REPORT - GRAPH NEURAL NETWORKS FOR NEXT POINT OF INTEREST RECOMMENDATION

Adriano Izzi 2048338

Gavriel Di Nepi 2067753

Jacopo Fabi 1809860

ABSTRACT

This report addresses the "Next Point of Interest" (Next-POI) problem, detailing our approach and implementation. The Next-POI problem involves predicting a user's next likely destination based on their current position and past movements. Our implementation leverages both spatial and temporal data to enhance forecasting accuracy. Specifically, we compare a baseline model that utilizes only temporal information with an model integrating both spatial and temporal data through a graph neural network (GNN). Our findings demonstrate that incorporating temporal-spatial information and accurate data analysis significantly improves model performance, underscoring its importance in accurate location prediction.

1 Dataset

The dataset comprises 227,428 check-ins from 1,083 users at 38,333 venues across 398 distinct categories, recorded between 2012 and 2013 in New York City. The data preparation stems from insights coming form various papers.

1.1 Data filtering

Our study began with an analysis of the dataset to identify patterns and extract relevant information that could enhance the model's ability to predict the next point of interest. Notably, we found that out of 38,333 venues, 33,198 were visited fewer than 10 times, leaving only 5.135 venues with a higher frequency of visits. This insight highlights a significant concentration of user activity in a limited subset of popular locations, which plays a crucial role in refining our model's predictive capabilities.

In general, we will name a model as MODEL-INTEGER to signal the number of visits that a label need to not be cut off during training and test.

So a MODEL-10 has been trained using only sequences with labels that have received at least 10 visits.

MODEL-ALL, instead, would be a model trained and tested on all sequences regardless of the visits the labels have received.

1.2 Grid and Hotness

To improve the model, we have created two new features: *hotness* and *grid*. Hotness represents the popularity of a location, measured on a scale from 0 to 5, based on the number of times the place has been visited.

The second feature, grid, is derived from the latitude and longitude coordinates of venues, dividing New York into 336 grid sections. This feature provides additional spatial information about the relative proximity of places, supplementing the movement patterns of visitors.

2 Sequence's creation

One of the key features in our analysis is the timestamp of each check-in. Predicting the next point of interest tends to be relatively straightforward when the request is close in time to the last check-in. However, the task becomes significantly

more challenging as the time gap between the last check-in and the target prediction increases, as users may have traveled further afield, making it harder to rely solely on location history. To address this, we created sequences with a maximum time gap of one day between check-ins, thus generating multiple sequences of varying lengths for each user while excluding isolated check-ins.

We found that among the 30% of the around 33,000 sequence, has a label that is one of the 20 most frequent one. This means that the model could achieve good accuracy simply by predicting these, which mostly correspond to offices, apartments, bars, and gyms. Since this value was too high and could bias our system towards frequently predicting these same 20 places, we decided to make some changes based on how the system could be used. Since the system could be used at any time of the day, we decided to take a random point from each sequence and try to predict it. This approach helped to increase the variation among the labels.

We also thought that in a real environment we would always have an actual sequence to start our prediction, and not only 1 or 2 places. Hence, we decided to drop all sequences that were shorter than 4 venues. As a byproduct, we also noticed that we eliminated an enormous bias in the data, since it seems that the vast majority of the sequences were costituted by 1 values only (exluding the label) and that those data were extremely skewed. Upon analyzing the venue distribution, we noticed that it was particularly favorable to the training to focus the attention on the most visited venues. As a result, we decided to keep these venues but to remove sequences with labels different from the main 5,135 venues, resulting in a total of 7,087 different sequences.

In general, we will name a sequence set as SEQUENCE-WRL to signal that the label has been chosen at random inside the original sequence (and that the sequence has been cut-off up to that venue). So, our final dateset, SEQUENCE-10-WRL, contains only sequences that ends with labels that have received at least 10 visits and that is has been chose at random inside a contiguous streak of visits.

The same denomination is valid for models too. So MODEL-10-WRL is a model that has been trained with the SEQUENCE-10-WRL.

Table 1: Frequency of Labels

Model	**Top-1**	**Top-5**	**Top-10**	**Top-20**
Sequence-10-WRL	1.38	4.79	7.86	11.74
Sequence-ALL-WRL	0.57	2.23	3.78	5.76

Table 2: Frequency of most popular labels if they were directly put in top1, top5, top10, and top20

3 Baseline LSTM

Our baseline approach aimed to solve the task using only temporal information within sequences. To achieve this, we implemented a Long Short-Term Memory (LSTM) model that processes sequences of venue IDs, grids, category, and hotness values. Each feature is individually embedded and then concatenated, forming a single embedding that encapsulates all relevant information for a given venue.

This concatenation is then traversed by a single Linear Layer which compute a projection going back to the desired embedding dimension.

The embedding is then fed into an LSTM layer, which produces a hidden state. The hidden state is then passed through two projection layers to generate logits, yielding a probability distribution across all possible venues.

4 Graph approach

4.1 Sequence to graph

For each sequence, we generated a corresponding graph representation. To manage the graph's size, we set the number of nodes equal to the maximum sequence length. Each node represents a check-in, with an edge from $node_i$ to $node_j$ if $node_i$ occurs before $node_j$ in the sequence. Each node contains four features: venue ID, grid, category, and hotness. The sequence order is indicated by the node index, where the first node, $node_0$, corresponds to the most recently visited venue and thus has no outgoing edges.

Figure 1 provides an example. As shown, we obtain a graph with connected nodes representing the sequence, while all other nodes remain disconnected and will not participate in the convolution.

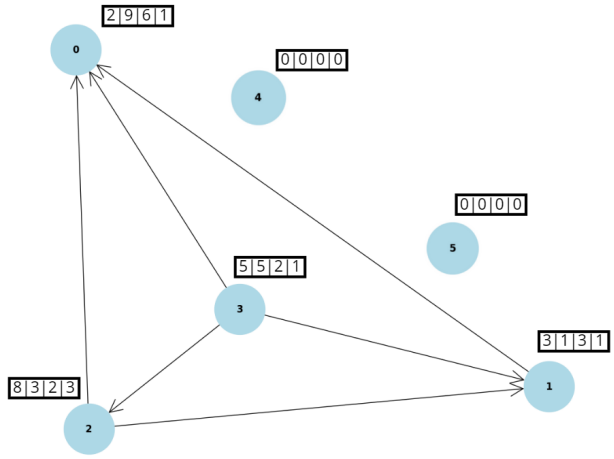

Figure 1: Graph representation of the 'venues-id' sequence [5, 8, 3, 2] with a maximum sequence length of six. Each node in the graph represents a position in the sequence, with the first value being the sequence ID and the remaining three values representing node features. Nodes that exceed the length of the original sequence are padded with zeros to ensure uniform length across all features.

This graph structure capture the task, which is to derive the next POI from a sequence of places visited in a streak.

4.2 Model

The *GCNSequence_LIGHT* model is a graph neural network that takes as input the graph representing a sequence and computes the probability distribution over the classes.

Specifically, it embeds each node's features into a larger latent space, then concatenates these embeddings, resulting in a tensor that is four times the size of the original embedded dimension. The concatenated tensor is then transformed to the hidden dimension, and a graph convolution operation is applied, producing a feature representation for each node with a dimensionality corresponding to the number of classes to predict. After information exchange occurs between the nodes, the first node, $node_0$, which represents the last node visited, is used to make the prediction.

5 Metrics

The model has been evaluated with different metrics, since the accuracy couldn't describe accurately the precision of the model. The metrics used for the evaluation are the Top 1, 5, 10 and 20, so the number of times that the correct prediction is in the first top-k position, and MRR, the Mean Reciprocal Rank.

6 Training

All models were trained using the early stopping technique, with the MRR (Mean Reciprocal Rank) metric used to determine improvement from one epoch to the next. We chose a low patience value of 2 because, during training, we observed that the model tended to reach its final performance metrics without double descents.

6.1 Hyper-parameters

During training, we conducted hyper-parameter tuning, resulting in the best hyper-parameters listed in Table 3. After finding these values, we applied the same number of embeddings and methodology to the baseline model to ensure that the same set of information was considered.

Hyperparameter	Value
Learning Rate	0.0001
Batch Size	128
Optimizer	Adam
Loss Function	Cross Entropy
Dropout Rate	0.25
Units per Hidden Layer	512
Activation Function	Relu, Tanh

Table 3: Hyperparameters for the Neural Network Model

7 Evaluation

The results of the model and the baselines are presented in Table 4. As we can see, the model's performance surpasses that of the baselines across all metrics, primarily because the convolution allows each venue to incorporate information from the connected ones.

We also trained MODEL-ALL-WRL (so the training, validation and test sets were slightly different but similar in size) and the Mean Reciprocal Rank was 3 points below MODEL-10-WRL. This demonstrate that using all labels to predict the Next POI lead to poorer performances.

Table 4: Performance Metrics for Different Models

Model	Top-1	Top-5	Top-10	Top-20	MRR
U-TOP-10-WRL	4.13	9.95	12.00	13.42	n/d
LSTM	7.33	14.10	16.64	19.46	10.47
Model-10-WRL	7.75	19.88	24.68	29.05	13.45

8 Conclusion

Starting from the modern literature, we further delved into analysis and manipulation of the original dataset in order to capture the task, which is the one of an user moving trough the city in a limited time frame and with a sequence of (not too few) places already visited.

After such analysis and manipulation, we demonstrated that our model is capable of generalize the sequences better than a simple baseline like U-TOP or an LSTM.