NEXT POINT OF INTEREST RECOMMENDATION THROUGH GRAPH NEURAL NETWORKS

Giancarlo Tedesco
Università degli Studi di Roma "La Sapienza"
Rome, Italy
tedesco.2057231@studenti.uniroma1.it

ABSTRACT

Point-of-Interest (POI) recommendation is a critical task in location-based services, aiming to provide personalized suggestions of places to users based on their preferences and historical behavior. This work explores the application of Graph Neural Networks (GNNs) to the POI recommendation problem. Two prominent GNN architectures were leveraged: graphSAGE and LightGCN, and evaluate their performance on a real-world NYC check-in dataset. The work delves into the challenges encountered in data preparation, model design, and evaluation, highlighting the importance of handling sparse data and adapting evaluation metrics to implicit feedback scenarios.

Keywords Point-of-Interest (POI) recommendation · Graph Neural Networks (GNNs) · graphSAGE · LightGCN · link prediction · recommender systems

1 Introduction

The widespread use of location-based services and the rapid growth of user-generated data have made Point-of-Interest (**POI**) recommendation systems increasingly important. POI recommendation involves predicting and suggesting places of interest to users based on their preferences, past activities, and contextual information.

Graph Neural Networks (**GNNs**) have emerged as a groundbreaking paradigm for modeling complex relationships and dependencies within data. Their unique ability to operate on graph-structured data makes them particularly well-suited for POI recommendation. GNNs can effectively represent the intricate network of connections between users, POIs, and various contextual attributes. By learning from these interconnected relationships, GNNs can uncover hidden patterns and correlations that traditional methods often miss.

2 Next POI recommendation

GNNs are Ideal for POI Recommendation, since the task has to consider carefully different aspects and handle different scenarios. In particular:

- **Capturing Complex Relationships:** GNNs can model the intricate relationships between users, POIs, social connections, and even temporal sequences.
- **Addressing Data Sparsity:** GNNs utilize the graph structure to propagate information and learn representations even for sparsely connected nodes.
- **Incorporating Multimodal Information:** GNNs can effortlessly integrate diverse data sources, such as user profiles, POI attributes, social networks, and check-in histories.
- **Handling Dynamic Factors:** GNNs can effectively model temporal dependencies and user mobility patterns. This dynamic capability allows them to adapt to changing user preferences and provide recommendations that are relevant to the current context.

Moreover, to obtain a well-made GNN model, one has to manage a representative graph structure where the correctness of its components is crucial: **nodes** represent entities such as users and POIs, each annotated with features like preferences, POI categories, and locations; **edges** represent relationships and interactions between these entities, with features such as the frequency of visits, temporal context, and social connections. Any inaccuracies can lead to misleading patterns and reduce the quality of recommendations.

3 Proposed Methods

In this work, two state-of-the-art GNN architectures are employed: **graphSAGE** and **LightGCN**. These models are applied to a carefully constructed **bipartite graph**, where users and POIs constitute the two disjoint sets of nodes, and edges represent historical check-ins and interactions. By learning on this structured representation, the framework effectively models user preferences and POI characteristics, enabling a personalized next POI recommendations.

3.1 Data Pre-processing

To prepare the data for analysis and modeling with GNNs, a series of meticulous pre-processing steps are performed. These steps aim to clean, transform, and enrich the data to ensure its suitability for learning meaningful representations.

- Data Cleaning and Filtering: the raw dataset is loaded, and its structure and content are carefully examined to identify potential issues like missing values, inconsistencies, or irrelevant attributes. Users with less than 26 check-in records are filtered out. This number is selected meticulously to ensure test and evaluation batches contain enough samples to be evaluated. Each user's check-in history is sorted chronologically. This step is crucial for capturing temporal dependencies in user behavior and establishing the sequence of their visits to various POIs.
- Data Transformation and Normalization: unique identifiers for venues and venue categories are reassigned to ensure consistency and facilitate subsequent analysis and modeling. The timestamps associated with check-ins are converted from UTC to the local time zone of each respective venue. Duplicate check-in records (likely caused by system errors) are identified and removed.
- Feature Engineering: the timestamps of user check-ins are transformed into a relative time representation. This representation indicates the time elapsed (in minutes) since the user's first recorded check-in. This approach emphasizes the temporal order of check-ins and can potentially help the model better capture the evolution of user preferences over time.

Unfortunately, the latter feature was not used because it had to be applied to a different architecture model that was discarded due to computationally demanding resources.

The dataset is divided into two distinct dataframes: one containing user-specific information (user ID, venue ID, and time), and another containing POI details (venue ID, location coordinates, and category). This separation facilitates the creation of a bipartite graph structure, where users and POIs are treated as distinct node types.

3.2 Graph Construction

A specialized graph data structure is created to represent the bipartite nature of the user-POI interactions. This structure distinguishes between two types of nodes (users and POIs) and defines the allowed connections between them.

- **Node Feature Creation:** While user nodes are initially assigned no explicit features, POIs are enriched with attributes derived from the dataset. These attributes typically include information like the POI category and its geographical coordinates (latitude and longitude). This step prepares the graph for learning representations that incorporate the unique characteristics of each POI.
- **Edge Definition:** Edges are established between user and POI nodes based on the historical check-in data. Each edge signifies a past interaction between a user and a POI, indicating that the user visited that particular location. The edges are carefully constructed to respect the bipartite nature of the graph, ensuring that connections only exist between nodes of different types.

The result is an undirected graph representation of the user-POI interaction data.

4 Architecture Design

This section delves deeper into the architectural designs of the two models mentioned in paragraph 3.

4.1 GraphSage

GraphSAGE (**Graph SAmple and AggreGatE**) is chosen as a possible approach to the POI recommendation task due to its unique ability to effectively learn node representations in large-scale graphs [1].

With this in mind, this methodology is exploited in a link prediction task to explore and become familiar with the world of graph neural networks.

As illustrated in its reference paper [1], GraphSAGE adopts inductive learning, meaning it can generalize to unseen nodes. This is a significant advantage in POI recommendation, as new users and POIs are constantly being added to the system.

GraphSAGE employs a sampling-based approach to aggregate information from neighboring nodes. In the current implementation, the default mean aggregation function is leveraged within SAGEConv layers [1].

GraphSAGE has demonstrated strong performance in various graph-based tasks, including node classification, link prediction, and recommendation [2].

4.1.1 Implementation

The dataset is divided into three distinct sets: a training set (70%), a validation set (10%), and a test set (10%) based on randomly splitting the edges. Moreover, negative samples (edges that don't actually exist in the data) are generated to provide a balanced representation of positive and negative interactions. Additionally, neighbor sampling is employed to select a subset of nodes connected to each node in the graph. This approach is very convenient because it allows for efficient handling of large graphs by focusing on relevant subgraphs, reducing computational overhead. By iteratively sampling neighbors, it maintains the structural integrity of the graph, ensuring that the generated subgraph is representative of the local topology.

4.1.2 Model Architecture Implementation

The node representation model is a 3-layer GraphSAGE GNN.

To leverage the learned node representations, a standard classifier based on the dot product is employed: given the representations of a user and a POI, the classifier computes their dot product. This operation assesses the similarity or affinity between the user's preferences and the POI's characteristics. A higher dot product signifies a stronger alignment between the user and the POI, suggesting a higher probability of the user visiting that POI.

The initial node representations are obtained using embedding layers for both users and POIs. User embeddings are learned directly, while POI embeddings are a combination of learned embeddings and a linear transformation applied to the POI features (venue category ID, latitude, and longitude).

The model is trained using the binary cross-entropy with logits loss function, since the task is restricted to whether a link exists or not.

4.2 LightGCN

LightGCN is chosen as the key model in this POI recommendation project thanks to its unique blend of simplicity, effectiveness, and scalability [3]. LightGCN's architecture is remarkably simple, involving only a few essential components. This simplicity translates to high computational efficiency: this architecture avoids expensive operations like feature transformations and nonlinear activations, enabling faster training.

LightGCN is specifically designed for collaborative filtering. It excels at capturing the latent relationships between users and items (POIs in this case) through message passing on the user-item interaction graph. By aggregating information from neighboring nodes, LightGCN learns informative embeddings that effectively represent user preferences and item characteristics.

Specifically, LightGCN learns user and item embeddings by linearly propagating them on the user-item interaction graph, and uses the weighted sum of the embeddings learned at all layers as the final embedding [3].

4.2.1 Implementation

The first step in implementing LightGCN involves preparing the data. Edge Information are extracted as a **sparse tensor** where each column represents an edge. This choice is fundamental in order to model inherent sparsity of user-POI interaction data [4].

The dataset is divided into three sets: one for training the model, one for tuning hyperparameters during training (validation), and a final held-out set for unbiased performance assessment (test). The split ratio perfomerd for these sets is 70/20/10.

4.2.2 Model Architecture Implementation

The main idea behind this method is that user preferences and POI attributes can be represented as embeddings. LightGCN iteratively refines these embeddings by propagating information through the user-POI interaction graph. The key assumption is that users who visit similar POIs are likely to have similar preferences, and POIs visited by similar users are likely to share characteristics.

The network begins by initializing random embeddings for each user and POI in the dataset. These embeddings serve as starting points for the model to learn meaningful representations. Specifically, a 3-layer LightGCN architecture is deployed to capture both local and global information within the graph. Each layer performs message passing, weighted average aggregation and embedding update. After the 3 layers of propagation, the embeddings learned at each layer for both users and POIs are combined by taking a simple average. To predict whether a user is likely to visit a POI, the dot product is performed between the final user and POI embeddings. A higher dot product value indicates a stronger affinity between the user and the POI, suggesting a higher probability of a future visit. The code implementation is explicitly taken from the official repository of the paper here [4] and carefully modified to work within the project.

Since the original dataset only contains positive interactions (user visited POI), negative samples are generated. This is done by randomly sampling user-POI pairs that are not present in the dataset. The ratio of negative to positive samples is a hyperparameter that can be tuned.

Recommendation is inherently about ranking items. **BPR** (Bayesian Personalized Ranking) loss is particularly well-suited for this goal. Unlike traditional loss functions that aim for absolute prediction accuracy, BPR loss focuses on the relative ranking of items. It ensures that the model ranks items a user has interacted with (positive samples) higher than items they haven't interacted with (negative samples). In each training step, positive and negative items for each user in the batch are sampled and they are fed into this specific loss function: by employing the BPR loss, model is driven to learn embeddings that effectively capture the relative preferences of users for different POIs [5]. The code implementation is explicitly taken from here [6] and carefully modified to work within the project.

5 Dataset

The dataset [7] utilized in this project comprises check-in records from New York City (NYC) spanning approximately ten months, from April 12, 2012, to February 16, 2013. It encompasses a total of 227,428 check-ins, each of which contains the following information: User ID, Venue ID, Venue Category ID, Venue Category Name, Latitude, Longitude, Timezone and UTC Time.

This dataset was originally collected to investigate the spatial-temporal regularity of user activity in Location-Based Social Networks (LBSNs). It offers a rich source of information for studying how users move around and interact with different types of locations in a bustling urban environment like NYC

6 Results

As the project was developed, it would come naturally to propose a comparison between the first model (GraphSAGE) and the second (LightGCN). With the intention of adapting the first model to baseline, the task was reduced to a link prediction, also causing a loss of meaning in the metrics to be reported: since the network has to evaluate the existence or non-existence of a link, and thus a binary prediction exists/does not exist, the **acc@k** metric becomes meaningless. Nevertheless, simple **accuracy** accompanied by **MRR** was used to evaluate the model's capabilities. In the second case, the problem was approached through a real recommendation model, which can provide all the necessary information for proper evaluation, obviously limited by the colab working environment and the computational demanding performances of the models.

6.1 GraphSAGE

First, from what can be seen from the graph in Figure 1, the training and validation losses follow a similar trend, indicating that the model was effectively learning to differentiate between positive and negative interactions. To confirm the results, the two previously illustrated metrics were computed during the various validation steps at the end of the various epochs providing interesting results.

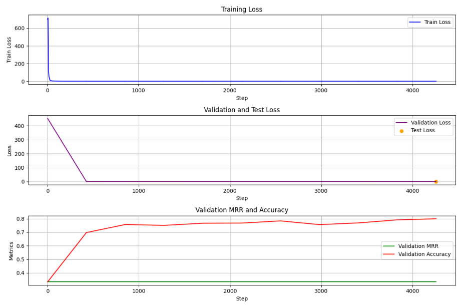

Figure 1: Result plots of GraphSage model. A validation step was run before training to perform a sanity check. Those results are logged as starting point values

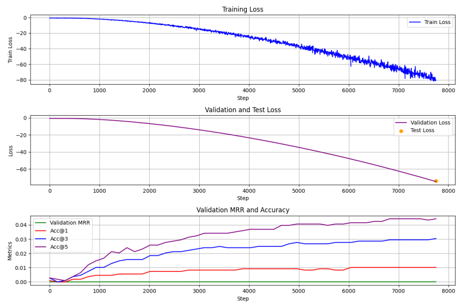

Figure 2: Result plots of LightGCN model. A validation step was run before training to perform a sanity check. Those results are logged as starting point values

- Accuracy: accuracy shows an upward trend standing at almost 80 percent. In contrast, the MRR shows a different behavior
- MRR: Interestingly, the MRR metric remained relatively constant throughout the training process. This can be attributed to the nature of the link prediction task and the evaluation methodology. Since the model is tasked with predicting binary interactions (visit or no visit), the ranking of POIs within a user's profile is not as crucial as simply identifying the correct positive POIs. Therefore, MRR, which focuses on the rank of the first positive item, may not be the most informative metric in this context.

6.2 LightGCN

A significant challenge arose during model evaluation on the test and validation sets. Due to the nature of these unseen datasets, some users had a very limited number of positive interactions (i.e., POIs they had actually visited). This posed a problem when computing top-K recommendations, as the number of positive interactions for certain users was less than the desired K. To avoid errors in the evaluation code, a maximum value for K was set based on the user with the fewest positive interactions within a batch. This meant that for users with more positive interactions, not all of their relevant POIs could be considered in the evaluation, potentially leading to an underestimation of the model's true performance.

An additional challenge arose in applying Mean Reciprocal Rank (MRR) to our POI recommendation task. MRR is traditionally used in scenarios where there's a clear ranking of items based on explicit user feedback (e.g., ratings). However, the dataset consists of implicit feedback (user visits), where there's no explicit indication of a user's preference order for different POIs. The POIs that a user interacted with during the training period were considered as the "ground truth" for that user. Then, MRR was calculated based on the rank of the single most relevant POI suggested by the model for each user

The LightGCN model demonstrated significant learning progress throughout the training process, as evidenced in Figure 2. The initial validation loss was -0.7, indicating a suboptimal starting point. However, the model quickly learned to optimize the Bayesian Personalized Ranking (BPR) loss, achieving a final test loss of -76.23.

- Accuracy@K: improved considerably. Initially, the accuracy was 0 for all K values (1, 3, and 5), suggesting that the model was not yet identifying the correct POIs. However, by the end of training, the model achieved an Accuracy@1 of 0.92%, Accuracy@3 of 2.49%, and Accuracy@5 of 3.69%.
- MRR: the MRR also saw a slight increase from its initial value to 5.99e-5. While this might seem small, it's important to note that MRR can be a challenging metric to improve significantly in large-scale recommendation settings due to the vast number of potential items.

7 Conclusion

This project represents my personal first experience into Graph Neural Networks (GNNs) and recommender systems. While both models showed promise, there were challenges, particularly in evaluating performance on test and validation sets with limited positive interactions. Additionally, I explored an alternative architecture that leveraged the bipartite graph structure as knowledge graphs, along with individual POI-POI graphs for each user, as illustrated in this paper [8].

However, this approach was computationally demanding, requiring over 16GB of VRAM even in the initial training stage, then it was not reasonable and evaluable.

Acknowledgments

This work has been done as project for the Deep Learning course held by Professor Fabrizio Silvestri during the academic year 2023/2024 in "La Sapienza" University of Rome.

References

[1] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs, 2018.

[2] Sarthak Bhatkar, Purva Gosavi, Vishakha Shelke, and John Kenny. Link prediction using graphsage. In *2023 International Conference on Advanced Computing Technologies and Applications (ICACTA)*, pages 1–5, 2023.

[3] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. Lightgcn: Simplifying and powering graph convolution network for recommendation, 2020.

[4] lightgcn recommender pyg. `https://github.com/ztor2/lightgcn_recommender_pyg`. Accessed: 2024-06-24.

[5] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback, 2012.

[6] Bpr gitrepo code. `https://github.com/sh0416/bpr/blob/master/train.py`. Accessed: 2024-06-25.

[7] Dingqi Yang, Daqing Zhang, Vincent W. Zheng, and Zhiyong Yu. Modeling user activity preference by leveraging user spatial temporal characteristics in lbsns. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 45(1):129–142, 2015.

[8] Junbeom Kim, Sihyun Jeong, Goeon Park, Kihoon Cha, Ilhyun Suh, and Byungkook Oh. Dynaposgnn: Dynamic-positional gnn for next poi recommendation. In *2021 International Conference on Data Mining Workshops (ICDMW)*, pages 36–44, 2021.

Appendix

Table 1: Loss, accuracy and MRR of GraphSAGE on test set

Model	**Loss**	**Accuracy**	**MRR**
GraphSAGE	0.464	0.799	0.333

Table 2: Loss, accuracy and MRR of LightGC on test set

Model	**Loss**	**Acc@1**	**Acc@3**	**Acc@5**	**MRR**
LightGC	-73.84	0.009	0.024	0.037	6.e-5