# GRAPH NEURAL NETWORKS FOR NEXT POINT OF INTEREST RECOMMENDATION

**Andrea Cadoli, Valerio Canavari, Davide Ceriola**
Sapienza University of Rome
{Cadoli.1837028, Canavari.2088936, Ceriola.1850563}@studenti.uniroma1.it

**ABSTRACT**

This report presents an advanced model for next Point of Interest (POI) recommendation, leveraging two Graph Neural Networks (GNNs) to handle spatial data and an LSTM to exploit temporal dependecies. By incorporating user clustering and embedding cluster centroids, the model captures shared POI preferences among similar users, improving prediction accuracy. Experiments on a real-world dataset show that the proposed model performances can compete with state-of-the-art methods in various metrics.

## 1 Introduction

Next Point of Interest (POI) prediction is a key task in location-based services, where the goal is to accurately forecast the next location a user is likely to visit based on their historical behavior. This task is challenging due to the complexity of human mobility patterns and the influence of both spatial and temporal factors. In our work, we draw inspiration from an existing model introduced in a previous paper [1], which effectively tackled these challenges by leveraging a combination of spatial and temporal data. Building upon this foundation, we introduce several changes, including user clustering and advanced embedding techniques, to further improve the model's ability to capture personalized user preferences and boost prediction accuracy.

## 2 Dataset

The dataset we utilized in our project is derived from the TSMC2014 dataset, specifically focusing on geospatial and temporal data collected in New York City. This dataset is a resource for analyzing urban dynamics, comprising detailed records of user check-ins at various locations across the city. Each record within the dataset is associated with multiple features, including temporal information, spatial coordinates, and the specific venues where the check-ins occurred.

### 2.1 Structure of the Data

The dataset is organized in a tab-separated format, where each row corresponds to a single check-in event. The primary attributes include:

- **User ID**: A unique identifier for each user, anonymized to protect privacy.
- **Venue ID**: A unique identifier for each location or venue where the check-in occurred.
- **Venue Category**: The type or category of the venue, such as a restaurant, park, or museum.
- **Latitude and Longitude**: The geographical coordinates of the venue, enabling spatial analysis.
- **Timestamp**: The date and time of the check-in, providing temporal context.

It comprehends 227428 check-ins in New York City, collected from Foursquare for about 10 months. We decided to adopt a 70-15-15 split practice based on users to divide the dataset into training, validation, and testing, to keep a balance between the need for a large training set to build a robust model, a sufficient validation set for fine-tuning, and a reliable testing set for final evaluation.

## 3 Model Architecture

The final architecture we adopted works with both spatial and temporal information at the same time. For each of them, we work with a different pipeline where we try to encode at best the dependencies. In addition to that, on another pipeline, we work on users' history to extract and encode further useful features. The resulting tensors are then concatenated and fed to a fully connected layer to predict the next POI ranked set.

In the following paragraphs, we describe each pipeline separately, providing the reasons behind our decisions and our final architecture is shown in Figure1.

### 3.1 Spatial Pipeline

In spatial analysis, we utilize graphs that represent users' movement patterns across different POIs, known as spatial graphs. By encoding the spatial proximity and connectivity between POIs, spatial graphs help us understand how location influences movement patterns. This approach allows for improved prediction accuracy in determining future movements by considering the spatial context of previous visits and how the physical arrangement of POIs influences user behavior. We employed Graph Neural Networks (GNNs) to process these spatial graphs, experimenting with both Graph Convolutional Networks (GCNs) and Graph Attention Networks (GATs).

**GCN:** GCNs operate by performing a convolution over graph-structured data, aggregating features from POIs based on their adjacency, effectively capturing spatial dependencies in the user's movement patterns. In our model, GCNs allowed us to learn the latent structure of user behavior by propagating and combining information across connected POIs, enabling the model to account for how nearby or directly connected locations influence a user's next visit.

**GAT:** On the other hand, GATs extend this capability by introducing a learnable attention mechanism that dynamically assigns different weights to each neighboring node. Unlike GCNs, which treat all neighbors equally during the aggregation process, GATs calculate attention coefficients based on the relevance of each neighboring POI to the current node. This allows the model to selectively focus on more influential POIs, which may have a stronger impact on a user's movement decisions.

### 3.2 Temporal Pipeline

In the temporal analysis, we utilize graphs that represent users' transition sequences between POIs, known as temporal graphs. With these structures, we can encode human mobility patterns by modeling routines, time-sensitive behaviors, and the dependencies between visited locations. Hence, by incorporating the order and timing of POI visits, temporal graphs allow us to improve the accuracy of predicting future movement patterns. In order to properly learn dependencies within the temporal graph we used an LSTM. It allows us to effectively model sequential data, capturing long-term dependencies and complex temporal patterns, crucial for understanding user movement behavior over time. Their architecture handles the vanishing gradient problem, ensuring accurate and context-aware predictions.

### 3.3 Users' History Pipeline

We leveraged the idea that users with similar characteristics often visit the same POIs. By analyzing user histories and identifying patterns of similarity, we aimed to enhance our model's performance. This approach allowed us to better capture user preferences and predict POIs by exploiting the correlations between users' behaviors and their visited locations.

**User Similarity:** To determine user similarity, we calculate the cosine similarity between the history $h_i$ of a user $i$ and the histories of all other users, identifying the most similar user $j$ to $i$. Once $j$ is identified, we utilize their spatial graph as described in Section 3.1, and compute a weighted average of the outputs from the GNNs for users $i$ and $j$ , assigning $70\%$ importance to user $i$ 's result and $30\%$ to user $j$ 's.

The decision to apply a weighted average, rather than an equal weighting, stems from our objective of prioritizing user $i$ 's individual movement patterns. This is particularly crucial in cases where user $i$ 's history exhibits low similarity with other users, as observed with outlier points (see Figure 3), where treating all histories equally could dilute the specific characteristics of user $i$ 's behavior.

**K-Means:** By applying the K-Means algorithm to divide users into 6 different classes based on a 2D representation of their histories, we aimed to capture distinct patterns in user mobility behavior. Assigning each user to the nearest centroid helps in reducing the complexity of individual trajectories by grouping similar behaviors together. To incorporate this clustering information into our model, the embedding of each user's assigned centroid is passed through a fully connected layer, in order for the model to integrate the cluster-level features into the learning process. This approach

helps the model generalize better by learning from these clustered patterns, leading to more robust predictions and potentially uncovering latent structure in the data that might be obscured in a high-dimensional space.

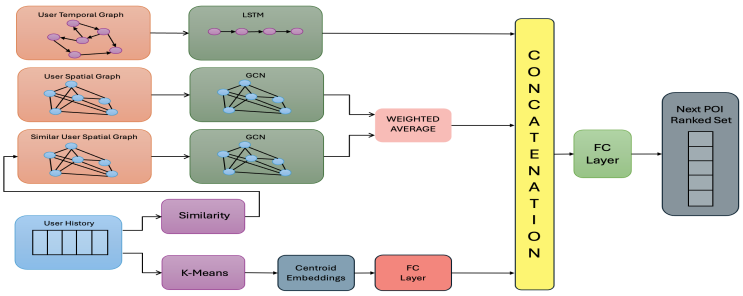


Figure 1: Illustration of final architecture.

# 4 Methodology

## 4.1 Preprocessing

To prepare the dataset for our deep learning model, we applied preprocessing steps, including converting textual timestamps into numerical values. This involved encoding the date and time as a single continuous variable by translating month names to numbers and concatenating the year, month, day, hour, minute, and second into a single integer for easier temporal comparisons.

### 4.1.1 Graphs

**Spatial:** Each user has its Spatial Graph and in order to build it we created an undirected and unweighted POI-POI graph denoted as $G_s = (V_s, E_s)$ where $V_s$and $E_s$ are the sets of POIs and edges. Each pair of POIs has adjacency only if their Harvesine distance is less than 5 km.

**Temporal:** In order to create Temporal graphs, for each user we create a directed and unweighted POI-POI graph denoted as $G_t = (V_t, E_t)$ where $V_s$and $E_s$ are the sets of POIs ad edges. We first partition the 10-month span into 49 time slots (one per week) and map each POI to its corresponding time slot. In every time slot, we order POIs ascending (time) order and each pair of POIs has adjacency if $time(POI_i) < time(POI_j) \wedge j = i + 1$. In this way, each user would have at least one temporal graph.

Since we used LSTMs, we converted the Temporal Graph into a sequence and we added padding in order to make all the sequences of the same lenght.

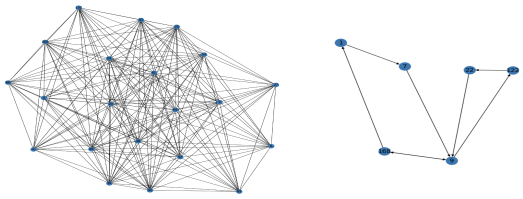

Figure 2: Example of Spatial and Temporal Graphs

### 4.1.2 History

As we wanted to work with users histories, we built a frequency list for each user that represent their history. These lists have the same lenght as the number of total POIs, therefore each position of the list correspond to a particular POI and its value is the number of visits the user has made in that POI. The next step was to normalize the frequency lists in order to have uniform data.

In addition to these steps, we performed a dimensionality reduction (PCA) also to better visualize the distribution of users' histories and choosing the right number of clusters.

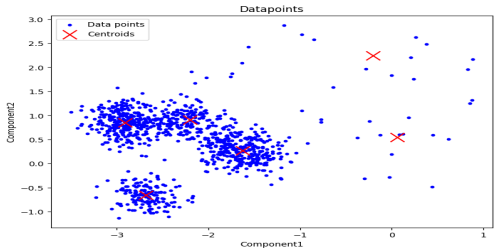


Figure 3: Points distribution and K-Means

#### 4.1.3 Labels

Each user u has its own labels. We started building them by sorting in descending order the POIs that u has visited, sorted from the most to the least visited, then we take the top 20 of them and build a one-hot list for each user where position p has value 1 if the relative POI is in user TOP 20, 0 otherwise. For evaluating Acc@K4.4, we use the original ranking labels to assess how well the model ranks the most visited POIs for each user. Along with ranking labels we used one-hot representation to compute the loss during training.

### 4.2 Training

We trained our model utilizing the Pytorch Lightning framework and within the limits of Google Colab free tier, therefore we had to come up with strategies in order to overcome the problem of limited resources.

We limited the dataset to 1000 users, taking into account the one who had at least 20 POIs (for labels 4.1.3 and Acc@20 4.4) trained our model for a maximum of 100 epochs and used early stopping with a patience value of 10.

We also used a batch size of 64, Adam optimizer with a learning rate of 0.001, dropout with probabilty 0.3 both temporal and spatial embeddings had a dimension of 100, while cluster embedding dimension was 25. This values were overall the best among the tested ones (Table 1).

| | Value 1 | Value 2 | Value 3 |
|---|---|---|---|
| Batch size | 32 | **64** | 128 |
| Learning Rate | 0.01 | **0.001** | 0.0001 |
| Temporal/Spatial Embeddings Dimension | 50 | **100** | 150 |
| Cluster Embeddings Dimension | **25** | 50 | 100 |
| Dropout Probability | 0.2 | **0.3** | 0.5 |

Table 1: Hyperparameter tuning

### 4.3 Loss

In this context, dealing with a large number of POIs presents a challenge: each POI prediction tends to have very low probabilities, causing a traditional cross-entropy loss to produce very high values. On the other hand, using one-hot labels results in a low loss value with a binary cross entropy, as most of the label elements are zeros. To address this imbalance, we decided to combine both losses as a weighted sum: $\mathcal{L} = \alpha\mathcal{L}_{bce} + \beta\mathcal{L}_{ce}$ with $\alpha = 0.85$, $\beta = 0.15$. Binary cross-entropy helps in managing the sparsity of the one-hot labels, while cross-entropy ensures that the model still accounts for the many low-probability predictions. This combination provides a more balanced loss function that better reflects the nature of the task.

### 4.4 Testing and Metrics

To test our model performances we utilized as metrics Accuracy@K and Mean Reciprocal Rank (MRR), which together provide a robust assessment of prediction quality.

**Accuracy@K:**

$$Acc@K = \frac{1}{N_{\text{test}}} \cdot \text{hit@K}$$

where hit@K is the number of samples with correct predictions made within the top $K$ of the ranked set for $K \in \{5, 10, 20\}$ and $N_{\text{test}}$ is the total number of test samples. It helps to understand the performance of the recommender system for the top $K$ recommendations.

**Mean Reciprocal Rank:**

$$MRR = \frac{1}{N_{test}} \cdot \sum_{v=1}^{N_{test}} \frac{1}{rank_v(l_{t_i})}$$

where $rank_v(l_{t_i})$ is the position of the ground truth next POI $l_{t_i}$ in the predicted ranked set for each $v$-th test sample. MRR gives an overall performance of the ranked set predicted.

# 5 Results and Conclusion

## 5.1 Results

The results shown in Table 2 and Figure 4 demonstrate a clear improvement in Next POI prediction performance as we progressively enhance the base architecture (our baseline is Spatial Graph processed with GATs). We initially experimented with both GAT and GCN models, but observed that GCNs performed slightly better, leading us to proceed with the GCN-based architecture. Starting with GNNs that utilize only spatial data, we observe incremental gains in accuracy and other evaluation metrics as we introduce additional components. Incorporating temporal data, user clustering, and embedding cluster centroids significantly boosts the model's effectiveness, validating the impact of these enhancements on capturing user preferences and mobility patterns more accurately, even slightly outperforming the state-of-the-art HMT-GRN proposed in [1].

| | ACC@1 | ACC@5 | ACC@10 | ACC@20 | MRR |
|---|---|---|---|---|---|
| GAT-SPATIAL | 0.121 | 0.227 | 0.286 | 0.338 | 0.182 |
| GCN-SPATIAL | 0.119 | 0.233 | 0.291 | 0.342 | 0.186 |
| GCN-SPATIAL+LSTM-TEMPORAL | 0.141 | 0.259 | 0.327 | 0.386 | 0.202 |
| GCN-SPATIAL+LSTM-TEMPORAL+GCN-SIMILAR | 0.163 | 0.346 | 0.457 | 0.523 | 0.243 |
| GCN-SPATIAL+LSTM-TEMPORAL+GCN-SIMILAR+CENTROID | **0.171** | **0.383** | **0.478** | **0.572** | **0.264** |
| HMT-GRN[1] | 0.167 | 0.335 | 0.414 | 0.498 | 0.251 |

Table 2: Caption

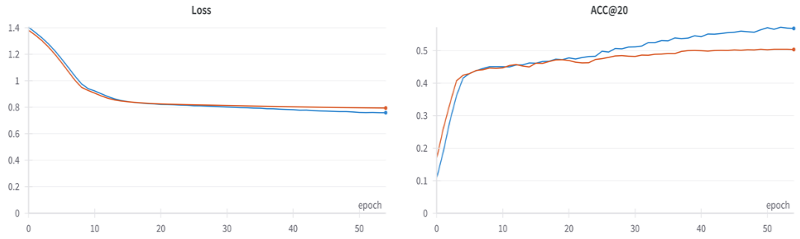


Figure 4: Plot of the our final model best run

## 5.2 Conclusion

In conclusion, our enhanced model demonstrates improved accuracy in Next POI prediction by integrating user clustering and advanced embedding techniques, showing that it can compete with state-of-the-art approaches[1]. However, there is room for further exploration. Future work could involve experimenting with different clustering methods, such as Gaussian Mixture Models (GMM), and scaling the model to leverage more computational resources, enabling deeper insights into user mobility patterns and further improving recommendation quality.

# References

[1] Nicholas Lim, Bryan Hooi, See-Kiong Ng, Yong Liang Goh, Renrong Weng, and Rui Tan. Hierarchical multi-task graph recurrent network for next poi recommendation. *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2022.