Graph-Attention integrated LSTM for next Point of Interest Recommendation with Geospatial-Tree Search

Deep Learning 2023/2024 Project Report

Francesco Danese
1926188
La Sapienza University of Rome
danese.1926188@studenti.uniroma1.it

Abstract

The following work builds upon the methods presented in the suggested reference paper, with some modifications and a complete code re-implementation from scratch. Predicting the next Point-of-Interest (POI) a user will visit is a complex task for personalized recommender systems due to the vast number of potential venues in the given area. To address this challenge, the Hierarchical Multi-Task Graph Recurrent Network (HMT-GRN) approach is employed to incorporate user-personalization and enhance prediction accuracy. This approach leverages a Hierarchical Beam Search (HBS) on the model's predictions across nested geographic regions, progressively refining the search space with increasing spatial detail to accurately predict the next POI. This method enhances efficiency, achieving significant speedups compared to exhaustive search methods. The model integrates an LSTM with a Graph Attention Network to account for spatio-temporal relationships between previously visited POIs. A novel selectivity layer is also introduced based on whether the predicted next POI is a location the user has visited before, balancing the need for personalization with the desire for exploration.

1 Dataset preprocessing

The initial dataset includes the following relevant features: *user-id, venue-id, latitude, longitude, timestamp*. Each row represents a visit of a specific user to a particular venue, located at the given global coordinates, occurred at the provided date and time. The model input is a sequence of POIs associated with its unique user. Leveraging the GeoHash API and the coordinates of each venue, we associated each venue with its corresponding geocells at precisions of 2, 3, 4, 5, and 6. A geohash cell is a rectangular region on the Earth's surface, encoded as a string of characters. The precision of a geohash determines the size of the cell: higher precision results in smaller, more specific regions, while lower precision results in larger, less specific regions. For a given venue, each higher-precision (smaller) geocell is nested within the corresponding lower-precision (larger) geocell. By using varying levels of geohash precision, we can analyze spatial data at different granularities. We removed users (sequences) with less than 20 or more than 50 visits, as well as venues visited by less than 10 users. The cleaned dataset has then a total of 4455 POIs distributed across 16636 sequences.

2 POI-POI graphs

We define the Spatial Graph SG as an undirected and unweighted graph with the set of POIs as vertices, and an edge between two POIs if they are within the same G@4 cell. Consequently, we[1] partition each day to 8 time slots of 3hrs each, with a total of 56 time slots, then map each visit in the dataset to its corresponding time slot, such that each POI will have a set of mapped time slots τ_p; we finally build the Temporal Graph TG with the set of POIs as vertices, and an edge between two POIs $(p_i,\, p_j)$ if their time slot sets have a Jaccard Similarity $\frac{|\tau_{p_i} \cap \tau_{p_j}|}{|\tau_{p_i} \cup \tau_{p_j}|}$ greater than 0.9.

[1]For the sake of style and formality throughout this report, the first-person plural pronoun "we" is used, even though the work was carried out by a single individual, myself.

3 Problem formulation

Given as input the tuple $SEQ_u = \langle u,\, S_u,\, G@2,\, ...,\, G@6,\, SG,\, TG \rangle$ where: u is a specific user; $S = (p_1, p_2, ..., p_t)$ is the sequence of POIs visited by u sorted in chronological order; $G@k = (g@k_1, g@k_2, ..., g@k_t)$ is the sequence of GeoHash cells of precision k containing the corresponding POI at time t; SG and TG are the fixed Spatial and Temporal Graphs; $\rightarrow$ the goal is to create a ranked set of POIs where the next visited p_{t+1} by user u is highly ranked.

4 Geo-Hierarchical Tree

A major improvement to the recommendation system is the creation of a "Geo-Hierarchical Tree", or "GeoTree", for short. The GeoTree is a structured representation of geographical points of interest (POIs) using geohash cells at varying levels of precision. This tree organizes POIs and geocells (geographical cells) in a hierarchical manner, where the hierarchy is based on the precision of the geohash cells, as shown in Fig.1 below. Each node in the tree represents a geocell. Nodes at a particular level represent geocells of the same precision. Nodes at a higher precision level (e.g. G@3) are children of nodes at the immediate lower precision level (e.g. G@2) if they are geographically contained within the parent node's area. This continues down the hierarchy until the highest precision level, where the leaves of the tree represent individual POIs.

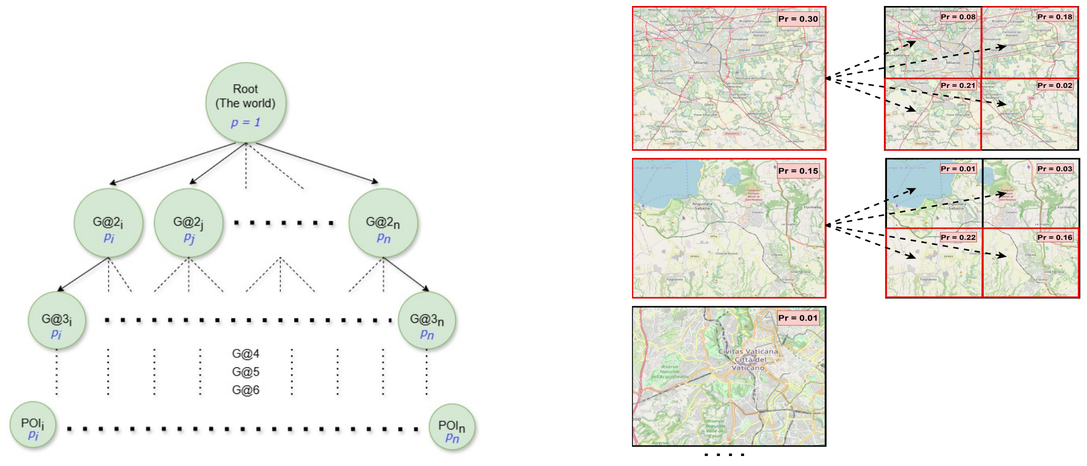


Figure 1: Geo-Hierarchical Tree: the model will predict, independently, the probability p for each node to be the next visited geocell (or POI) in a multitask setting. The joint probability for a POI i is the product of the probabilities of all nodes in the path from root to POI i. On the right, a step of Beam Search algorithm with Beam Width = 2.

5 Baseline: HMT-RN

Preliminaries: An LSTM (Long Short-Term Memory) is a type of recurrent neural network (RNN) architecture that is particularly effective at learning and remembering over long sequences. It is designed to combat the vanishing gradient problem, allowing it to maintain and learn dependencies over extended periods of time. LSTMs achieve this by using a set of gates to control the flow of information into and out of its memory cell.

$$\begin{aligned} i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) & \tilde{c}_t &= \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \\ f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) & c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \\ o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) & h_t &= o_t \odot \tanh(c_t) \end{aligned}$$

where x_t represents the input at time step t, while $[a, b]$ indicates concatenation and $\odot$ element-wise multiplication.

Learning next POI and GeoCell distributions: The idea is to predict not only the next POI, but also the next regions ($G@k$) where the next POI resides in, denoting $k \in \{2, 3, 4, 5, 6\}$ as all the precision levels to be considered in the model and $TK = \{G@2, G@3, G@4, G@5, G@6, POI\}$ as all the tasks to be learned. Given an input SEQ and a time step t, we first map the user u, the previous visited POI p_{t-1} and each of its corresponding geocell $g@k_{t-1}$ to a vector representation, by using a trainable embedding layer:

$$\vec{u},\, \vec{p_{t-1}},\, g@\vec{k}_{t-1} = E_{\mathbf{W}}(u,\, p_{t-1},\, g@k_{t-1})$$

$$\mathbf{W} = \{\mathbf{W}_u \in \Re^{|U| \times \epsilon}, \mathbf{W}_p \in \Re^{|V| \times \epsilon}, \mathbf{W}_{g@k} \in \Re^{|V^{g@k}| \times \epsilon}\}$$

Where $\mathbf{W}$ is the collection of weight matrices holding the embeddings for each entity; ϵ is the embedding dimension, U and V are the set containing all users and POIs in the dataset respectively, while $V^{g@k}$ is the set of all geocell of precision k after mapping all venues in V to their corresponding GeoHash cell. Next, an LSTM layer is used to learn the sequential dependencies among the POI sequence, with $\vec{p_{t-1}}$ as input:

$$\vec{h_t} = LSTM(\vec{p_{t-1}})$$

where we assume the previous hidden state $\vec{h_{t-1}}$, (that the model manipulates together with the input in order to compute the next state) to be computed similarly. Finally, we compute the probabilities of the next POI distribution as:

$$Pr(p_t | p_{t-1}, u) = Softmax(\, FC_{\mathbf{W}_l}(\, DropOut(\, [\vec{h_t},\, \vec{u}]\,)\,)\,)$$

with FC as a fully connected linear layer parameterized by $\mathbf{W}_l \in \Re^{hdim+\epsilon \times |V|}$ to project to $|V|$ POIs. Those logits are finally passed to the softmax function for producing the conditional probability distribution over the set of POIs. We can then sort all POIs in V, according to their predicted probability in descending order, to retrieve the *ranked set*. Together with the next POI task, the model learns, in a multitask fashion, to also predict the next $G@k$, that is the GeoHash cell of precision k where the next POI p_t is located. This is done for all $k \in [2, 3, 4, 5, 6]$, using the same $\vec{h_t}$ respresentation produced by the LSTM layer, but instead of coupling it with the user embedding $\vec{u}$, we concatenate the embedding of the previous visited geocell $g@k_{t-1}$. This means considering the grid cell of a specific precision associated to the previous visited POI to compute the probability of the next grid cell at the same precision:

$$Pr(g@k_t | p_{t-1}, g@k_{t-1}) = Softmax(\, FC_{\mathbf{W}_{l@k}}(\, DropOut(\, [\vec{h_t},\, g@\vec{k}_{t-1}]\,)\,)\,)$$
$$\forall k \in [2, 3, 4, 5, 6]$$

We call this overall model **HMT-RN** (Hierarchical Multi-Task Recurrent Network), and it will serve as primary baseline.

6 GeoTree search strategies

We use the predicted probabilities of $g@k_t$ and p_t to fill the levels of the GeoTree in Sec. 4. Then, we need to compute the joint probability $Pr(g@2_t, g@3_t..., g@6_t, p_t)$ to rank all POIs and predict the most likely one. We can perform an exhaustive search, such as Breadth-first search (BFS), to compute the sums of log-probabilities by traversing all paths of the root to each leaf (POI). Alternatively, we can run a Beam Search heuristic algorithm, that explores a graph by expanding the most promising nodes, in our case the ones with highest marginal probability. It reduces memory consumption by only keeping a fixed number of best candidates at each level, known as the beam width, while also increasing search speed, as depicted in Fig. 1. When the beam width is set to the maximum (i.e the level width), it behaves like BFS.

Selectivity layer Instead of performing the tree search at every inference, a selectivity layer choose whether to explore the search space (tree) or not. It does so by taking the next predicted POI $\hat{p_t} = argmax(Pr(p_t|p_{t-1}, u))$ and returning:

$$y_t = \begin{cases} Pr(p_t|p_{t-1}, u) & \text{if } \hat{p_t} \in S_u^{train} \\ Pr(g@2_t, g@3_t..., g@6_t, p_t) & \text{otherwise} \end{cases}$$

where each user's POI sequence is divided into S_u^{train} using the first 80% POIs and S_u^{test} using the last 20%.

7 HMT-GRN

To incorporate POI-POI spatio-temporal relationships, we use a Dimensional Graph Attention Network (DGAT) module:

$$\vec{\alpha}_{j,t-1} = \frac{exp(LeakyReLU(\mathbf{W}_a[\mathbf{W}_p\vec{p_{t-1}},\, \mathbf{W}_p\vec{j}]))}{\sum_{\vec{k}\in N[p_{t-1}]} exp(LeakyReLU(\mathbf{W}_a[\mathbf{W}_p\vec{p_{t-1}},\, \mathbf{W}_p\vec{k}]))} \qquad \text{and} \qquad \vec{n_{t-1}} = \sum_{\vec{j}\in N[p_{t-1}]} \vec{\alpha}_{j,t-1} \odot \mathbf{W}_p\vec{p_{t-1}}$$

where $\mathbf{W}_a \in \Re^{2 \cdot hdim \times hdim}$ predicts the attention weights between each dimension of the hidden representation for the current node-POI p_t and one of its neighbors j, subject to a softmax over all neighbors in $N[p_{t-1}]$. The predicted weights $\vec{\alpha}_{j,t-1}$ are then applied to compute a weighted sum of its respective neighbors, outputting the updated node representation $\vec{n_{t-1}}$. Then, the LSTM of the baseline (Sec. 5) is extended by concatenating its input with the two node representations produce by the DGAT module for the Spatial and Temporal Graph, namely SG and TG (Sec. 2):

$$\vec{h_t} = LSTM([\vec{p_{t-1}},\, n_{t-1}^{\vec{SG}},\, n_{t-1}^{\vec{TG}}])$$

We call this model **HMT-GRN** (Hierarchical Multi-Task Graph Recurrent Network) and it will be compared to the HMT-RN baseline.

Training For each task in $TK = \{G@2, ..., G@6, POI\}$ we treat the job as a classification problem, employing the standard Cross-Entropy (CE) loss $\mathcal{L}_{TK} = -\sum_{t=1}^{T} log(\hat{pr}_t)$ where $\hat{pr}_t$ is the predicted probability of the ground truth next POI or cell at time t, summed across all time steps in the sequence. For $G@2$ and $G@3$ tasks, instead of the plain CE loss, we compute the Focal Loss $FL(p_t) = -(1 - \hat{pr}_t)^{\gamma} \log(\hat{pr}_t)$, proved to be useful in case of strong class imbalance as in the case of precision 2 and 3 cell distribution, as shown in Fig. 2. The final Multi-Task objective to be optimized is $\mathcal{L} = \frac{\mathcal{L}_{TK}}{|TK|}$, the mean of the losses of each task. We used the Adam optimizer, 25 epochs with early stopping (based on accuracy on the direct POI task), a batch size of 32, a learning rate of 0.0001, beam width = 100 for beam search, and set the dropout rate to be 0.5. Then, set the POI, user, geocell embedding dimension ϵ and hidden dimension *hdim* to be the same of 512. Since the HMT-GRN model is inherently larger than the baseline, due to the addition of the DGAT module, we set the number of lstm layers of just the HMT-RN to 2. This is done to mitigate the performance differences caused by just a greater number of parameters rather than the diverse architecture.

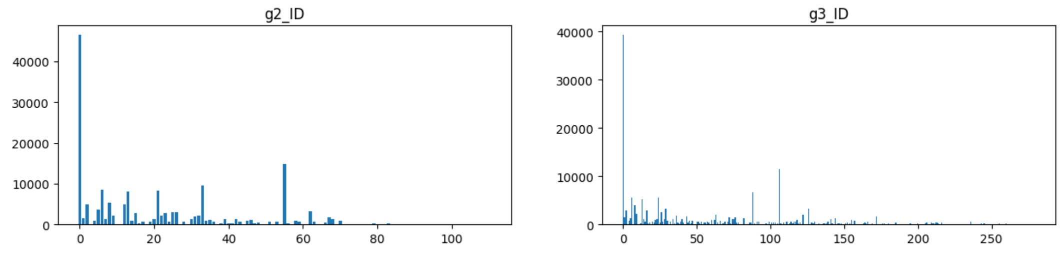


Figure 2: Distribution of GeoCells of precision 2 and 3. Since those are the largest geographical grids, a large amount of POIs is concentrated on few GeoCells, possibly causing class imbalance issues when learning $G@2$ and $G@3$ tasks.

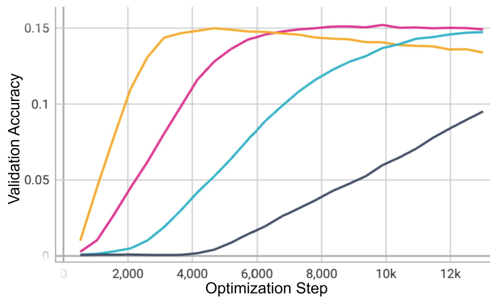
8 Results

Metrics We use the common metric of Acc@K = $\frac{\#hit@K}{\#test}$ where $\#hit@K$ is the number of samples with the correct predictions made within the top K of the ranked set for $K \in \{1, 5, 10, 20\}$, and *#test* is the total number of test samples. We also evaluate for the metric of Mean Reciprocal Rank, $MRR = \frac{1}{\#test} \sum_{i}^{\#test} \frac{1}{rank_i(p_t)}$ where $rank_i(p_t)$ is the position of the ground truth next POI in the predicted ranked set for each i-th test sample. Table 1 shows the performance comparison of the two models, with the selectivity layer activated and a beam width (β) = 100 for the Beam Search on the Geo-Hierarchical Tree. Only for the MRR metric, BFS was applied instead of Beam Search, since the latter iteratively "prunes" the search space, including the last level with the POI estimated joint probabilities. It therefore outputs a ranked set of size equals to β, the top-β solutions of the entire set V. Since the real visited POI p_t could be outside this limited set, the value of $rank_i(p_t)$ would be ambiguous. The HMT-GRN model outperformed the baseline across all metrics, as shown by the relative improvement in the last row. This demonstrates the significant advantage of integrating spatio-temporal POI-POI relationships, and highlights the DGAT module's capability to extract key features.

Table 1: Performance in Acc@k and MRR for all next POI test samples.

Type	Acc@1	Acc@5	Acc@10	Acc@20	MRR
HMT-RN	0.1477	0.2930	0.3656	0.4323	0.2202
HMT-GRN	**0.1549**	**0.3154**	**0.3875**	**0.4589**	**0.2334**
Improvement	4.87%	7.65%	5.99%	6.15%	6.00%

Hyperparameter search For simplicity, we set the embedding size and hidden state dimension to be the same in every run. We trained and validated the model testing each value of $hdim \wedge \epsilon \in [128, 256, 512, 1024]$. A model checkpoint was saved for each choice, when the network reached its maximum accuracy computed *directly* on the POI task, with no tree-search due to time and computational limitations. Another checkpoint was retrieved when the model reached the lowest multi-task loss, to check if the hierarchical search could subsequently benefit from a overall better estimation of all tasks. This actually resulted in a slightly better top-1 accuracy, but with a great worsening in MRR and top-5, 10 and 20 accuracy. Hence, we chose to retrieve the best model, with size = 512, according to the first method, and to report the metric results after performing the tree-search. Fig. 3 plots the performance of GRN for various values of $hdim$ and ϵ, showing validation accuracy and loss throughout training for each hyperparameter. It indicates potential overfitting for the highest value of 1024, despite it being the fastest to optimize, as it is eventually outperformed by 512 at their respective peaks. Finally, the dropout probability was consistently set to 0.5 in all experiments.



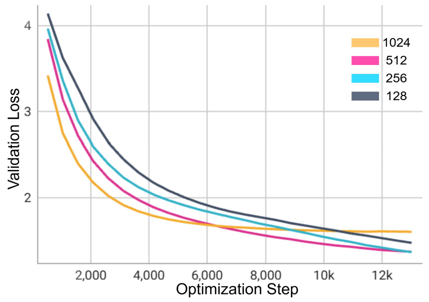


Figure 3: Validation loss and accuracy at each training step, for different $hdim$ and ϵ sizes. The accuracy is computed only on the raw POI task, while the loss is multitask. The hyperparameter 512 was selected and used to compute the final metrics adding the selectivity layer and follow-up Beam Search to consider all task predictions.

Search methods We examined the effect of reducing the search space on system performance for the HMT-GRN model. Figure 4 illustrates how varying the Beam Width β influences accuracy, peaking at a value of 10, which matches BFS, while significantly lowering memory and time usage by keeping track of only the top-β nodes instead of thousands of geocells. For the final evaluations, β was set to 100, as a larger beam could improve the set of predictions (top-k) without any noticeable slowdown. Moreover, we analyzed the importance of the Selectivity Layer by comparing metrics under two conditions: when it is deactivated, causing the Beam Search to always be performed indiscriminately, and when it is activated, limiting the search to cases where the predicted next POI was not in the user's training sequence. As the layer is deactivated, we notice in Fig. 5 a clear drop in performance. If the predicted next POI is not in S_u^{train}, the model based the prediction on POI sequential relations found in other users' sequences, lacking personalization. In such cases, it is more effective to conduct the search along the tree that encapsulates predictions for nested geographical areas specific to the user, enabling better exploration.

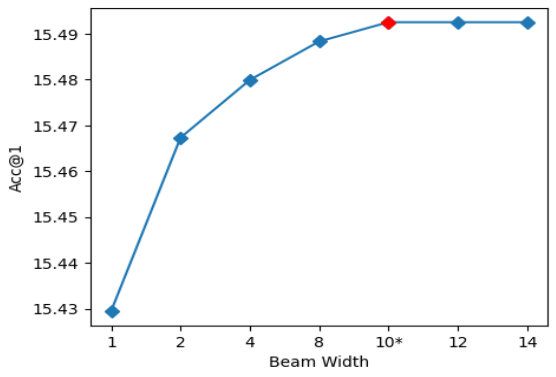


Figure 4: Acc@1 on raw POI task at varying β

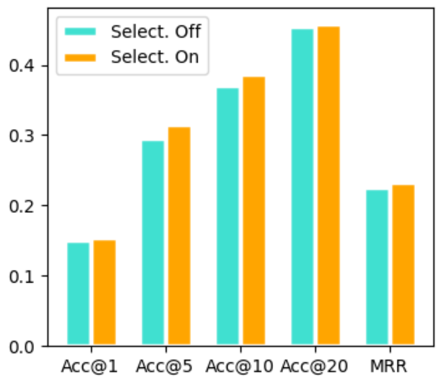


Figure 5: Impact of Selectivity Layer on different metrics

9 Conclusion

This study presents the Hierarchical Multi-Task Graph Recurrent Network (HMT-GRN), a novel approach for next Point-of-Interest (POI) recommendation that integrates spatio-temporal relationships using Graph Attention Networks. By leveraging a hierarchical tree structure with geohash cells of varying precision and employing a beam search algorithm, the model significantly improves the efficiency and accuracy of POI predictions compared to traditional methods. The inclusion of a selectivity layer further optimizes the recommendation process by balancing exploration and personalization. Experimental results demonstrate that HMT-GRN outperforms the baseline HMT-RN model in all evaluated metrics, underscoring the effectiveness of incorporating spatio-temporal dynamics into the recommendation framework. Future work could explore further optimizations and extensions of the model, including real-time adaptability and scalability to larger datasets.