Community Detection, Visualization and POI Prediction Based on Spatial-Temporal Trajectories

Jiaxu Feng
Department of Physics
Fudan University

Jia-ao Wu
School of Data Science
Fudan University

Zhiling Zhou
School of Data Science
Fudan University

Abstract

Large amounts of spatial-temporal data obtained from Location-Based Services have made it possible to understand user behavior further, yielding huge values for users and service providers. Three parts of work have been done in our project to exploit values from such data. Firstly, we constructed a social network graph from users' spatial-temporal records and used the Louvain algorithm to detect communities. We used word cloud to demonstrate the community features. Secondly, we utilized `D3.js` and `Leaflet.js` to visualize users' spatial-temporal trajectories and community features for further interpretations. Thirdly, we improved the GETNext model [19] with a bidirectional trajectory flow map and warm-up learning rate and achieved better performance for users' next POI prediction.

1 Introduction

With the prevalence of GPS-equipped mobile devices, Location-Based Services (LBS) are gaining increasing popularity. On Foursquare, for example, users could share their moments and experiences at their points of interest (POI). This accumulated spatial-temporal data has enormous potential value in terms of helping users find communities of similar interests and explore their surroundings, as well as providing insights for businesses' marketing strategies. In our work, we mainly focus on three tasks.

- Divide users into communities based on their spatial-temporal check-in records.
- Visualize the communities and spatial-temporal trajectories of users.
- Design a model to predict the next POI of users based on their historical POI.

The dataset we use comes from the work of Yang et al. [18]. It contains 227,428 check-ins in New York City lasting about 10 months (from 12 April 2012 to 16 February 2013) from 1083 users. Each check-in is associated with its time stamp, its GPS coordinates, and its semantic meaning (represented by fine-grained venue categories).

2 Social Network Construction and Community Detection

2.1 Social Network Construction

To construct a social network graph from users' check-in records, we first process the dataset by deleting unrelated check-ins. For example, visits to airports, subways, or private housing could not provide useful information in terms of which community users should belong to. Then we delete inactive users whose check-ins are less than 50 in the whole dataset. These users are likely to be isolated nodes and are not the main focus of our analysis. After data cleaning, we use the following algorithm to construct a social network graph.

Preprint. Under review.

Algorithm 1 Social Network Graph Construction

```
Initiate an empty graph G
for each unique user ID in dataset do
    Add a node named user ID to G
end for
for each pair of users (userA and userB) do
    Find common venues that they both have visited
    Initiate weight as 0
    for each venue in common venues do
        Count the number of visits to the venue of userA as AFreq and userB as BFreq
        Count the times when users visited the venue within an hour as visitWithinHour and times when they visited the venue within a day but not within an hour as visitWithinDay
        AFreq ← AFreq + 2 × visitWithinHour + visitWithinDay
        BFreq ← BFreq + 2 × visitWithinHour + visitWithinDay
        count ← total number of appearances of the venue in the dataset
        weight ← weight + (AFreq + BFreq) / abs(AFreq - BFreq) / count
    end for
    if weight > 1 then
        Add an edge between userA and userB to the graph G with edge weight as weight
    end if
end for
```

We use the divisor `abs(AFreq - BFreq)` for each sub-weight because the user pair with visit frequencies of 1 and 9 is less similar than the user pair with visit frequencies of 5 and 5. Moreover, the divisor `count` is used to prevent common places (like subway or bus stations) from creating strong similarities between users.

2.2 Community Detection

We use the Louvain algorithm for community detection. It first initializes every node as an independent community. For each node, it traverses its neighbor nodes and calculates the modularity gain when the node is moved to the community where the neighbor node is located. Select the move with the largest modularity gain and move the node to the community where the neighbor node belongs. After moving all nodes, merge each community into a supernode and build a new graph with supernodes. Repeat the previous step until there is no further modularity gain.[1]

We can demonstrate the features of each community detected by using the word cloud. We choose venue categories as feature tags and calculate the weight of each tag with the following method.

$$\text{Tag Specificity} = \frac{\text{times a venue category was visited by users in the community}}{\text{times a venue category was visited by all the users in the dataset}}$$

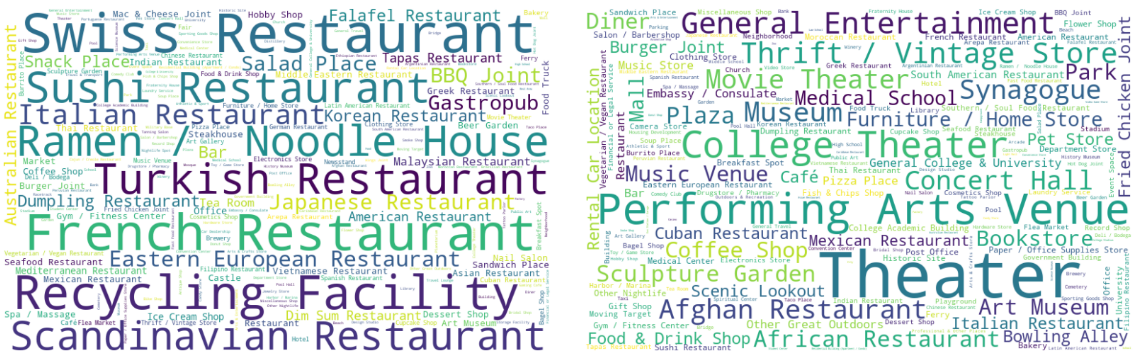
$$\text{Tag Representativeness} = \left(\frac{\text{number of users in the community who have visited the venue category}}{\text{total number of people in the community}}\right)^{\alpha}, \alpha = const$$

$$\text{Tag Weight} = \text{Tag Specificity} \times \text{Tag Representativeness}$$

Tag Specificity measures how unique this tag is to the community. Tag Representativeness measures how much percentage of users this tag can represent in the community. Using this visualization method, we can see from Fig. 1 that our algorithm successfully detected the communities including foodies and drama fans.

3 Community and Trajectory Visualization

To enhance the geographical representation and further validate the accuracy of our algorithm, we utilized `Leaflet.js` and `D3.js` for crafting interactive visualizations. `Leaflet.js`, known for its efficiency and ease of



(a) Key Word: **Restaurant**.

(b) Key Word: **Theater**.

Figure 1: Word cloud of the foodie community (left) and drama fan community (right)

use in mapping applications, alongside `D3.js`, a powerful tool for data-driven visualizations provide the foundation for this part. Leveraging these tools, we can not only gain a better understanding of the distribution of our original data but also explain the subtler aspects of social significance tied to these locations.

3.1 Overall Heat Map

The heat map is designed to reflect the frequency of visits to different POI. Areas with higher visit frequencies are depicted in darker red shades, indicating significant public engagement. This pattern is particularly noticeable in locations such as subway and bus stations, reflecting their role as critical public infrastructure. The visualization is interactive, allowing users to adjust the threshold of visit frequencies via a slider. This feature helps mitigate the issue of over-clustered points, ensuring that the map remains comprehensible even when representing areas with high POI densities.

3.2 Community Internal Heat Map

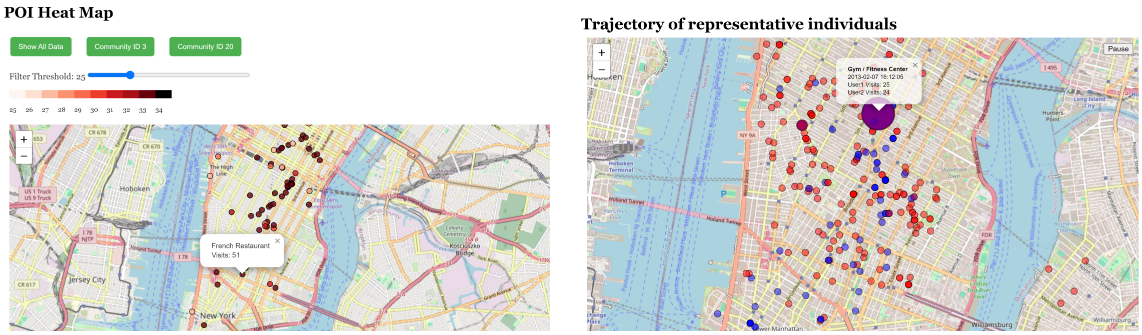
Focusing on the community level, the internal heat map visualizes the frequented spots within a specific community. As seen in Fig. 2 (a), we can specify which community to present and can click the circle dots to obtain the information of the locations. We have chosen a foodie community for visualization and most of the dark red dots are restaurants, which align with the findings of a prior word cloud analysis, underscoring the validity of both approaches. The internal heat map provides an insightful view into the community's dynamics, highlighting areas of high social and commercial activity.

3.3 Trajectory Visualization

The trajectory visualization tracks the movement patterns of two selected users. The system is dynamic, presenting both tracks in chronological order, as shown in Fig. 2 (b). Red and blue represent the two users respectively, and purple stands for the mixture color where the actions of the two converge. Initially, their paths show little overlap, suggesting independent routines. However, at a certain point—symbolizing their acquaintance—their trajectories begin to intersect frequently, narrating a compelling story of evolving friendship. This visualization not only maps physical movement but also beautifully encapsulates the development of human relationships over time.

4 Next POI Prediction

In this part, we focus on the task of the next POI prediction task, which intends to forecast users' immediate future movements given their current status and historical information. Specifically, we focus on the implementation, improvement, and interpretation of classical work of Yang et al. [19] from SIGIR22.



(a) Community internal heat map

(b) Trajectory of two users

Figure 2: Visualization of the community

4.1 Vanilla Method

In the literature, there has been much work in the POI prediction field. Some of them [2, 20] modeled the trajectory as the Markov model, and others [10] used Recurrent Neural Network (RNN) to find the POI as a sequential prediction task with spatial-temporal information. However, those works can not fully discover the information among users and places, which can be captured as a graph, thus inspiring models in [18]. Recently, some work [12] explored the famous attention mechanism for mining more complex information among embeddings.

However, those works do not pay much attention to the trajectory flow among places, which captures how many people tend to start from one place to another place (probability). Also, the models used in previous work are not powerful enough, and RNN-based models face the hardness of training. To solve that problem, [19] proposed a trajectory flow map enhanced transformer-based model, which combined the graph information inside the trajectory flow map with typical spatio-temporal information. Its basic structure is shown below in Fig. 3.

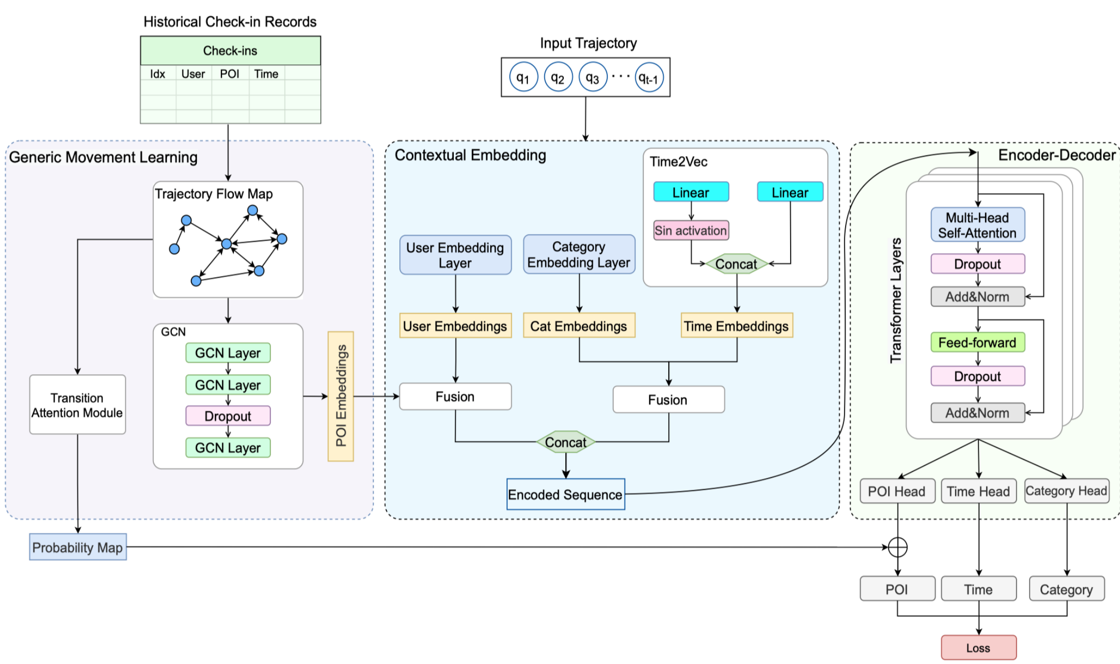


Figure 3: Structure of Vanilla GETNext Model (Figure from [19]).

To fully introduce the basic model, we break it down into three parts: trajectory flow map, embedding part, and transformer part, offering some brief and intuitive explanations in our way.

The trajectory is a directed graph, whose nodes represent places in the dataset, with check-in frequencies, locations, and categories. The edges in the graph are the frequencies from the outgoing nodes to the incoming nodes. After constructing the graph, we use Graph Convolution Network [9] to process the information in the graph and output the embeddings of nodes as embeddings of places. Moreover, we use Graph Attention Mechanism [15] to model the transition probability between places and act as a regularization of prediction, which can solve the overfitting problem and make the training process more smooth.

The embedding part focuses on the embedding and fusion of users, places, categories, and timestamps. The fusion part combines the user embedding and place embedding, the category embedding, and the timestamp embedding. After end-to-end training, this fusion strategy can smoothly model the user preference (where the user likes to go), and the connection between time and category (the user usually goes home at night and seldom goes to a bar in the morning).

The sequential prediction part using popular transformer model [14] encoder with linear layer as a decoder. It has three prediction heads, places (POI), category, and time. When training, the loss is the weighted sum of their loss. Specifically, the POI prediction is regularized by the transition map computed by the Graph Attention Mechanism [15] with a trajectory flow map.

For detailed information about the vanilla model, please refer to the original paper [19].

4.2 Improvement of the Model

In this part, we focus on how we can further improve the vanilla model. After exploration, we found that the original model from [19] is not good enough for the following reasons:

- The trajectory flow map is a directed graph, which makes the information aggregation in GCN [9] unbalanced, which might result in the hardness of training and insufficient information fusion.
- The transformer model used in the code from the original paper is not well-trained, for it does not use the efficient learning rate scheduler.

We focus on those parts and improve the model.

4.2.1 Using Bidirectional Trajectory Map

As shown in Fig. 4, the nodes in GCN [9] aggregate information from their neighbors. For a directed graph, nodes only aggregate information from nodes that have outgoing edges pointing to them, so the aggregation is unbalanced for places in the trajectory map.

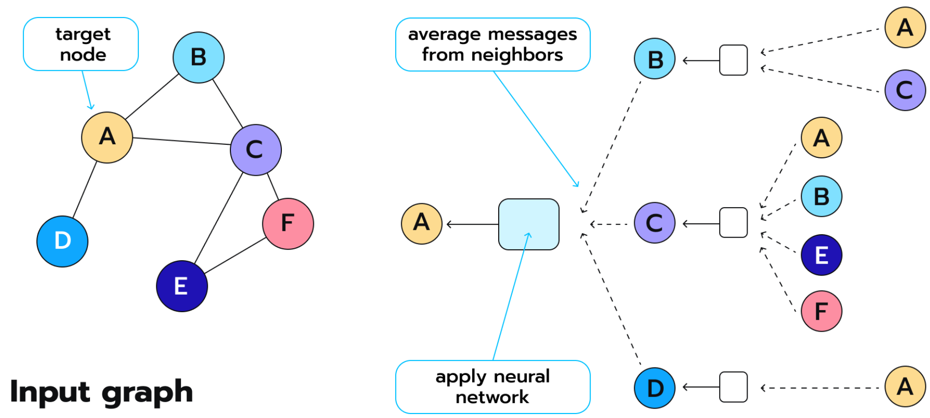


Figure 4: Structure of Graph Convolution Model. (Figure from [13])

To solve the problem, we average the incoming and outgoing frequencies, which is the value in the adjacency matrix, to make the trajectory graph undirected. Specifically, for adjacency matrix A,

$$\hat{A} = \frac{A + A^T}{2} \tag{4.1}$$

As the local picture of the trajectory flow map shows in Fig. 5, the bidirectional one is denser than the unidirectional one. We also display the global picture of the map in **??** in the Appendix. With higher density, the GCN can be trained more smoothly.

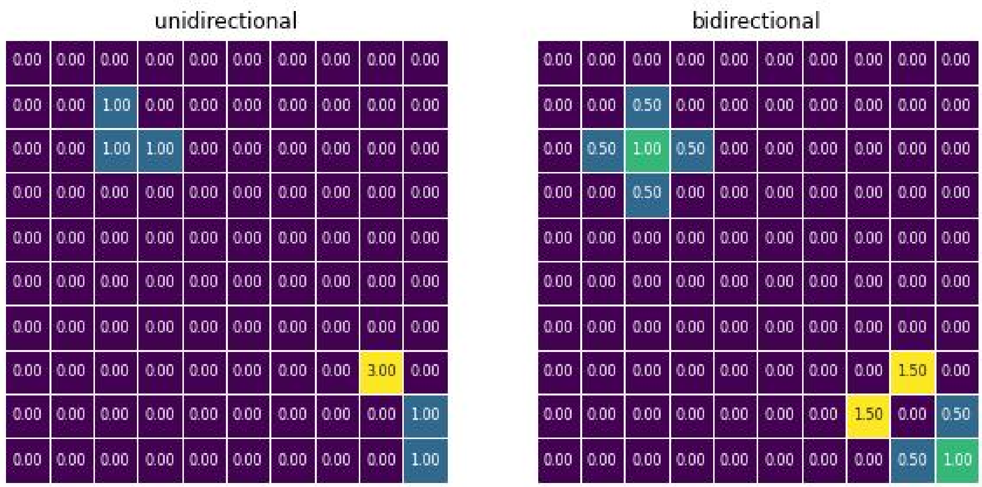


Figure 5: Local picture of the trajectory flow map, unidirectional version (left) and bidirectional version (right).

However, for the transition probability map, we still used the directed graph for it can better model how frequently a user moves from one point to another. The reason for this strategy is that the transition probability is highly linked to the chronological order.

4.2.2 Warm-up, Cosine Scheduler, and AdamW

In the code [1] of the original paper, it optimizes the model with Adam optimizer [8] and uses the validation loss to adjust the learning rate. When the validation loss increases, it will decrease the learning rate. However, this strategy will make the model more easily to be trapped in local minima because it does not have enough learning rate to escape. According to our experiments, the training of the model is unstable and our reproduction is worse than the result in the paper with the same quantity of parameters.

To make the training more stable and enhance the model's generalization capability, we choose to optimize the model with AdamW [11] and use a cosine learning scheduler with a warm-up step.

AdamW is an improved version of Adam, it decouples the weight decay regularization from the momentum but concentrates on a gradient. As is proposed in [11], it might enhance the generalization ability of models for its regularization. Besides the optimizer, the warm-up step, proposed in [5], is a commonly used strategy in transformer training. It initiates the learning rate as a small value and gradually increases it to a set value, after which the learning rate will follow the cosine annealing scheduler. The cosine annealing scheduler can help the model escape local minima. The warm-up step is illustrated in Fig. 6. In previous work [7, 16], the warm-up step is well studied that it can benefit the training of transformer models, for it can better update the parameters in the first few steps.

4.2.3 Experimental Results

In our experiments, we follow all the parameters set as those in the vanilla model, besides the learning rate, for a fair comparison. We compare our results with classical models using recurrent neural network, LSTM [6], ST-RNN [10], models using GNN, ST-GCN [4], and previous models that explore attention mechanism STAN [12].

[1] `https://github.com/songyangme/GETNext`

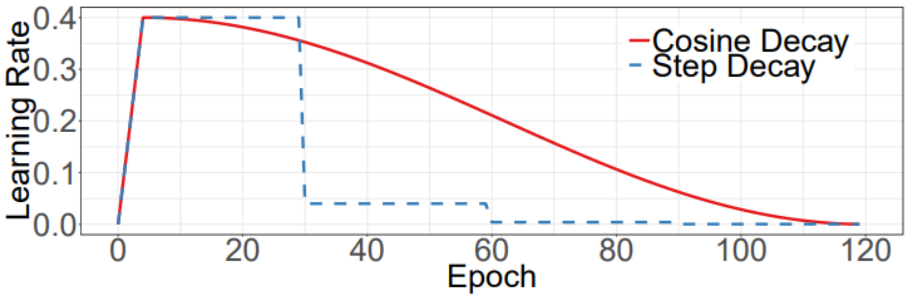


Figure 6: Warm-up with cosine decay scheduler. (Figure from [5])

As is shown in Tab. 1, "Original" stands for those results provided in the original paper. "Reproduction" stands for our reproduction following the original settings. "Larger Embeddings" stands for a higher dimension of embeddings of the users. "GIN" stands for using Graph Isomorphism Network [17] for POI embedding. "ELU" stands for replacing ReLU activation with ELU activation [3]. "ProbMap-Bi + GCN-Bi" stands for using undirected graphs in POI embedding and transition probability map computation. "ProbMap-Uni + GCN-Bi" stands for using undirected graphs in POI embedding but directed graphs in transition probability map computation. "Warm-up + ProbMap-Uni + GCN-Bi" stands for using warm-up cosine scheduler and AdamW with "Uni+Bidirectional" settings. The last three lines are our ablation study.

Table 1: The accuracy of point of interest prediction

Model	Top1	Top5	Top10	Top20	MRR
LSTM	0.1305	0.2719	0.3283	0.3568	0.1857
ST-RNN	0.1483	0.2923	0.3622	0.4502	0.2198
STGCN	0.1799	0.3425	0.4279	0.5370	0.2806
STAN	0.2231	0.4582	0.5734	0.6328	0.3253
Original	0.2435	0.5089	0.6143	0.6880	0.3621
Reproduction	0.2370	0.5085	0.6148	0.6819	0.3627
Larger Embedding	0.2387	0.5001	0.6047	0.6770	0.3576
GIN	0.2462	0.5003	0.6137	**0.6932**	0.3654
ELU	0.2359	0.5029	0.6015	0.6868	0.3604
ProbMap-Bi + GCN-Bi	0.2465	0.5169	0.6162	0.6882	0.3678
ProbMap-Uni + GCN-Bi	0.2532	0.5014	0.6121	0.6783	0.3646
Warm-up + ProbMap-Uni + GCN-Bi	**0.2585**	**0.5201**	**0.6224**	0.6863	**0.3759**

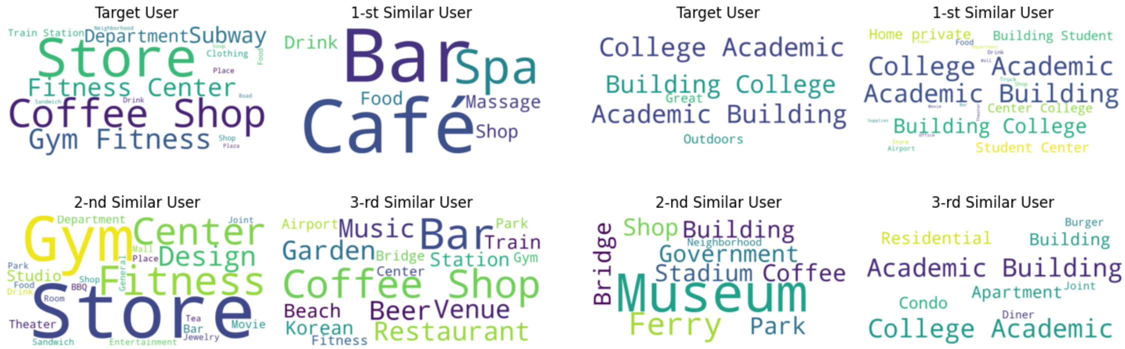
From the results in Table 1, we can observe that blindly adding the parameters, changing models, and activation functions barely help improve the models. Using a bidirectional trajectory flow map can improve the accuracy. However, only using it in POI embedding can further improve the model. With warm-up cosine scheduler and AdamW, the model can achieve the best results, which is at least 1% higher than the vanilla model. According to these results and the ablation study, it can be seen that our method is effective.

4.3 Interpretation of the Model

To further analyze the model, we try to use the user embedding to explain why the model is good. We choose our best model and get the embedding of all the users, which is a 128-dimensional vector in this case. We compute the cosine distance (shown in Eq. 4.2) of them and find the 3 most similar users among all users. We use word cloud figures to show the category names of the places they visited to show their similarity.

$$d(x,y) = \frac{x^T y}{\|x\|\|y\|} \tag{4.2}$$

As shown in Fig. 7, the left one is a user who prefers to go to cafes and stores, and the similar users we find have almost the same points of interest. The right one is a college professor or student, and the similar users we found also



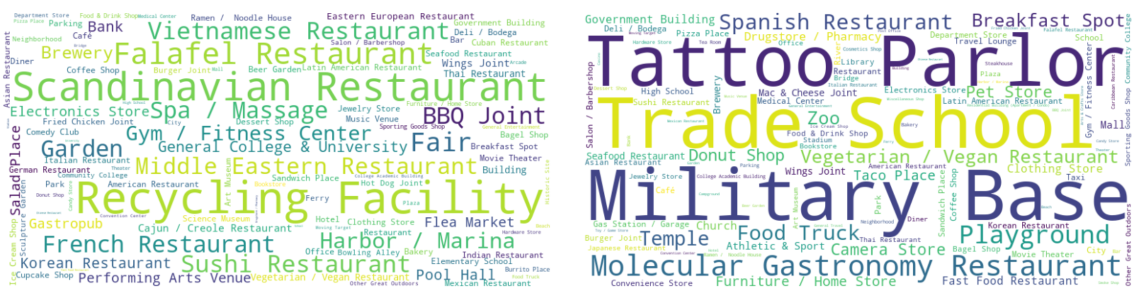
(a) Key Word: **Coffee / Store**.

(b) Key Word: **College / Academic**.

Figure 7: Word cloud of the places that users and their similar users are interested in. For each example, the top left one is the target user and the other is its similar users.

have the same pattern. These observations demonstrate that the model, using an end-to-end training paradigm, can capture the profile of the users.

Besides, we connect every user with its 3 most similar users to build a graph and use the same method before to detect communities using [1].



(a) Key Word: **Restaurant**.

(b) Key Word: **Tatto**.

Figure 8: Word cloud of the foodie community (left) and tattoo lovers community (right).

As is shown in Fig. 8, we can find that we can not only find similar results as in the Community Detection part but also find other communities with interesting features. For example, people in the right graph who like to go to tattoo parlors are usually from trade schools or the army. These results can further interpret our model, which can also act as prior in the downstream task, such as recommendation.

5 Conclusion

In this project, we construct a social network graph of using heuristic spatial-temporal information and group users into communities using the Louvain algorithm. Additionally, we visualize the check-in frequencies at POI and plot users' spatial-temporal trajectories. Subsequently, we employ a trajectory flow map-enhanced transformer model to forecast the next point in the trajectory. To enhance the model, we incorporate bidirectional flow maps and a warm-up learning rate. However, the existing model and method solely rely on spatial-temporal information, overlooking the semantic details of locations. Moving forward, the model can be further refined by incorporating semantic information or a combination of LLM.

References

[1] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008, 2008.

[2] Chen Cheng, Haiqin Yang, Michael R Lyu, and Irwin King. Where you like to go next: Successive point-of-interest recommendation. In *Twenty-Third international joint conference on Artificial Intelligence*, 2013.

[3] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[4] Thomas MJ Fruchterman and Edward M Reingold. Graph drawing by force-directed placement. *Software: Practice and experience*, 21(11):1129–1164, 1991.

[5] Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. Bag of tricks for image classification with convolutional neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 558–567, 2019.

[6] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[7] Xiao Shi Huang, Felipe Perez, Jimmy Ba, and Maksims Volkovs. Improving transformer optimization through better initialization. In *International Conference on Machine Learning*, pages 4475–4483. PMLR, 2020.

[8] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[9] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[10] Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. Predicting the next location: A recurrent model with spatial and temporal contexts. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

[11] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[12] Yingtao Luo, Qiang Liu, and Zhaocheng Liu. Stan: Spatio-temporal attention network for next location recommendation. In *Proceedings of the web conference 2021*, pages 2177–2185, 2021.

[13] Benjamin Sanchez-Lengeling, Emily Reif, Adam Pearce, and Alexander B Wiltschko. A gentle introduction to graph neural networks. *Distill*, 6(9):e33, 2021.

[14] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[15] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[16] Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. On layer normalization in the transformer architecture. In *International Conference on Machine Learning*, pages 10524–10533. PMLR, 2020.

[17] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[18] 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, 2014.

[19] Song Yang, Jiamou Liu, and Kaiqi Zhao. Getnext: trajectory flow map enhanced transformer for next poi recommendation. In *Proceedings of the 45th International ACM SIGIR Conference on research and development in information retrieval*, pages 1144–1153, 2022.

[20] Jia-Dong Zhang, Chi-Yin Chow, and Yanhua Li. Lore: Exploiting sequential influence for location recommendations. In *Proceedings of the 22nd ACM SIGSPATIAL international conference on advances in geographic information systems*, pages 103–112, 2014.

Appendix: Task Allocation

- Jiaxu Feng: Social Network Construction and Community Detection
- Jia-ao Wu: Community and Trajectory Visualization
- Zhiling Zhou: Next POI Prediction