GRAPH NEURAL NETWORKS FOR NEXT POINT OF INTEREST RECOMMENDATION

Dario Loi
1940849

Elena Muià
1938610

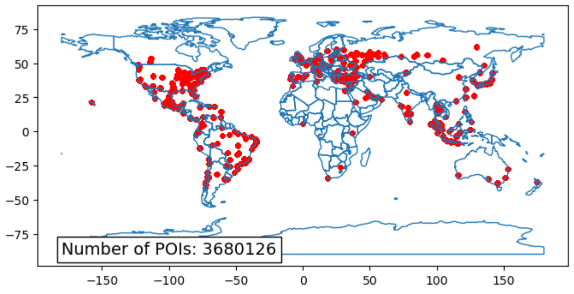
Martina Doku
1938629

ABSTRACT

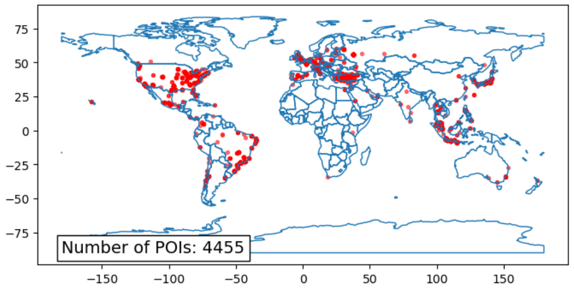
A POI (Point of Interest) recommendation technique essentially exploits users' historical check-ins and other multi-modal information such as POI attributes and friendship network, to recommend the next set of POIs suitable for a user. A POI recommendation technique essentially exploits users' historical check-ins and other multimodal information to recommend the next set of POIs suitable for a user.

1 Introduction

This work is a re-implementation of the paper "Hierarchical Multi-Task Graph Recurrent Network for Next POI Recommendation" [1]. The paper proposes a novel Graph Neural Network (GNN) model for the task of next Point of Interest (POI) recommendation, this model builds upon previous graph attention networks[2] by taking into account different levels of spatial granularity through the use of geo-hash grid codes. The model is designed to capture the complex relationships between users and POIs, and to leverage the rich information available in the user-POI interaction graph. The proposed model is evaluated on a real-world dataset and compared with a state-of-the-art baseline. The experimental results show that the proposed model outperforms the baseline and achieves the paper performance.



(a) Original Points of Interest on the world map



(b) Remaining Points of Interest after dataset pruning

Figure 1: Comparison of Points of Interest on the world map before (fig. 1a) and after (fig. 1b) preprocessing

2 Data Preprocessing

2.1 Dataset

The dataset used in this work is the Foursquare dataset[3]. It includes long-term (about 18 months from April 2012 to September 2013) global-scale check-in data collected from Foursquare. It contains 33,278,683 checkins by 266,909 users on 3,680,126 venues (in 415 cities in 77 countries). Those 415 cities are the most checked 415 cities by Foursquare users in the world, each of which contains at least 10K check-ins.

2.2 Data Preprocessing

The dataset is preprocessed to filter out irrelevant data and to prepare it for training the models. The preprocessing follows the same steps as in the original paper, which are:

- Filter out users with visit counts less than 20 and more than 50

Table 1: Number of POIs

Before Preprocessing	After Preprocessing
$3,680,126$	$4,455$

- Remove POIs visited by less than 10 users
- Sort the timestamps in chronological order
- Divide the data into training and testing sets keeping the first 80% of the visits for training and the last 20% for testing

The number of POIs before and after processing matches exactly the numbers reported in the original paper, as shown in table 1.

2.3 Graph Construction

The GNN model proposed in the paper leverages two types of graphs: spatial and temporal, that have been constructed as reported in the following sections.

2.3.1 Spatial Graph

The spatial graph encodes the relationships between different points of interest. It is constructed based on the physical or functional proximity of POIs, by considering the geo-hash codes of the POIs. The adjacency matrix for the spatial graph is constructed as follows:

$$a_{ij} = \begin{cases} 1 & \text{if } G@4(i) = G@4(j) \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

2.3.2 Temporal Graph

The temporal graph captures the sequential relationships between POIs. As done in the original paper, we divided the week in 56 time slots and assigned each check-in to the corresponding time slots v^{slot}. Then we constructed the adjacency matrix for the temporal graph as follows:

$$a_{ij} = \begin{cases} 1 & \text{if } \text{Jaccard}(v_i^{slot}, v_j^{slot}) > 0.9 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where $\text{Jaccard}(v_i^{slot}, v_j^{slot})$ is the Jaccard similarity between the two time slots:

$$\text{Jaccard}(v_i^{slot}, v_j^{slot}) = \frac{|v_i^{slot} \cap v_j^{slot}|}{|v_i^{slot} \cup v_j^{slot}|} \tag{3}$$

3 HMT-RN

The HMT_RN (Hierarchical MultiTask Recurrent Network) is a neural network architecture designed for multi-task learning. It employs embedding layers for users, POIs, and different geo-hash levels, effectively capturing the hierarchical nature of spatial data.

Central to the HMT_RN is an LSTM (Long ShortTerm Memory) layer, which processes sequences of embedded data, capturing temporal dependencies. The output from the LSTM is concatenated with the embeddings and passed through several linear layers, each responsible for predicting the next element in different hierarchical levels (e.g., POI, geo-hash levels 2 through 6). The model uses a dropout layer to mitigate overfitting and a criterion based on cross-entropy loss, applied selectively to handle padded sequences. Additionally, it incorporates accuracy metrics at various levels (Top-1, Top-5, Top-10, Top-20)[1] and Mean Reciprocal Rank (MRR) for performance evaluation.

4 GRN

The GRN (Graph Recurrent Network) is an attention based model, developed to leverage both spatial and temporal graph structures for enhanced predictive capabilities.

At its core, the GRN integrates graph attention mechanisms with recurrent layers to precisely process and forecast sequential data points. This amalgamation enables the model to effectively capture the intricate spatial and temporal relationships inherent in the data.

The model architecture is composed of multiple key components. Firstly, it employs graph self-attention mechanisms to dynamically weigh the importance of neighboring data points within both spatial and temporal contexts. Central to the GRN is an attentional LSTM (Long Short-Term Memory) layer, designed to handle sequences of embedded data while capturing temporal dependencies. Furthermore, the GRN incorporates various linear layers responsible for predicting the next element in the sequence across different hierarchical levels. These layers enable the model to forecast outcomes at multiple granularities, ranging from individual points of interest (POIs) to higher-order spatial structures represented by different geo-hash levels.

As with the previous model, the GRN integrates dropout regularization and uses cross-entropy loss as the optimization criterion.

5 Experiments

The training procedure utilized the PyTorch Lightning framework[4], augmented with the WandbLogger for real-time monitoring and logging of training metrics. The training dataset was split into train and test sets as detailed in section 2.2. We used a batch size of 2048 for the LSTM model and 128 for the GRN model, loading was parallelized across 4 workers for faster training. The GRN model was instantiated with specific hyperparameters, including a hidden dimension of 1024 and a dropout rate of

[1]Also called Accuracy@K in the paper[1]

0.9 as specified in the reference paper. Although not specified by the authors, we utilized Kaiming initialization[5] for the model weights to ensure stable training. The training process was configured for a maximum of 200 epochs and equipped with callbacks for dynamic adjustment of learning rates, model checkpointing, and early stopping based on validation loss.

Throughout the training process, careful attention was paid to memory management and optimization techniques. The experimental setup aimed to strike a balance between computational efficiency and model performance, laying the groundwork for robust experimentation and evaluation of the GRN architecture.

The values of the metrics during the training process are shown in figs. 2 to 6.

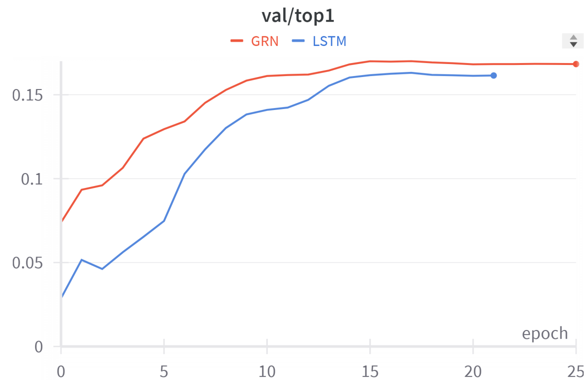


Figure 2: Top-1 accuracy

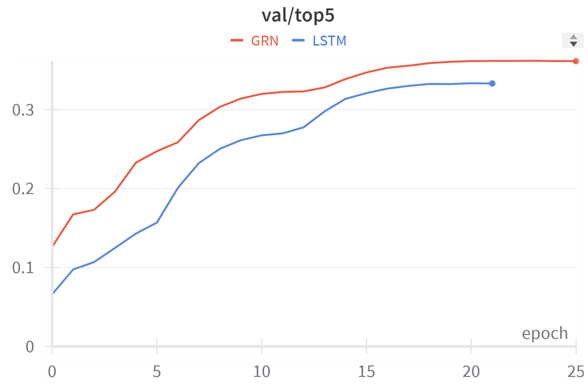


Figure 3: Top-5 accuracy

5.1 Hyperparameters tuning

In the process of training the Graph Neural Network (GNN) classifier, a systematic hyperparameter tuning procedure was employed to optimize the model's performance. Initially, the batch size of the classifier was scaled using the power scaling method provided by the `tuner.scale_batch_size` function, which adjusts the batch size to the most appropriate value for efficient training. Following this, the learning rate was fine-tuned using a learning rate finder, invoked through the `tuner.lr_find` function. This function explored a range of learning rates from 5×10^{-5} to 1×10^{-3} over 100 training iterations to identify the optimal learning rate that minimizes the loss.

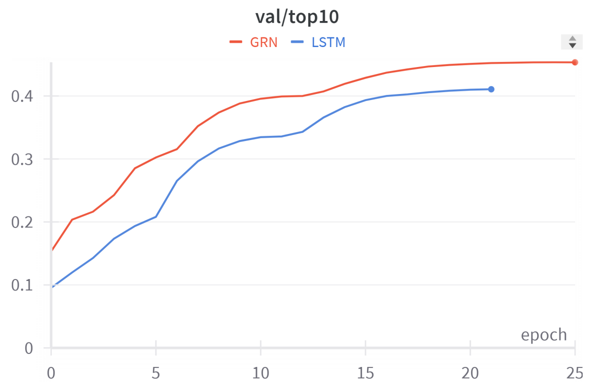


Figure 4: Top-10 accuracy

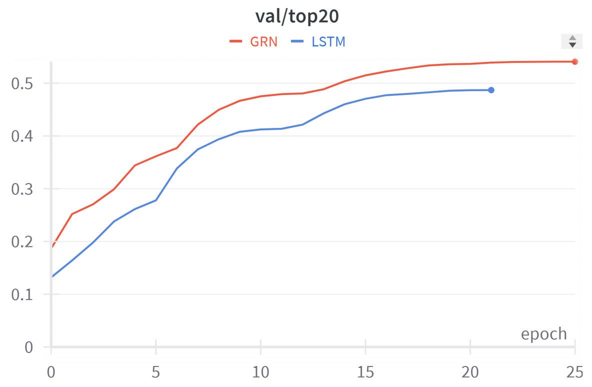


Figure 5: Top-20 accuracy

6 Results

6.1 Metrics

The performance of the models is evaluated using the following metrics:

- Top-1, Top-5, Top-10, Top-20 accuracy
- Mean Reciprocal Rank (MRR)

where the top-k accuracy is calculated as follows:

$$\text{Top-k accuracy} = \frac{1}{N} \sum_{i=1}^{N} \delta(i \in \text{Top-k}) \tag{4}$$

Table 2: Results on the Foursquare dataset, test set. The first two rows are the results from the original paper[1], the last two rows are the results from our implementation, the best metrics are highlighted in bold.

Name	Top-1	Top-5	Top-10	Top-20	MRR
HMT-RN[1]	0.143	0.267	0.321	0.378	0.205
GRN[1]	0.145	0.278	0.339	0.403	0.212
HMT-RN (ours)	**0.161**	**0.333**	**0.410**	**0.486**	**0.246**
GRN (ours)	**0.168**	**0.361**	**0.435**	**0.540**	**0.262**

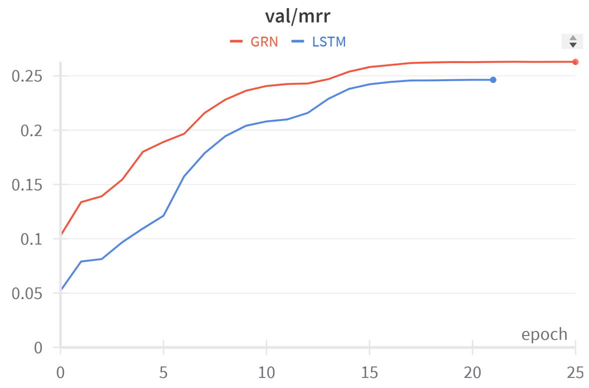


Figure 6: Mean Reciprocal Rank (MRR)

and the MRR is calculated as follows:

$$\text{MRR} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{\text{rank}_i} \tag{5}$$

where δ is the indicator function, N is the number of test samples, Top-k is the set of the k POIs with the highest predicted scores, and rank_i is the rank of the correct POI for the i-th test sample.

6.2 Results

The results obtained from the experiments are summarized in table 2. The models achieve similar performance to the original paper.

7 Conclusion

We have re-implemented the HMT-RN and GRN models for the task of next POI recommendation. The models were evaluated on the refence Foursquare dataset and compared with the original results. The results show that the models achieve similar performance to the original paper, demonstrating the effectiveness of the proposed approach for next POI recommendation.

References

[1] N. Lim, B. Hooi, S.-K. Ng, Y. L. Goh, R. Weng, and R. Tan, "Hierarchical multi-task graph recurrent network for next poi recommendation," in *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, p. 11331143, 2022.

[2] N. Lim, B. Hooi, S. Ng, X. Wang, Y. L. Goh, R. Weng, and J. Varadarajan, "STP-UDGAT: spatial-temporal-preference user dimensional graph attention network for next POI recommendation," *CoRR*, vol. abs/2010.07024, 2020.

[3] D. Yang, D. Zhang, and B. Qu, "Participatory cultural mapping based on collective behavior data in location based social networks," *ACM Transactions on Intelligent Systems and Technology*, vol. 7, 09 2015.

[4] W. Falcon and The PyTorch Lightning team, "PyTorch Lightning," Mar. 2019.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," *CoRR*, vol. abs/1502.01852, 2015.