

In this section, we discuss the related works. We have broadly classified it into three categories non-graph-based travel recommendation systems, graph-based travel recommendation systems, and general graph-based recommendation systems.

A. Non-graph-based travel recommendation systems

Below we mentioned some of the important works in non-graph-based travel recommendation systems.

Analysis of tourism recommendation systems works since 2008, focusing on content-based suggestions, has been mentioned in the conference survey article proposed by Bentaleb *et al* (2021) [2]. The author discusses several methods for making traditional recommendations, such as content-based methods, collaborative filtering, and hybrid methods.

Wang *et al*. (2020) [12] proposed a tourism collaborative system using deep learning methodology. They have used techniques such as Convolutional Neural Networks (CNN), Deep Neural Networks (DNN) to process user reviews and tourism service items.

Kbaier *et al* (2017) [13] proposed a personalized hybrid recommendation system for tourists. They have used techniques and algorithms such as KNN Algorithm using Collaborative filtering, content-based techniques, and decision tree for demographic filtering.

The above approaches use traditional methods for suggesting travel-based recommendations. Unlike our approach, these methods do not leverage graph structure.

B. Graph-based travel recommendation systems

Below we mentioned some of the important works in graph-based travel recommendation systems.

Xin *et al*. (2021) [11] proposed an Out-of-Town Recommendation system. They have used techniques such as Graph Neural Networks, Neural Topic Model (NTM), and Matrix Factorization. In this work, they are suggesting recommendations based on the user's travel behavior.

Chen *et al* (2021) [4] proposed a Multi-view Graph Attention Network for the Travel Recommendation system. In this work, they are predicting the click probability of a user.

The above works have not considered rating attribute, unlike our approach. As there are very few works in travel or place recommendation systems using Graph-based models, in our work we propose an implementation of a travel recommendation system using the LightGCN approach.

In our work, we are implementing the LightGCN graph-based model proposed by He *et al* (2020) [1] for travel or place recommendations. This model uses travel or trip-related datasets (Yelp reviews, Indonesia tourism destination dataset) with users and places as nodes and user ratings as the weight to edges between the nodes. It uses rating as a feedback metric to suggest travel places or destinations or business places to users.

C. General Graph-based recommendation systems

Here we discuss general graph-based recommendation systems (not related to travel) A graph convolution matrix completion for recommender systems was proposed by Berg *et al*. (2017) [10] considering the link prediction on graphs using user data, movie ratings, and reviews (movies dataset). On the bipartite interaction graph, they have considered a graph auto-encoder architecture using differentiable message passing.

Wang *et al* (2019) [14] proposed the Neural Graph Collaborative filtering (NGCF) model. This work discusses the application of GCN (Graph neural network) concepts for recommendations. They use GCN features such as Feature Transformation, Neighborhood Aggregation, and Non-linearity activation function for calculating the user and item embeddings for predicting recommendations.

Zhang M, and Chen Y proposed Inductive matrix completion (IGMC) based on graph neural networks (2019) [15]. This work addresses the issues of the Rianne van den Berg *et al* [10] paper for Matrix completion. Here the generalization of the learned embeddings is extended to unseen data. They have considered movies dataset.

Methodology

In this section, we discuss the LightGCN model, its architecture, and loss and optimization functions.

The LightGCN model is proposed by He, Xiangnan, *et al* [1]. In this paper, a thorough ablation analysis has been performed on Graph Convolution Neural Network (GCN) model Neural Graph collaborative filtering (NGCF) model proposed for recommendations.

In the NGCF model for graph-based recommendations (a variant of GCN) the concepts from the GCN such as feature transformation and linearity activation function are directly taken into consideration without much understanding of why these features were considered for recommendations. In general, they are mainly useful for node classification-related tasks for graphs. The GCN for recommendations has experimented in the NGCF [14] model. This work is taken as the basis for the LightGCN model.

From the analysis of ablation studies performed on the earlier works for recommendation systems in the Graph Neural Networks (GNN) NGCF model, it has been found that two operations, feature transformation, and nonlinearity activation function will not contribute much to the performance of the recommendation system. Moreover, it has been observed that adding these functionalities will

reduce performance and will make the training of the system more complex.

Removing these functions has shown significant improvement in the performance of the recommendation model using Graph Neural Networks. This model is called LightGCN. The main intuition behind removing these functionalities is that we merely consider the user and item as identification factors (node IDs) without any rich features for a user-item interaction graph. Performing non-linear feature transformation for several message-passing layers will not add any benefit to the performance of the system.

A. LightGCN Architecture

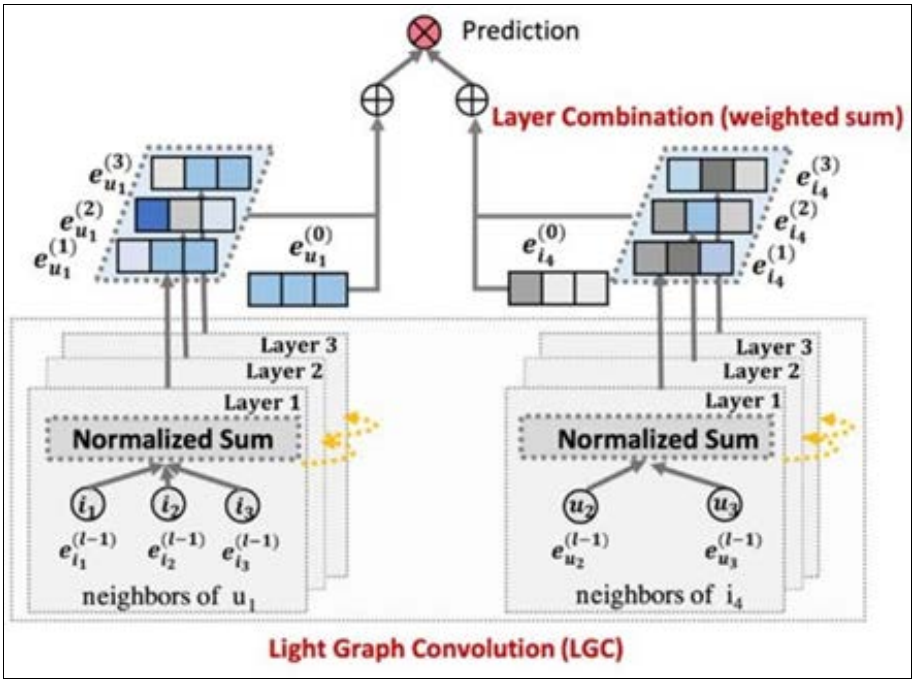

Fig 1: LightGCN architecture

The LightGCN model focuses on "neighborhood aggregation". This model leverages the salient structure of the graph for updating the node embeddings. In general user-item interaction, graphs are used for recommendations in graph models.

For providing recommendations of items or products to the user in the LightGCN model the initial user and item embeddings are randomly initialized. In this model, the initial user and item embeddings are the only learnable parameters. These initial embeddings are linearly propagated on the user-item interaction graph. This process is repeated for several layers like any neural network model. Mean of weighted summation of embeddings at every layer is calculated to obtain the final embeddings which would be used for loss metrics and backpropagation to update the initial embeddings (weights). These updated initial embeddings of users and items are used to calculate the similarity (dot product of initial user and item embeddings) to suggest items to a user.

B. Mathematical equations

Neighborhood aggregation (updating of items and user's embeddings)

$$\mathbf{e}_u^{(0)} \quad \mathbf{e}_i^{(0)} \quad \text{(Initial user, item embeddings)} \tag{1}$$

Graph convolution operation (neighborhood aggregation)

$$\mathbf{e}_u^{(k+1)} = \sum_{i \in \mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_i|}} \mathbf{e}_i^{(k)},$$
$$\mathbf{e}_i^{(k+1)} = \sum_{u \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_i|}\sqrt{|\mathcal{N}_u|}} \mathbf{e}_u^{(k)}. \tag{2}$$

Nu refers to all the items that interacted with a user u (degree of the user) Ni refers to all the users that interacted with an item I (degree of the item)

Final embeddings (average weighted summation of embeddings at each layer)

$$\mathbf{e}_u = \sum_{k=0}^{K} \alpha_k \mathbf{e}_u^{(k)}; \quad \mathbf{e}_i = \sum_{k=0}^{K} \alpha_k \mathbf{e}_i^{(k)} \tag{3}$$

Prediction (similarity between users and items) In matrix form,

$$\mathbf{A} = \begin{pmatrix} \mathbf{0} & \mathbf{R} \\ \mathbf{R}^T & \mathbf{0} \end{pmatrix} \tag{4}$$

This is the user-to-item interactions i.e., the adjacency matrix. Here 'R' refers to the rating provided by the user to an item. This adjacency matrix represents the structure of the graph. This is used for message passing between the user and item nodes in the user-item interaction graph.

For the graph convolution operation (neighborhood aggregation) for learning the user and item embeddings, the matrix form of equations is as below:

$$\mathbf{E}^{(k+1)} = (\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}})\mathbf{E}^{(k)} \quad (5)$$

Upon calculating the embeddings of the user and item at each layer, a weighted summation of these embeddings is taken for generating the final representations. The matrix equations are as follows:

$$\begin{aligned}\mathbf{E} &= \alpha_0\mathbf{E}^{(0)} + \alpha_1\mathbf{E}^{(1)} + \alpha_2\mathbf{E}^{(2)} + \ldots + \alpha_K\mathbf{E}^{(K)} \\ &= \alpha_0\mathbf{E}^{(0)} + \alpha_1\tilde{\mathbf{A}}\mathbf{E}^{(0)} + \alpha_2\tilde{\mathbf{A}}^2\mathbf{E}^{(0)} + \ldots + \alpha_K\tilde{\mathbf{A}}^K\mathbf{E}^{(0)}\end{aligned} \quad (6)$$

The matrix equation for finding the similarity between the initial weighted (trained) embeddings upon performing the loss function and optimization on the above final embeddings is as below:

$$\tilde{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}} \quad (7)$$

C. Loss Functions

In this proposed work we have used Bayesian Personalized Ranking (BPR) [20] Loss for calculating the losses for LightGCN implementation.

$$L_{BPR} = -\sum_{u=1}^{M}\sum_{i\in N_u}\sum_{j\notin N_u}\ln\sigma(\hat{y}_{ui} - \hat{y}_{uj}) + \lambda||E^{(0)}||^2$$

$\hat{y}_{ui}$: predicted score of a positive sample
$\hat{y}_{uj}$: predicted score of a negative sample
λ: hyperparameter which controls the L2 regularization strength (8)

In the Bayesian Personalized Ranking loss function, we have calculated positive and negative samples (of places or items) for a user. For this, we considered a tuple (i, j, k), where for everyuser 'i' a positive item (place or destination) 'j' is considered, and a negative item (place or destination) 'k' is considered.
Adam optimizer is used for LightGCN. We have implemented this LightGCN model using two datasets Yelp Reviews and Indonesia Tourism Destination Dataset for travel recommendations.
The description and details of the datasets and implementationare mentioned below in section IV.

Performance evaluation

In this section, we discuss the implementation of LightGCN on travel datasets. We briefly describe the comparison baselines Matrix Factorization and Random baseline. We then compare the LightGCN approach with Matrix Factorization and Randombaseline. We have also described the dataset and its features. We have mentioned the frameworks used. We then explained the evaluation metrics used and the data processing details followed by hyperparameter tuning and loss graphs.

A. Comparision Baselines

Matrix Factorization: We have implemented a traditional recommendation approach of matrix factorization for travel or place recommendations. We then compared it with the graph- based approach to observe the performance.
Matrix Factorization [16, 17] is a popular collaborativefiltering technique. As the name suggests the rating or the feedback matrix is factorized into user and item representations.

Rating Matrix is a matrix with users as rows and Items as columns. In general, the user's respective rating of a particular item or any travel product or destination (travel product or destination) is stored in the corresponding cell.
In this methodology, we initially generated random user and item embeddings. These initial embeddings are further trained via loss calculations and optimizations. These final trained embeddings are used for calculating the rating prediction score or for finding the similarity between the users and the items.
For the data processing, we have considered both the yelp reviews and the Indonesia tourism dataset. Like LightGCN dataprocessing, we have only included the ratings >=4 in the range of 0-5. We have calculated unique mappings of users and items(places) and generated initial random embeddings. These embeddings are further trained, and the losses are optimized togenerate final user and place embeddings.
We have used the best hyperparameters for MF for both Yelp Reviews and the Indonesia tourism datasets as per [22, 23]. The parameters are as follows:

Table 1: Matrix factorization hyperparameters

Iterations	800
Batch size	1024
Learning rate (LR)	1
Lambda	0.0002

We have used the Mean square error loss function [19] for matrix factorization.

$$E(U,V) = \frac{1}{N}\sum_{(i,j):r_{ij}=1}(y_{ij} - u_i \cdot v_j)^2 \quad (9)$$

In the mean square error loss function, we calculated the mean of squared differences of the actual rating values (ground truth)and the predicted rating score (normalized dot product of the trained user and item embeddings).
We have used Gradient descent optimization for Matrix factorization.

Random Baseline: In this method, we are populating the rating matrix with random values. Then we calculate the metrics to observe its performance.
The LightGCN model results are compared with the Matrix Factorization approach and random baseline method. The comparison results and the performance evaluation are detailedbelow.

B. Evaluation Metrics

We have used the below metrics to evaluate our implementation [18] of LightGCN, Matrix Factorization, and Random baselineapproach:

Precision@K: By definition, it is the percentage of recommended items in the top-k that are relevant. Here, we have calculated the proportion of the positive items (relevant items) to the overall top k positive items (places) for a user and we havetaken an average across all the users. Precision refers to the quality of the metrics.

Recall@K: By definition, it is the percentage of relevant items found in the top-k recommendations. It refers to the quantity of the metrics. Here we have considered the

proportion of positiveitems of a user for top k positive items for overall ground truth(actual ratings of a user to the item). We have then taken an average across all the users.

$$P = \frac{\#\text{ of our recommendations that are relevant}}{\#\text{ of items we recommended}}$$

Precision

$$r = \frac{\#\text{ of our recommendations that are relevant}}{\#\text{ of all the possible relevant items}}$$

Recall (10)

NDCG@K (Normalized Discounted Cumulative Gain): By definition, NDCG is the ratio of the suggested order or ranking of the items (DCG) over the ideal order. While precision and recall consider the relevant positive items for top k items. NDCG metric also considers the ranking factor. Here DCG calculates the summation of the weighted scores based on the predicted ranking or order or position of the items for a given user. It then takes the ratio of this DCG to the ideal order of the travel places or items for a given user. We then calculate the average acrossall the users.

$$\mathrm{DCG_p} = \sum_{i=1}^{p} \frac{2^{rel_i} - 1}{\log_2(i+1)}$$

$$\mathrm{IDCG_p} = \sum_{i=1}^{|REL_p|} \frac{2^{rel_i} - 1}{\log_2(i+1)}$$

$$\mathrm{nDCG_p} = \frac{\mathrm{DCG}_p}{\mathrm{nDCG}_p}$$

p: a particular rank position
$rel_i \in \{0, 1\}$: graded relevance of the result at position i
$|REL_p|$: list of items ordered by their relevance up to position p

(11)

C. Dataset Explanation

For the implementation of the LightGCN model we are using the following open-source datasets:

1. Yelp review datasets
2. Indonesia tourism Kaggle dataset Volume and Features of the datasets:

Yelp review dataset: The data is a collection of Yelp reviews, businesses, users, and check-ins for the Phoenix, AZ metropolitan area [8]. It has 229,907 yelp reviews.

There are 200K+ records of user and place ratings and 32 features. The features in this dataset include user information such as user id, Name, Reviewer type, Review count, place details such as business city, business full address, business id, business categories, and stars field for users' feedback to placesvisited.

Indonesia tourism Kaggle dataset: Indonesia tourism Kaggle dataset [7] contains several tourist attractions in 5 major cities in Indonesia, namely Jakarta, Yogyakarta, Semarang, Bandung, and Surabaya. It has data since 2015. There are 10,000 records with 16 features. The features of this dataset include user information such as user Id, Location, Age,and place information such as Place Id, Place Name, Description, Category, City, Price, Coordinate, Lat, and rating field for users' feedback to places visited.

D. Training Framework

For implementing the LightGCN model and Matrix Factorization we have used libraries: PyTorch, Pytorch_geometric – It is PyTorch library, mainly used for Graph Neural Networks (GNN) models.
Networkx – Python package for graph networks.
Torch-scatter, Torch-sparse, Sklearn, NumPy, Pandas, Matplotlib

E. Training Procedure

LightGCN data preprocessing: We have taken the Yelp reviews dataset and the Indonesia tourism destination dataset and converted them to graph format considering explicitly the relation or interaction between the users given to the travel items (destinations) using user and place unique mappings. Here the users and the places are represented as nodes and the interaction ie., the user rating to a place is considered as the edge or the link between the nodes. As we wanted to recommend the destinations that the user is most interested in, we have only considered the edges whose ratings are >=4 for a range of 0-5 ratings (user rating to a place).
We have taken this data and split it to test, train, and validation data sets. The ratio of data split is 90, 5, 5 for train, validation, and test sets respectively. We have used the train dataset for training the LightGCN algorithm and validated our results using the validation set. Finally, the rating prediction scores or recommendations of places or destinations for a user are provided using the trained initial embeddings on the test dataset.

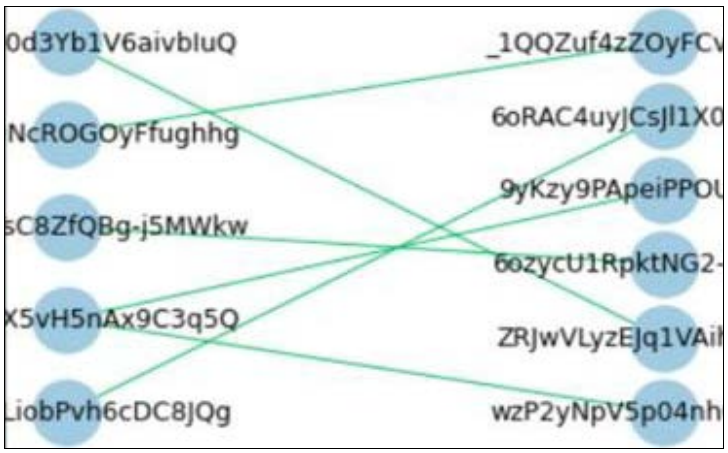

Fig 2: Yelp Reviews sample graph data

Fig. 2 represents the graph data for the yelp reviews dataset. Here the nodes are users and the destinations (places) and the edges represent the rating (link) for a user to an item (place).

Hyperparameter Tuning: For the Travel or trip recommendation system, we have performed hyperparameter tuning for both yelp reviews and Indonesia tourism destination datasets for the LightGCN model. For this, we have varied each hyperparameter individually and observed the performance metrics and finalized the parameters based on overall performance.

Hyperparameter tuning for Yelp Reviews Dataset: Upon training the LightGCN model, we have varied the number of message-passing layers for the validation set to observe the performance evaluation.

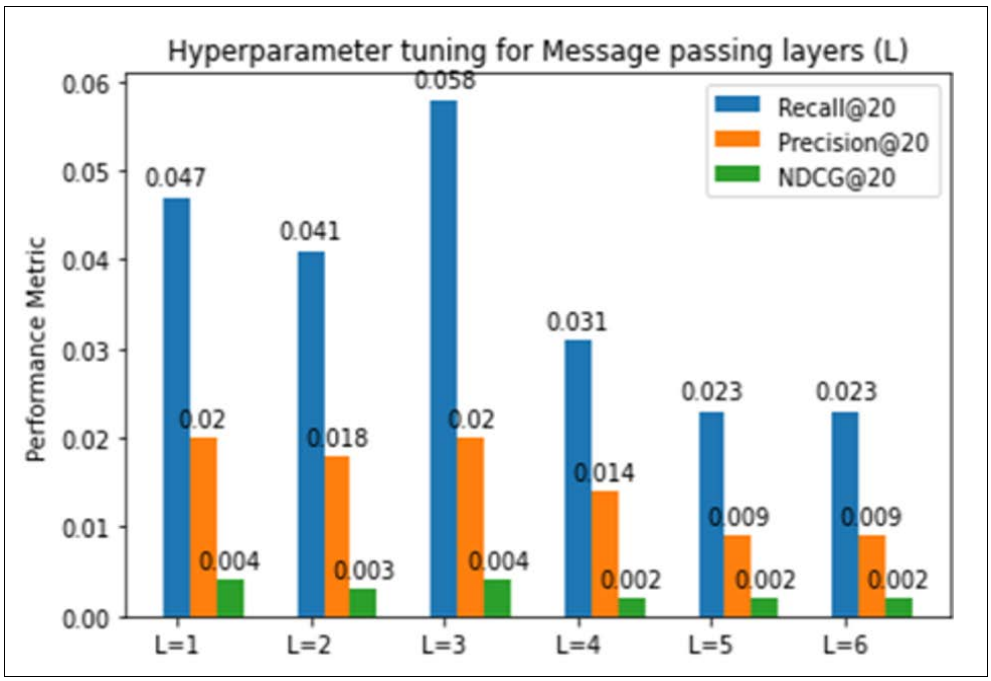

Fig 3: Varying message passing layers for Yelp reviews dataset

From Fig.3, we can observe that when using three message- passing layers the precision, recall, and NDCG metrics show better performance.

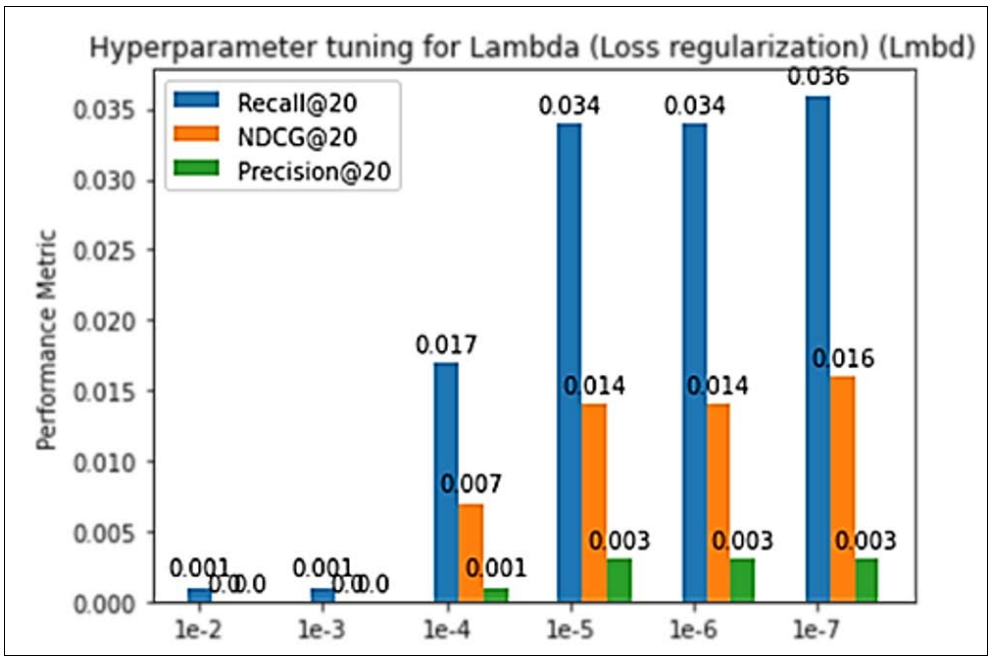

Fig 4: Hyperparameter tuning for lambda loss for the yelp dataset

In Fig. 4 we have varied loss regularization function lambda and could observe better performance metric results for lambda value 1e-7.

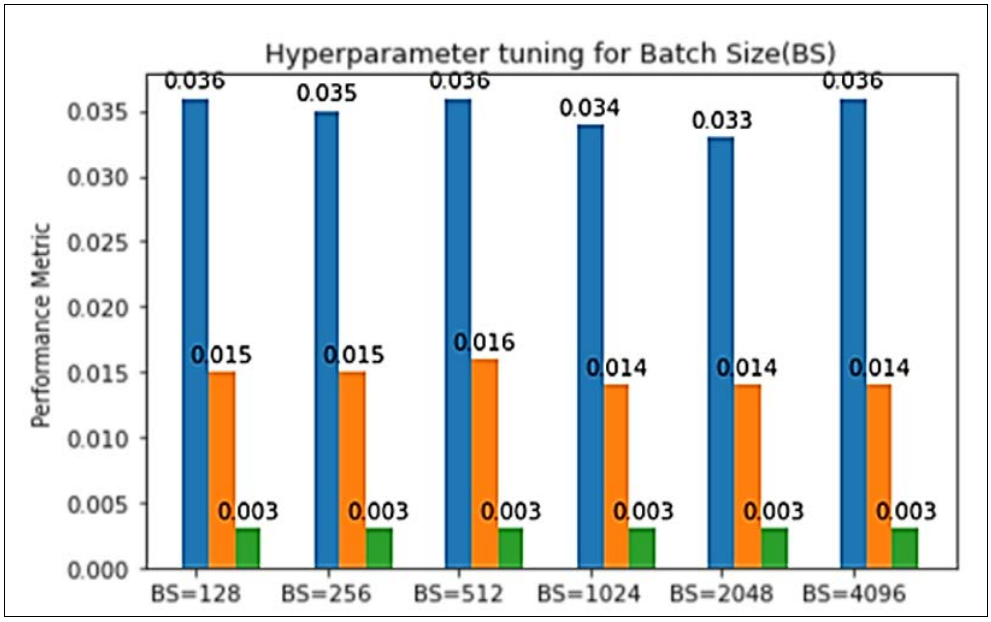

Fig 5: Hyperparameter tuning for batch size for the yelp dataset

We have varied the batch size in Fig. 5 and observed that there is not much difference in varying the batch size metrics.

We have similarly performed hyperparameter tuning for the Indonesia Tourism destination Dataset. The hyperparameter tuning is as shown below:

Hyperparameter tuning for the Indonesia Tourism destination Dataset: For Message-passing layers variation-hyperparameter tuning, Fig. 6 has better performance metrics for two message-passing layers for this dataset.

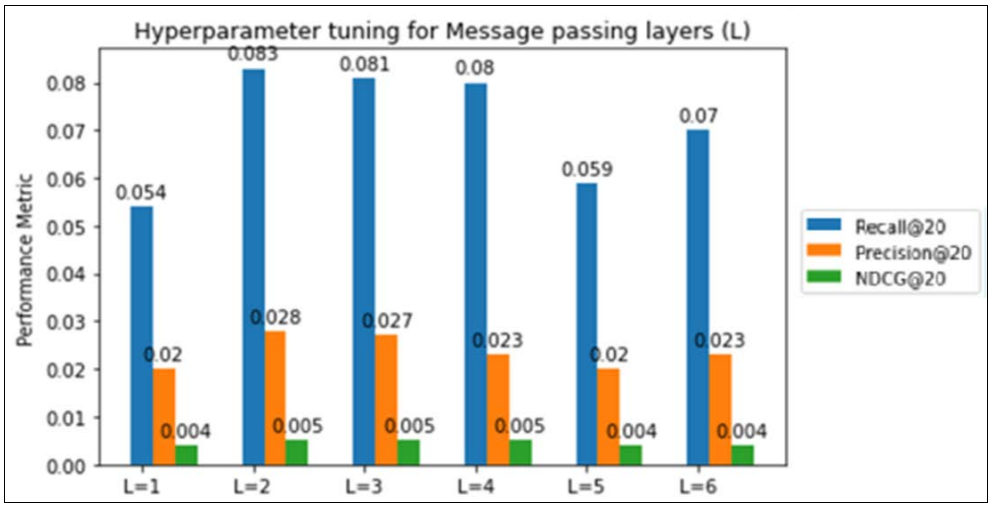

Fig 6: Hyperparameter tuning for message passing layers Indonesia dataset

In Fig. 7 we have performed Lambda loss regularization variation and observed that for lambda value 1e-3 the performance is better than other values.

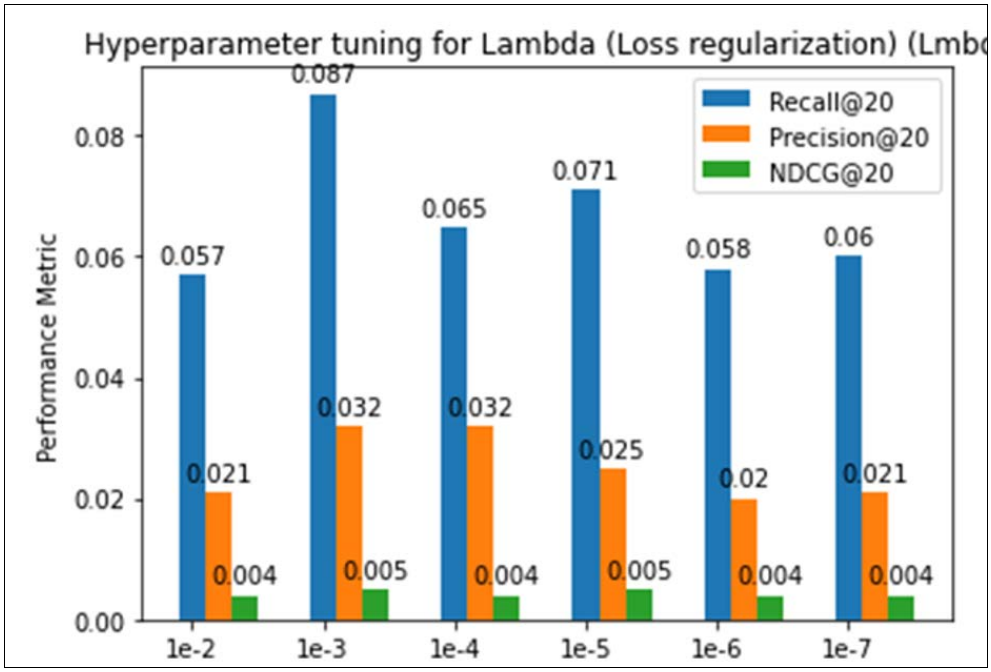

Fig 7: Hyperparameter tuning for loss function Indonesia dataset

We have varied the batch size in Fig. 8 and observed theperformance is better with batch sizes 256 and 1024.

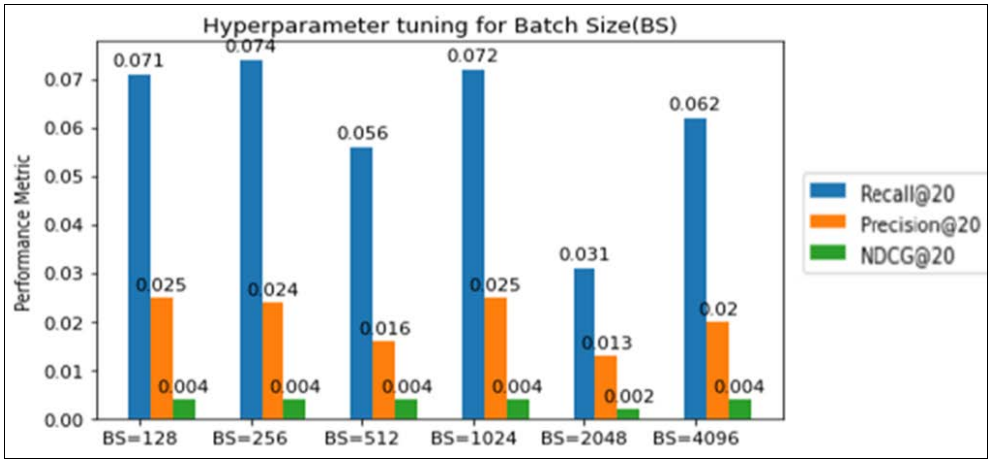

Fig 8: Hyperparameter tuning for Indonesia tourism dataset for batch sizes

Upon performing the hyperparameter tuning we have finalizedthe parameters as shown below:

Table 2: Final LightGCN Parameters

Final LightGCN Trainingparameters	Yelp Review Dataset	Indonesia Tourism dataset
Iterations	10,000	10,000
Batch size	1024	1024
Learning rate (LR)	1.00E-03	1.00E-03
Iterations per evaluation	200	200
Iterations per LR	200	200
Lambda (Loss Regularization)	1.00E-07	1.00E-03
Message passing Layers	3	2

Loss function graphs

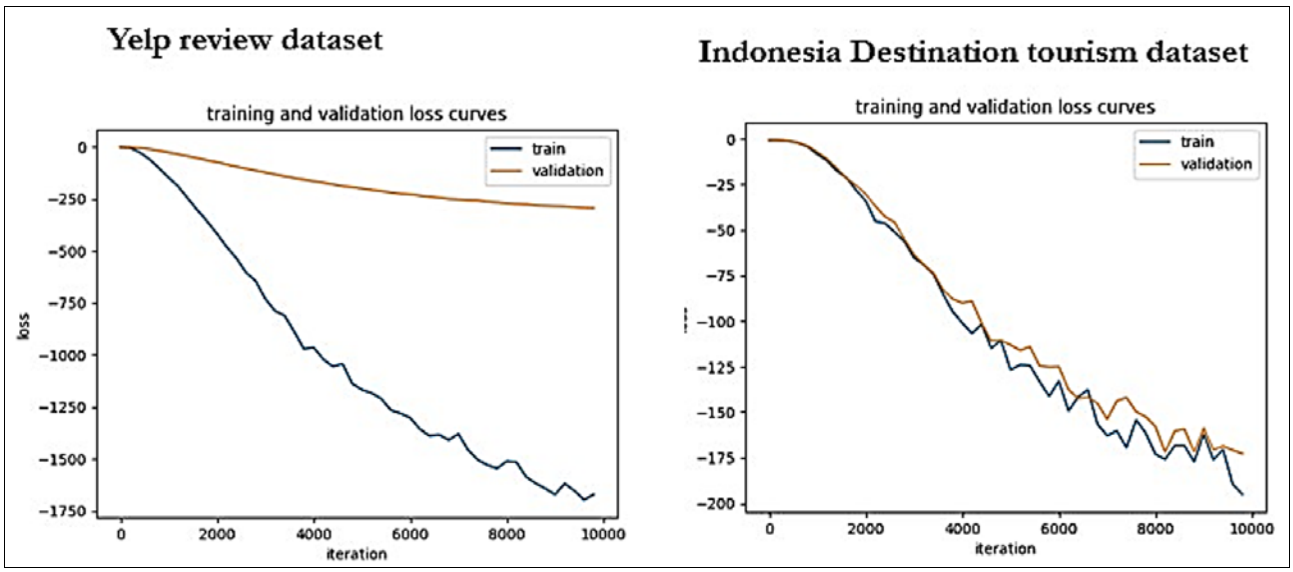

Fig 9: Loss function graphs

Fig. 9 shows the loss function plot. We have observed a significant decrease in the validation and training dataset loss.

Our code is available at https://github.com/SravaniPrakki/TravelRecommendation_ML

Results

In this section, we mention the comparison of LightGCN results for travel recommendations with Matrix Factorization and Random baseline. We also mention the sample place recommendations suggested to the user using LightGCN.

Comparison of Results: We have performed a comparison of the evaluation metrics for the three methods LightGCN, Matrix Factorization, and Random baseline. From the below figures, Fig. 10 and Fig. 11, we observe that the graph-based model LightGCN has better performance results for all the metrics Recall@20, Precision@20, and NDCG@20 in comparison to the non-graph-based traditional MF model and Random baseline.

Table 3: Performance evaluation for yelp reviews dataset

Test Set			
Methods	**Recall@20**	**Precision@20**	**NDCG@20**
LightGCN	0.03404	0.00252	0.01319
Matrix Factorization	0.01274	0.00083	0.00664
Random Baseline	0.00125	9e-5	0.00043
Validation Set			
Methods	Recall@20	Precision@20	NDCG@20
LightGCN	0.03167	0.0025	0.01201
Matrix Factorization	0.01367	0.00089	0.00767
Random Baseline	0.00196	0.00014	0.00085
Train Set			
Methods	Recall@20	Precision@20	NDCG@20
LightGCN	0.03511	0.00771	0.01908
Matrix Factorization	0.01452	0.0026	0.00993

Table 4: Performance evaluation for Indonesia tourism destination dataset

Test Set			
Methods	**Recall@20**	**Precision@20**	**NDCG@20**
LightGCN	0.09195	0.00655	0.03492
Matrix Factorization	0.04195	0.00379	0.0162
Random Baseline	0.02586	0.00207	0.00963
Validation Set			
Methods	Recall@20	Precision@20	NDCG@20
LightGCN	0.07135	0.00445	0.03011
Matrix Factorization	0.0468	0.00274	0.01405
Random Baseline	0.0234	0.00205	0.01039
Train Set			
Methods	Recall@20	Precision@20	NDCG@20

LightGCN	0.04955	0.0305	0.04002
Matrix Factorization	0.04349	0.02683	0.03419

From the comparison of results, we can observe that LightGCN has achieved very good performance in comparison to Matrix Factorization and Random Baseline. We have achieved a 167% increase in precision, a 203% increase in recall, and a 98.6% NDCG increase in metrics in comparison to Matrix Factorization.

The comparison of results for travel-based recommendations isshown below:

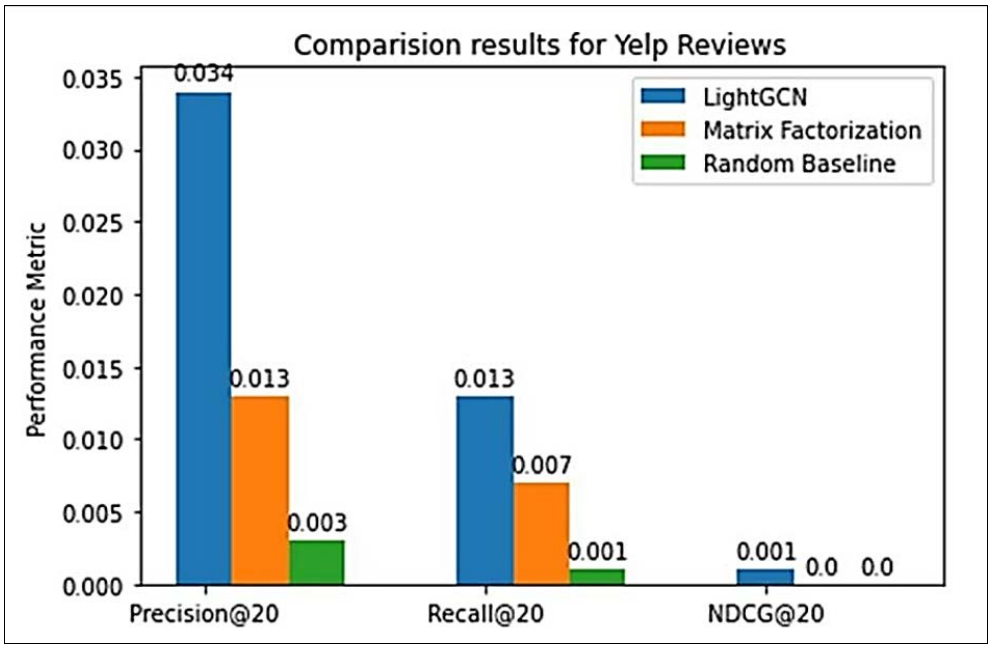

Fig 10: Comparison results for yelp reviews

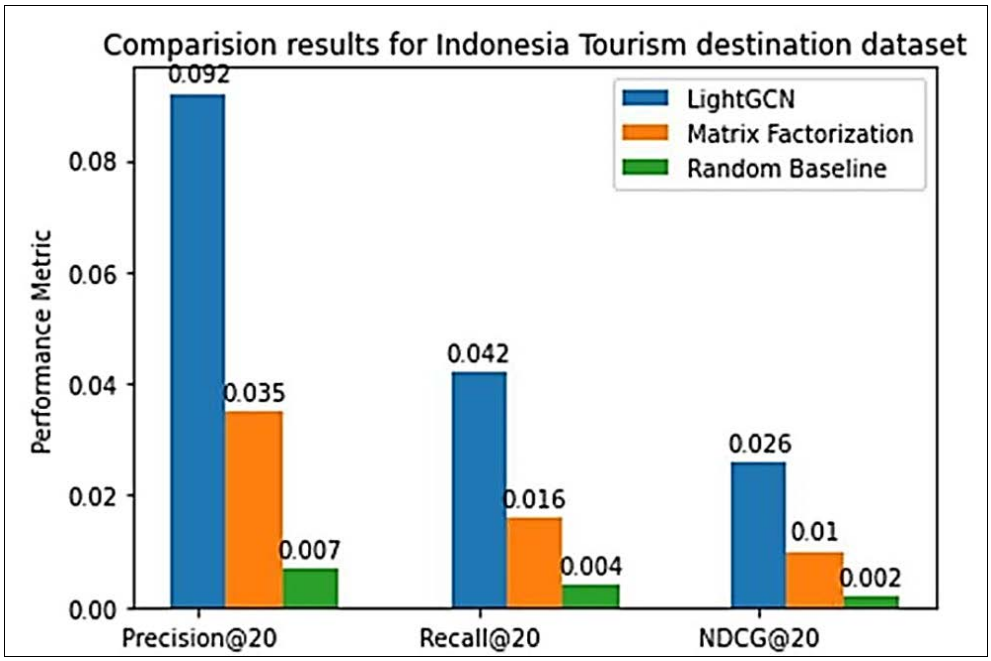

Fig 11: Comparison results for Indonesia tourism dataset

Sample Results: Top K place recommendations for a givenuser using the LightGCN approach for both the yelp review and Indonesia tourism datasets:

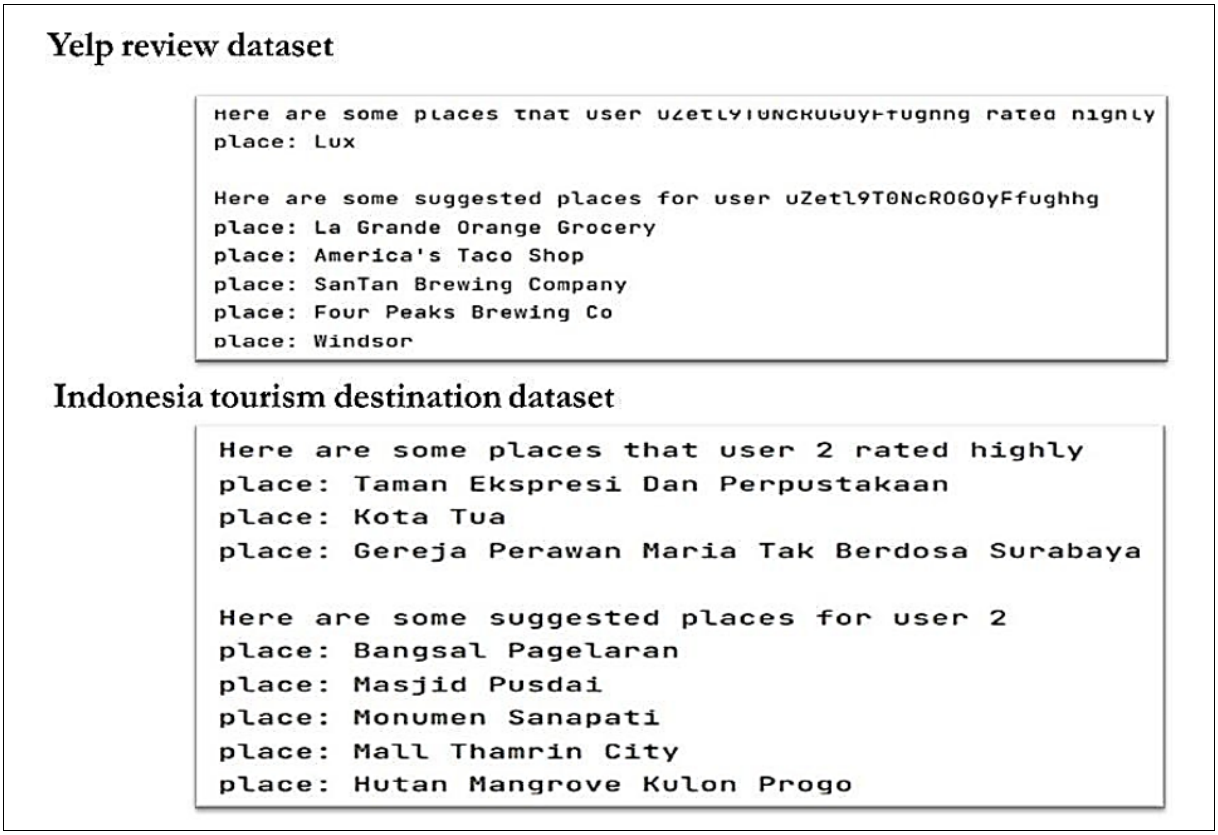

Fig 12: Sample results

Summary

We have implemented a graph-based model- "LightGCN" and a non-graph-based traditional model- "Matrix Factorization" for travel or trip recommendation systems on yelp reviews and Indonesia tourism datasets. We have performed training and hyperparameter tuning for both LightGCN and Matrix Factorization models for travel or place recommendations. We have evaluated our LightGCN model for place recommendations using Precision@K, Recall@K, and NDCG@K metrics and generated results after training the model to predict the top destination recommendations for a given user. We then compared our results of the LightGCN approach with the Matrix Factorization and Random baseline approaches.

We achieved better performances [167% increase in precision, 203% increase in recall, and 98.6% NDCG increase in metrics] for the LightGCN model (graph-based approach) compared to the Matrix Factorization approach for travel or place recommendations.

Future Work

In this work, we have implemented a transductive model. Here the learned embeddings cannot be generalized to unknown values or values other than the implemented dataset. We can extend this approach to adapt to inductive models such as IGMC (Inductive Graph Matrix Completion) which can be generalized to any unknown or unseen data. This experiment also focuses on user-item interaction for link-based predictions. The model could be further extended to include node-level and graph-level predictions. This work can also be extended to implement other Graph models such as Graph sage and Graph attention neural networks to observe the metrics for travel recommendations.

References

1. He, Xiangnan, *et al*. Lightgcn: Simplifying and powering graph convolution network for recommendation." Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval; c2020.
2. Bentaleb A, El Bouzekri El Idrissi Y, Ait Lahcen A. A Review on Content Based Recommender Systems in Tourism. In: Motahhir, S., Bossoufi, B. (eds) Digital Technologies and Applications. ICDTA 2021. Lecture Notes in Networks and Systems. 2021, 211. Springer, Cham. https://doi.org/10.1007/978-3-030-73882-2_48
3. Jia Z, Yang Y, Gao W, Chen X. User-Based Collaborative Filtering for Tourist Attraction Recommendations, IEEE International Conference on Computational Intelligence & Communication Technology; c2015. p. 22-25. DOI:10.1109/CICT.2015.20.
4. Chen Lei, *et al*. Multi-view graph attention network for travel recommendation. Expert Systems with Applications. 2022;191:116234.
5. https://towardsdatascience.com/graph-neural-network-gnn-architectures- for-recommendation-systems-7b9dd0de0856
6. https://neptune.ai/blog/graph-neural-network-and-some-of-gnn- applications
7. https://www.kaggle.com/datasets/aprabowo/indonesia-tourism- destination
8. https://data.world/brianray/yelp-reviews/workspace/file?filename=yelp_training_set_review.csv
9. https://medium.com/stanford-cs224w/lightgcn-with-pytorch-geometric- 91bab836471e
10. Berg Rianne Van Den, Thomas Kipf N, Max Welling. Graph convolutional matrix completion. arXiv preprint.

2017;ar14:1706.02263.
11. Xin Haoran, *et al*. Out-of-town recommendation with travel intention modeling. Proceedings of the AAAI Conference on Artificial Intelligence; c2021, 35(5).
12. Wang Meng. Applying Internet information technology combined with deep learning to tourism collaborative recommendation system. Plos one. 2020;15(12):e0240656.
13. Kbaier Mohamed Elyes Ben Haj, *et al*. A Personalized Hybrid TourismRecommender System. IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA); c2017. p. 244-250.
14. Wang Xiang, *et al*. Neural graph collaborative filtering. Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval; c2019.
15. Zhang Muhan, Yixin Chen. Inductive matrix completion based on graph neural networks. arXiv preprint arXiv:1904.12058; c2019.
16. Koren, Yehuda, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. Computer. 2009;42(8):30-37.
17. https://en.wikipedia.org/wiki/Matrix_factorization_(recommender_syste ms)
18. https://medium.com/@m_n_malaeb/recall-and-precision-at-k-for- recommender-systems-618483226c54
19. https://towardsdatascience.com/recommender-systems-matrix- factorization-using-pytorch-bd52f46aa199https://medium.com/stanford-cs224w/lightgcn-with-pytorch-geometric-91bab836471e
20. https://github.com/pyg-team/pytorch_geometric/issues/1876
21. https://jovian.ai/aakanksha-ns/anime-ratings-matrix-factorization/v/10?utm_source=embed
22. https://towardsdatascience.com/recommendation-system-matrix- factorization-d61978660b4b
23. https://github.com/XingxingHuang/Machine_Learning_Practices/blob/master/ML_python_implementation/matrix_factorization.py
24. https://datascience.stackexchange.com/questions/40590/does-a-matrix- factorization-recommendation-engine-use-user-item-related-features
25. https://d2l.ai/chapter_recommender-systems/mf.html
26. https://datascience.stackexchange.com/questions/40590/does-a-matrix- factorization-recommendation-engine-use-user-item-related-features
27. https://beckernick.github.io/matrix-factorization-recommender/
28. https://github.com/beckernick/matrix_factorization_recommenders
29. https://everdark.github.io/k9/notebooks/ml/matrix_factorization/matrix_ factorization.nb.html
30. https://machinelearningmastery.com/precision-recall-and-f-measure-for- imbalanced-classification/#:~:text=Once%20precision%20and%20recall%20have,)
31. %20%2F%20(Precision%20%2B%20Recall)
32. https://www.google.com/search?q=precision%2C+recall+formula&rlz=1C1CHBF_enUS938US938&oq=precision%2C+recall+formula&aqs=chrome.69i57j0i512j0i22i30l8.6756j0j7&sourceid=chrome&ie=UTF-8
33. https://medium.com/towards-data-science/factorization-machines-for- item-recommendation-with-implicit-feedback-data-5655a7c749db
34. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_recall_fsco re_support.html
35. https://machinelearningmastery.com/precision-recall-and-f-measure-for- imbalanced-classification/
36. https://medium.com/@MohammedS/performance-metrics-for- classification-problems-in-machine-learning-part-i-b085d432082b
37. https://www.analyticsvidhya.com/blog/2020/04/confusion-matrix- machine-learning/
38. https://www.analyticsvidhya.com/blog/2018/06/comprehensive-guide- recommendation-engine-python/?
39. https://docs.fast.ai/tutorial.collab.html
40. https://github.com/EsratMaria/Improved-Movie-Recommendation- System-with-KNN-and-Cosine-Similarity/blob/master/Recommendation%20System%20with%20KNN
41. %20and%20Cosine%20similarity.ipynb
42. https://towardsdatascience.com/collaborative-filtering-on-anime-dataset- using-fastai2-130ae32fe433
43. https://medium.com/unpackai/metrics-for-classification-in-fastai- ab110fd8fbe8
44. https://forums.fast.ai/t/getting-metrics-on-validation-set/79827/4
45. https://towardsdev.com/compute-performance-metrics-f1-score- precision-accuracy-for-cnn-in-fastai-959d86b6f8ad
46. https://medium.com/unpackai/metrics-for-classification-in-fastai- ab110fd8fbe8
47. https://towardsdatascience.com/collaborative-filtering-on-anime-dataset- using-fastai2-130ae32fe433
48. https://forums.fast.ai/t/assertion-error/79647?replies_to_post_number=2
49. https://stackoverflow.com/questions/60449931/fastai-cuda-error-device- side-assert-triggered
50. https://colab.research.google.com/drive/1Nd9alFlfNvlFrTRGFYFhmLG kP7Iy1kle?usp=sharing#scrollTo=OxyQhPsJe2ZN
51. https://medium.com/stanford-cs224w/friend-recommendation-using- graphsage-ffcda2aaf8d6
52. https://colab.research.google.com/drive/1Nd9alFlfNvlFrTRGFYFhmLG kP7Iy1kle?usp=sharing
53. https://colab.research.google.com/drive/1uF6sLL65HjWh6mgL4wTyC Aq8xtxoXhgw
54. https://colab.research.google.com/drive/1phV7VouGG0T-7eBtHo_44bDCRB9xJyPA#scrollTo=o2P3zYR8Q8EX
55. https://colab.research.google.com/drive/1pl9F02cFbGDCBWUhUcylF9 qWj27zNMzv#scrollTo=nPgvoKnlyEGN
56. https://github.com/AntonioLonga/PytorchGeometricTutorial/blob/main/ Tutorial1/Tutorial1.ipynb
57. https://colab.research.google.com/drive/1h3-vJGRVloF5zStxL5I0rSy4ZUPNsjy8?usp=sharing
58. https://medium.com/@benalex/implement-your-own-music- recommender-with-graph-neural-networks-lightgcn-f59e3bf5f8f5
59. https://colab.research.google.com/drive/1EdgZaTb8mtc4vEnedNNtRyg Z_Ls-jQqy?usp=sharing

60. https://colab.research.google.com/drive/1EdgZaTb8mtc4vEnedNNtRyg Z_Ls-jQqy?usp=sharing#scrollTo=3gxBd579e8Ox
61. https://www.youtube.com/watch?v=AQU3akndun4&t=154s
62. https://medium.com/stanford-cs224w
63. https://medium.com/analytics-vidhya/how-to-use-google-colab-with- github-via-google-drive-68efb23a42d
64. https://www.jianshu.com/p/767950b560c4
65. https://medium.com/@jn2279/better-recommender-systems-with- lightgcn-a0e764af14f9
66. https://chowdera.com/2022/04/202204211353274448.html
67. http://ethen8181.github.io/machine-learning/recsys/4_bpr.html
68. https://discuss.pytorch.org/t/issue-about-bpr-loss/134840
69. https://medium.com/@bhawna7374/movie-recommendation-system- with-neural-networks-and-collaborative-filtering-explicit-feedback- d2afaafef350
70. https://discuss.pytorch.org/t/problem-with-dimension-in-graph-neural- networks/127296
71. https://colab.research.google.com/drive/1KKugoFyUdydYC0XRyd dcROzfQdMwDcnO?usp=sharing
72. https://towardsdatascience.com/building-a-recommendation-system- using-neural-network-embeddings-1ef92e5c80c9
73. https://github.com/bhawnapaliwal/Collaborative-Filtering-with-Neural-Network/blob/master/COLLABORATIVE%20FILTERING%20USING
74. %20NEURAL%20NETWORKS%20FOR%20EXPLICIT%20%20FEEDBACK%20RECOMMENDATION%20SYSTEMS.pdf
75. https://colab.research.google.com/drive/1KftirIug97suNj0rrd0f7IHF8lc7 RLNa
76. https://colab.research.google.com/drive/1eAgTLR4lCy2_j8kNDZA7OlZ7b0G8CETY#scrollTo=lG6dE0LoS4D0
77. https://colab.research.google.com/github/sparsh-ai/coldstart-recsys/blob/main/docs/L281872_Cold_Start_Recommendations.ipynb#s crollTo=3XcP6T_wjzz8
78. https://colab.research.google.com/drive/17Eq-hjU0fSUmSRqBK6TjZNCfDaNy_DLR#scrollTo=14-NSkJ2VHGd
79. https://colab.research.google.com/github/google/eng-edu/blob/main/ml/recommendation-systems/recommendation- systems.ipynb https://www.youtube.com/@machinelearningalchemy9621/videos
80. https://www.youtube.com/@welcomeaioverlords