2020 Master's Thesis

Point of Interest Recommendation Acceleration Using Clustering

A Thesis Submitted to the Department of Computer Science and Communications Engineering, the Graduate School of Fundamental Science and Engineering of Waseda University in Partial Fulfillment of the Requirements for the Degree of Master of Engineering

Submission Date: January 25th, 2021

Advisor: Prof. Hayato Yamana
Research guidance: Research on Parallel and Distributed Architecture

Department of Computer Science and Communications Engineering,
Graduate School of Fundamental Science and Engineering,
Waseda University
Student ID: 5119FG06-1

Huida Jiao

Abstract

Point of Interest (POI) recommendation systems exploit information in location-based social networks to predict locations that users may be interested in. POI recommendations have been widely adopted in many applications, which are helpful for daily life. POI recommendation services receive a huge volume of visit history data generated by users' daily lives with mobile devices. However, POI recommendation systems require long time to build a model from such a huge volume of check-in data and recommend suitable POIs to users. In industry, the recommendation system needs to respond quickly to user requests. Thus, it is indispensable to shorten the execution time of POI recommendation system in a big data era.

In this study, we propose a clustering-based method to divide the data into multiple subsets to accelerate the POI recommendation's execution while maintaining accuracy. Our proposed method can be adapted to any general POI recommendation algorithm. We divide the whole data, that is, users and POIs, into subsets with a tree structure to balance the size of subsets according to both geographical information and user check-in distribution. Evaluation results show that we successfully accelerate the base algorithms over 17 to 39 times faster while keeping the accuracy almost the same.

Contents

1. Introduction

Recommendation systems aiming to predict the items that users are potentially interested in, face information overload problems in the big data era. Point of Interest (POI) recommendation systems focus on advising users on locations to visit. With the widespread use of mobile devices, location-based social network (LBSN) services can collect a huge volume of check-in data generated by users. By exploiting features from such data, personalized potential interest locations can be recommended to the users, which is beneficial for both users and merchandises. Users are more satisfied if preferred locations can be found using POI recommendation system.

Many researchers have adopted different techniques to improve the accuracy of the recommendation systems [1]. However, besides accuracy, computation cost, especially the execution time when training the model and recommendation, are crucial in practice because the LBSN service needs to generate a list of recommended POIs in a short time. In the industry, a real-time recommendation system is usually required to provide the result within one second [2]. Meanwhile, the data volume is large and keeps growing fast because it receives daily users' activities. For example, Yelp[1] earns 2 million new reviews of POIs each month. Thus, building a highly efficient POI recommendation system is challenging. As pointed out in [3], some POI recommendation systems lack scalability. Furthermore, applying deep learning to a recommendation system, which is a popular trend, is suspected to be extremely time-consuming [4]. Therefore, the efficiency of the POI recommendation systems also needs to be considered in a real scenario.

In the POI recommendation systems, geographical information is one of the most influential aspects. Many users tend to visit locations near their home or active area [5], which could be an insight to accelerate the recommendation. For a specified user, only a small portion of his/her nearby POIs are possible to be visited in most cases. Because the recommendation execution time, which includes model training and query processing, can be decreased if we narrow down the number of POIs to analyze, dividing both POIs and users into small subsets becomes a way to speed up the recommendation process.

In this study, we propose a novel clustering-based acceleration method to shorten the execution time of the POI recommendation system. The original input data are divided into smaller subsets and fed into the recommendation model. The POIs in a small region,

[1] https://www.yelp-press.com/company/fast-facts/default.aspx

along with users who are active in such regions, are clustered into the same subset to reduce the information loss. The subsets are organized in a tree structure to balance the size of subsets to effectively shorten the execution time. Our division method can be adapted to any general POI recommendation algorithm.

In the rest of the paper, related work is introduced in Chapter 2. Details of the proposed method are described in Chapter 3. We present the experimental result in Chapter 4 and the conclusion in Chapter 5.

2. Related Work

We briefly introduce general POI recommendation models and review recent work on high-speed models to achieve real-time recommendations. We also list the previous work applying clustering in POI recommendation systems.

2.1.POI Recommendation

Ye et al. [6] proposed USG, a pioneering algorithm to use geographical information in the POI recommendation field. Geographical information is also combined with other aspects to model user preferences [7][8]. Social relationship information can also be used because users' check-in preferences are inferred from their friends [9][10]. Factorization-based models [11][12] use matrix factorization to decompose user-poi check-in matrix. Deep neural networks (DNNs) have also proved to be effective in POI recommendations. Yu et al. [13] designed LSTM-based deep encoders to capture the user's categorical preference and POI preference. Zhao et al. [14] proposed a variant of LSTM to capture the spatio-temporal relationship between POIs. However, these models focus on accuracy, but the efficiency is ignored, and both DNN-based models and models without DNN are time-consuming [4][15].

2.2.Real-time POI Recommendation

Efforts have been made to speed up the execution time of POI recommendation systems, especially by researchers in industries. Agarwal et al. [16] proposed a two-stage approach to accelerate the top-k information retrieval system by approximating the scores of items adapted to the common recommendation system. Jiao et al. [17] proposed a real-time next POI recommendation system that integrates a target user's geographical and temporal preferences and uses a convolutional neural network to classify target user's personal reviews. The goal of the system is to recommend the next POI given current time and location. Fan et al. [18] adapted a Golang-based architecture to implement a real-time online POI recommendation system focusing on the ride-hailing transport service, and the goal of the system is to predict and suggest the start and end point of user's ride. Wang et al. [19] leveraged a lightweight neural network to compress the model to execute the next POI recommendation task on mobile devices with limited computation resources. However, they aimed at specific tasks, such as the next POI and searching POI with names. To the best of our knowledge, none of the existing real-time POI recommendation systems generate a list of recommended POIs for users.

2.3.POI Recommendation Models using Clustering

A clustering technique can also be applied to POI recommendation models to achieve different goals. Si et al. [20] clustered all users into active users and inactive users according to their behavior using k-means and used different criteria to select a similar user for different clusters. Zhu et al. [21] clustered users' trust matrix using Fuzzy-C-Means to predict POI and friendship in LBSN. Massimo et al. [22] adopted non-negative matrix factorization to enable a spot recommendation for tourists by clustering the attributes of POIs in visit history. None of the previous studies adopted clustering to accelerate the recommendation process.

2.4.Summary

In this chapter, related work about POI recommendation system is introduced. Specificly, we introduced researches focusing on acceleration and models using clustering technique, which are shown in Table 1.

Table 1 Summary of related work

Research	**Proposal**	**Shortcoming**
Agarwal et al. [16]	A two-stage approach to accelerate top-k information retrieval system	Not using POI information
Jiao et al. [17]	Integrating user's geographical and temporal preference with review	Next POI generation, not a recommendation list
Fan et al. [18]	A Golang-based architecture to implement real-time POI recommendation	Focusing on specific task of ride-hailing transport service, not general
Wang et al. [19]	A light-weighted network that can be used on resource-constraint devices	Next POI generation, not a recommendation list
Si et al. [20]	Clustering all users into active and inactive groups and adapting different strategy	Not focusing on acceleration, Not considering geographical information
Zhu et al. [21]	Clustering users' trust matrix to predict POI and friendship	
Massimo et al. [22]	Clustering attributes of POIs to recommend spot for users	

3. Proposed Method

3.1.Method Overview

First, we present an overview of a common POI recommendation system with notations. As shown in Figure 1, the input of the POI recommendation system is a tuple $D =< P, U, C, S >$, where P is the POI set, U is the user set, C is the check-in set, and S is the social relationship set. For each POI $p \in P$, the associated geographical coordinates $< lon_p, lat_p >$ represent the longitude and latitude of POI p. A check-in history $c \in C$ is denoted by $c =< u, p >$, which represents the user $u \in U$ visiting POI $p \in P$. For convenience, we denote C^u as all the checked-in POIs of user u. Social relationship S is the friendship between users $S = \{(u, v) | u \in U, v \in U, u \neq v\}$. The output of the POI recommendation system is the prediction $\hat{C} = \{< u, p > | u \in U,\ p \in P \setminus C^u\}$, which recommends unvisited POIs to the user.

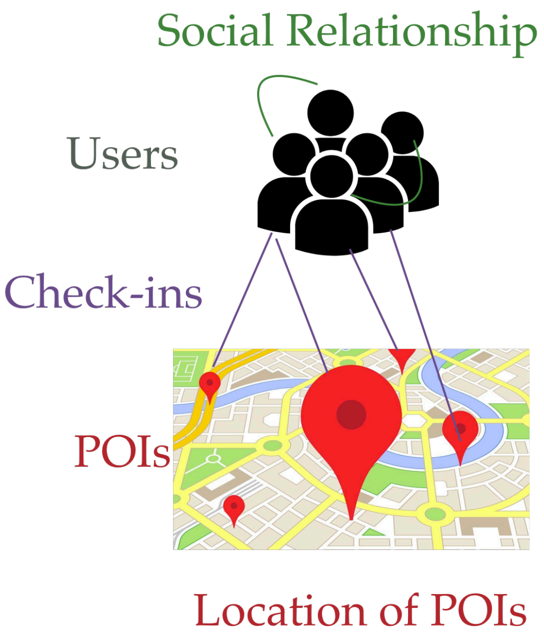

Figure 1 Dataset of common POI recommendation system

The goal of our proposed method is dividing the original input dataset D into n disjoint subsets, denoted as $D_i =< P_i, U_i, C_i, S_i >$, followed by inputting each subset to a recommendation system. Because the input data size is reduced, the recommendation system can generate the recommendation result much faster. Specifically, each POI p in set P is assigned to one of the subsets P_i. Similarly, each user u in set U is assigned to one of the subsets U_i. Every POI and user are kept and mapped to a subset, i.e., Equation (1) holds.

$$P = \bigcup_{i=1}^{n} P_i \, , U = \bigcup_{i=1}^{n} U_i \tag{1}$$

The division results in some information loss of the check-ins and social relationships. For subset C_i, only the check-ins of users $u \in U_i$ for POI $p \in P_i$ are included, i.e., $C_i = \{< u, p > | u \in U_i, p \in P_i\}$. Namely, even if the user u visited other POIs $p \notin P_i$, such check-ins are not included. Similarly, only the social relationships between two users in the same user subset are kept in subset $S_i = \{< u, v > | u \in U_i, v \in U_i, u \neq v\}$.

Dataset division is achieved in multiple rounds each of which is shown below:

1. Select the largest POI subset (initially, whole POIs) to cluster into k small subsets and organize them in a tree structure;
2. Assign each user to one of the subsets in which his check-ins are included most frequently;
3. Compute the average coverage ratio, which is the average ratio of user's checked POIs in his most frequently checked cluster to his all checked POIs.

At the end of each round, the calculated average coverage ratio is compared to the threshold λ; if the criterion is met, the POI and user division is completed; otherwise, the largest subset is extracted and divided again using the procedure above.

After finishing the POI and user division, the check-in history and social relationship are divided into the subsets according to the assignment of the POIs and users. The entire division process is shown in Algorithm 1.

Algorithm 1: Dataset division

```
Input: <P, U, C, S>: original dataset
           k, $\lambda$: parameters of number of clusters and threshold
Output: $P_i, U_i, C_i, S_i$: divided dataset

1    T ← new tree (P)
2    ratio ←1
3    **while** (ratio > $\lambda$) **do**
4        x←ExtractLargestSubset(T)
5        Cluster POI set x into k subsets
6        Add cluster result (k subsets) to T
7        $U_i$ ←UserAssignment(Leaf nodes of T)
8        ratio ← AverrageCoverageRatio (T, C)
9    **end while**
10   divide check-in set C according to $P_i$ and $U_i$
11   divide social relation set S to $S_i$ according to $U_i$
12   **return** $P_i, U_i, C_i, S_i$
```

After the original dataset D is divided into several subsets D_i, each subset is fed into the POI recommendation system, and the POI recommendation system will generate the recommendation result.

The POI clustering process and tree structure are introduced in Section B; the average coverage ratio and the rest of the processes are introduced in Section C.

3.2. POI Clustering with Tree Structure

The proposed division procedure mainly relies on clustering POIs according to the coordinates because geographical information is the most impacting aspect. However, simply clustering POIs into small subsets will generate imbalanced subsets, which contributes slightly to acceleration. Thus, we cluster the POIs for multiple rounds and organize the result of each round's clustering of POI into a tree structure to balance the size of subsets.

Each node of the tree represents a set of POIs belonging to the same cluster. First, a single root node with all the POIs is created. In each round, the leaf node with the largest number of POIs is selected to be clustered. The selected set of POIs is divided into k subsets that become the child nodes of the selected node. After clustering the selected leaf node at each round, the users originally assigned to the selected node are re-assigned to one of the clustered nodes. The average converge ratio is then calculated, which is introduced in section C. If the ratio is larger than the threshold λ, the next round of clustering is applied; otherwise, the current tree structure composes the final POI clustering result. Each leaf node represents one subset of all POIs.

Figure 2 shows an example of the tree structure of dividing a dataset after three rounds when $k = 4$. Each node represents a subset of POIs; the number of the node shows the size of the POI set; the edges are added from parent node to child node; the leaf nodes are

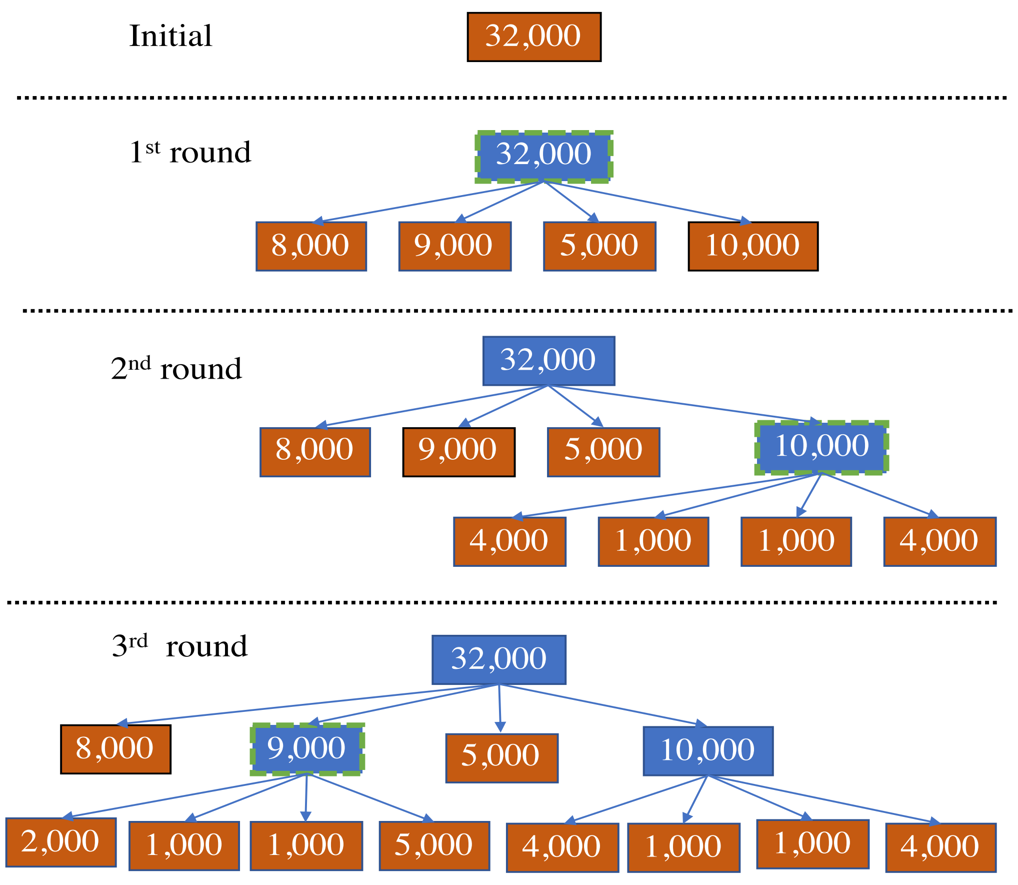

Figure 2 Example of POI clustering with tree structure (k=4)

colored in brown; the largest subset is dotted.

The original 32,000 POIs are divided into ten subsets after three rounds; each subset has less than 10,000 POIs.

We adopted k-means, a commonly used and effective clustering algorithm, to cluster the POIs. We choose k-means as our clustering algorithm because of its efficiency and the easy specification of the number of clusters.

3.3. User Assignment and Average Coverage Ratio

In each round, POIs are divided into subsets. Besides, the users are divided according to the assignment of their checked POIs. Then the average coverage ratio is calculated according to the POI clustering and user assignment. The user assignment and average coverage ratio calculation are shown in Algorithm 2.

As explained in section B, suppose the current tree structure has m leaf nodes, i.e., the whole POI set P is divided into disjoint subsets $P_1, P_2, P_3, \ldots, P_m$. For the user u, we retrieved all his checked POIs $p \in C^u$, and counted the frequency of his checked subset as (2):

$$freq_i^u = \frac{|\{p|p \in (C^u \cap P_i)\}|}{|C^u|} \tag{2}$$

As shown in (3), the subset that includes the user's most frequently checked POIs is assigned as the user's subset.

$$tag_u = argmax_i(freq_i^u) \tag{3}$$

The average coverage ratio of all users, shown in (4), indicates how much checked POI information is kept under the current division. A larger coverage ratio means that more POIs are covered for each user after division, and less information is lost.

$$AvgCovRatio = \frac{\sum_{u \in U} freq_{tag_u}^u}{|U|} \tag{4}$$

The average coverage ratio is initialized as 1 because the original dataset is regarded as only one set. With more rounds of division, the ratio keeps decreasing because more POIs are split into small subsets. The ratio should not be too small because the accuracy

Algorithm 2: User assignment and average coverage ratio calculation

```
Input: $(P_1, P_2, \dots, P_m)$: divided POI set
            C: check-in information
Output: $\{U_i\}$: divided user set
            ratio: average coverage ratio
1    Initialize each $U_i$ as $\emptyset$
2    SumRatio = 0
3    for $u \in U$ do
4        initialize $\mathrm{freq}_i^u$, $\mathrm{tag}_u$ , MaxFreq as 0
5        for p $\in$ $C^u$ do
6            t ← obtain assigned cluster number of p
7            $\mathrm{freq}_t^u$ ← $\mathrm{freq}_t^u$ +1
8        ▷$\mathrm{freq}_t^u$ means the percentage of POIs that
9                user $u$ checked in subset $P_t$
10       end for
11       for i ← 1 to m do
12           if $\mathrm{freq}_i^u$ > MaxFreq then
13               MaxFreq ← $\mathrm{freq}_i^u$   ▷save coverage ratio
14               $\mathrm{tag}_u$ ← i
15               ▷ user $u$ most frequently visits POIs in set
16                 $P_i$, thus i becomes user $u$'s tag (assignment)
17           end if
18       end for
19       $U_{\mathrm{tag}_u} \leftarrow U_{\mathrm{tag}_u} \cup u$
20       SumRatio ← SumRatio+MaxFreq
21   end for
22   ratio ← SumRatio / |U|
23   return ratio, $\{U_i\}$
```

will drop owing to the loss of check-in information. Thus, we set a threshold parameter λ to control. As shown in Algorithm 1, the division iterates until the ratio is less than λ, i.e., the stop condition is ratio$< \lambda$.

3.4. Social Relationship Division

After the final round, each user $u \in U$ is assigned to one of the subsets U_i. Because each subset will be calculated separately, the original social relationship set S should also be divided into subsets according to the assignment of users.

For one relationship $< u, v > \in S$ between user u and user v, $< u, v >$ will be assigned to subset S_i if $u \in U_i$ and $v \in U_i$. Otherwise, the relationship between users in different subsets is discarded.

4. Evaluation

4.1.Datasets

To evaluate the effectiveness of our proposed method, we chose two datasets, Gowalla[1] and Yelp[2], to conduct the experiments. For each user, the earliest 70% of check-ins are used as a training set, the latest 20% are used as a testing set, and the remaining 10% are used as a tuning set. The split ordered by check-in time avoids data leakage; thus, we can conduct an accurate evaluation. We use the dataset cleaned and split by Liu et al. [3]. The details of the datasets and filtering criteria are shown in Table 2.

Table 2 Detail of Dataset

Dataset	**Size after filtering**		**Filtering criteria**	
	# of POI	***# of user***	***Cold POI***	***Cold User***
Gowalla	32,510	18,737	<10 visitor	<15 checked POI
Yelp	18,995	30,877	<10 visitor	<10 checked POI

4.2.Base Algorithm

As there are many different techniques used in POI recommendation systems, we selected four representative algorithms as base algorithms to experiment:

1. USG [6] is a hybrid algorithm that linearly combines geographical information, user preference, and social relationship to recommend POIs.
2. LFBCA [9] mainly mines social relationship information to model social influence. The similarity between users is computed based on the friendship and user preference, and user-based collaborative filtering is used to recommend POIs.
3. MGMPFM [11] adapted the Poisson factor model to factorize the check-in matrix and combine the user's region preference.

[1] http://snap.stanford.edu/data/loc-gowalla.html

[2] Yelp dataset challenge round 7, https://www.yelp.com/dataset challenge

4. iGSLR [10] is a hybrid model using social relationship and geographical information. User’s most frequently checked POIs are treated as residence, and user similarity is calculated based on distance between residences.

We slightly altered these base algorithms' implementation from [4]; we enabled time measuring, feeding split data, and evaluation of the result in subsets. All the codes are written in Python for a fair comparison.

4.3.Metric

We use precision and nDCG to evaluate the quality of recommendation results in addition to execution time to evaluate the acceleration.

Precision is the ratio of the correct recommended POI in the recommendation result list. Let L_k^u be user u’s first k recommended POIs in the result list, and G^u be user u’s ground truth, the $Precision@N$ of user u is calculated as (5).

$$Precision@N = \frac{|L_N^u \cap G^u|}{N} \qquad (5)$$

nDCG is a common metric that evaluates the ranking quality of information retrieval systems, and is calculated by $nDCG@N = \frac{DCG@N}{iDCG@N}$, and DCG is calculated as (6):

$$DCG@N = \sum_{i=1}^{N} \frac{2^{rel_i} - 1}{log_2(i+1)} \qquad (6)$$

, where rel_i is the relevance score of the i-th POI in the recommendation result list, and $iDCG$ is the DCG of the ideal recommendation result.

The execution time comprises three steps: clustering the dataset, training base algorithm, and query processing. Shortening the query processing time is the most important because recommendation systems in the industry should respond to the user’s request in time. On the contrary, the dividing and the training processes can be executed offline. Here, the query processing time is shown as the execution time per user for a fair comparison.

We use the time command on Linux server to obtain the execution time. The experiment environment was CentOS 6 on an Intel Xeon E5-2620 @2.10 GHz CPU and

128GB memory. Note that although these executions can be run in parallel because the divided subsets are independent, we report the execution time with one thread for a fair comparison.

4.4. Parameter Tuning

We have two parameters, k and λ to tune. The main task is to tune λ because it is directly related to the division process (smaller λ leads to more subsets). To tune λ, we first need to decide the value of k. According to our proposed division algorithm, large k leads to a drastic drop in the average coverage ratio (because POIs are clustered in more subsets) during the clustering and causes the missing of the optimal value of λ, while small k slows down the division process. Thus, we empirically choose $k = 4$ to achieve a suitable granularity to tune λ.

We tuned the parameter λ on the tuning set of the Gowalla and Yelp datasets. We use a grid search from 0.50 to 0.98 with an interval of 0.02. A smaller λ can achieve more acceleration, but the accuracy will decrease. We try to find the lowest possible λ while maintaining the accuracy. By observing the Precision@10 and execution time simultaneously, we choose $\lambda = 0.80$ for the Gowalla dataset and $\lambda = 0.60$ for the Yelp dataset.

4.5. Evaluation Result and Discussion

We adopted our proposed method to four base algorithms on two datasets. The results are shown in Table 3. In our experiment, we set $N = 10$ for precision and nDCG metrics. The results of the original model that is without division are also listed. The better results (higher accuracy or shorter time) are in bold.

We accelerated the querying of USG by 17 times for the Gowalla dataset and 28 times for the Yelp dataset while maintaining the accuracy. The $Precision@10$ and $nDCG@10$ are almost similar or increased slightly, which is because of the elimination of noise. Note that USG does not have a training process, thus N/A is shown in Table 2.

Similarly, the querying speed of LFBCA is accelerated by 29 times for the Gowalla dataset and 22 times for the Yelp dataset. The training time is also reduced because the training time of the LFBCA is sensitive to the number of users. The accuracy is also increased slightly.

Although the accuracy and training time of MGMPFM is very close to the original ones, the querying of the original model is accelerated drastically, which is 18 times faster on the Gowalla dataset and 30 times faster on the Yelp dataset.

iGSLR, the slowest model among 4 base algorithms, is accelerated drastically. Although the precision of iGSLR is decreased very slightly (less than 0.002) for both datasets, the querying is over 24 and 39 times faster on Gowalla and Yelp dataset, respectively. iGSLR does not have a training process either, thus N/A is shown for training time.

Table 3 Evaluation Result

Dataset	**Base Algorithm**	**Configuration**	***Precision*@10**	***nDCG*@10**	**Clustering Time(sec)**	**Querying Time per user(sec)**	**Training Time (sec)**
Gowalla	USG	original	0.0483	0.0724	N/A	3.532	N/A
		proposed	**0.0486**	**0.0734**	26	**0.206**	N/A
	LFBCA	original	0.0459	0.0677	N/A	0.088	1,380
		proposed	**0.0466**	**0.0693**	26	**0.003**	**306**
	MGMPFM	original	**0.0222***	**0.0336**	N/A	4.339	**351**
		proposed	0.0222	0.0326	26	**0.235**	408
	iGSLR	original	**0.0283**	**0.0403**	N/A	4.766	N/A
		proposed	0.0265	0.0373	26	**0.192**	N/A
Yelp	USG	original	0.0224	0.0385	N/A	1.644	N/A
		proposed	**0.0240**	**0.0403**	32	**0.057**	N/A
	LFBCA	original	0.0188	0.0320	N/A	0.090	2,802
		proposed	**0.0209**	**0.0351**	32	**0.004**	**498**
	MGMPFM	original	0.0149	**0.0254**	N/A	2.392	341
		proposed	**0.0150**	0.0253	32	**0.079**	**337**
	iGSLR	original	**0.0113**	**0.0177**	N/A	5.239	**N/A**
		proposed	0.0110	0.0171	32	**0.132**	**N/A**

* The Precision@10 of original MGMPFM is 0.02224, higher than that of proposed one 0.02220.

(All the difference of Precision and nDCG between original and proposed method is statistically siginifcant at $p<0.001$)

To better address the performance among base algorithms on two datasets, Figure 3~6 are shown below for clear comparison.

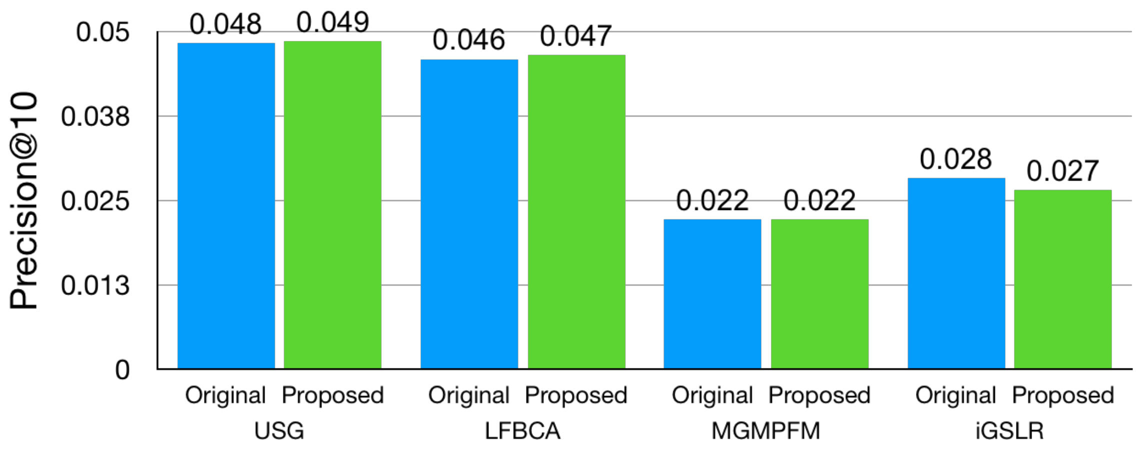

Figure 3 Precision on Gowalla dataset

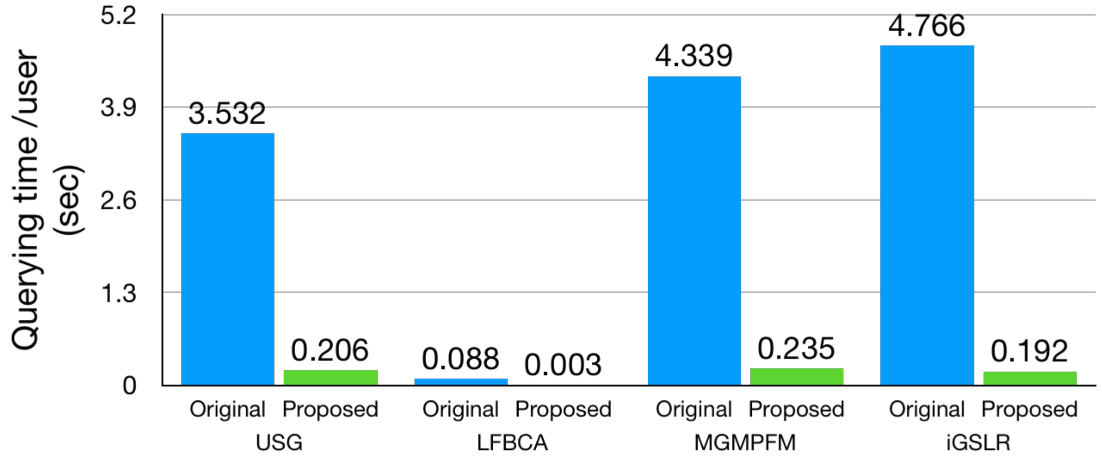

Figure 4 Querying time on Gowalla dataset

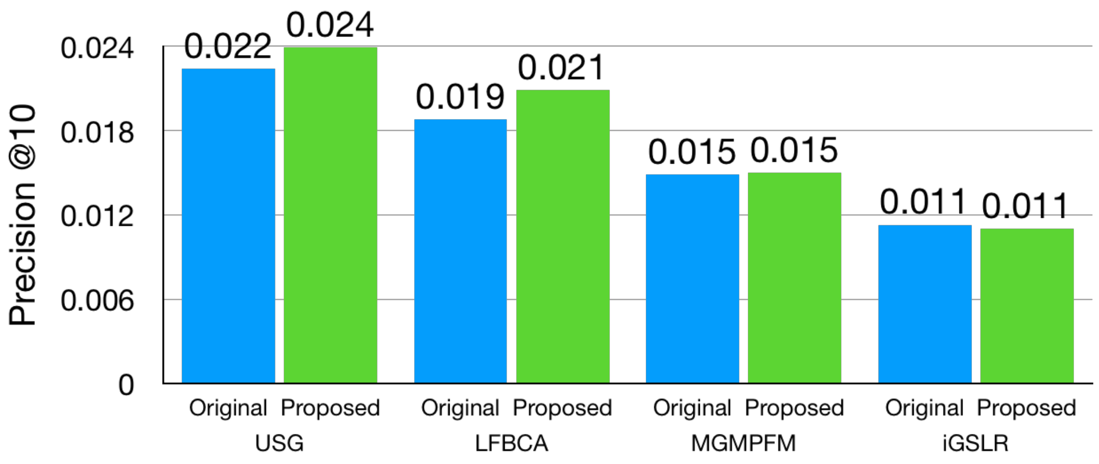

Figure 5 Precision on Yelp dataset

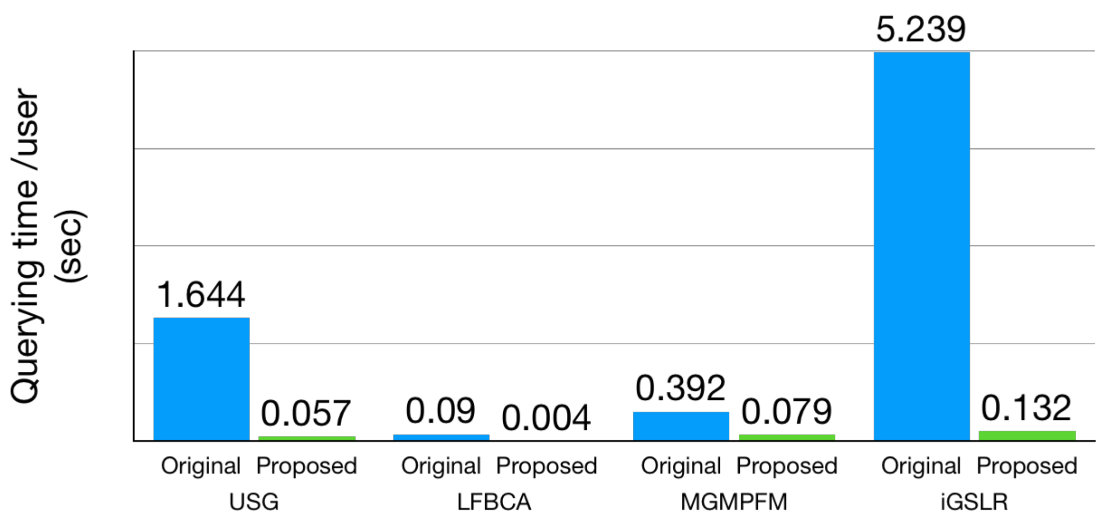

Figure 6 Querying time on Yelp dataset

The acceleration effect and increase in accuracy are slightly more significant in the Yelp dataset. The overall performance of the proposed method is similar for two datasets with similar sparsity.

Theoretically, query processing of USG, LFBCA, and MGMPFM has linear time complexity with respect to the number of POIs [4]. We denote N as the number of POIs. USG and LFBCA have querying complexity $O(MN)$, where M is the number of users. The querying complexity of MGMPFM is $O((G_u + K)N)$, where G_u is the number of user's active centers and K is the dimension of the latent space. Thus, the querying time is decreased because the number of POIs in the subset is decreased. The acceleration of querying is achieved best for iGSLR because the query time of iGSLR increases cubically with respect to the number of user's checked POIs; thus the acceleration of querying is significant for iGSLR due to the decreased number of POIs.

The acceleration of training is achieved best for LFBCA, because LFBCA used Bookmark-Coloring algorithm to train the model and its complexity is $O(M^2)$, where M is the number of users. By dividing the whole dataset into small subsets, the number of users in each subset is much smaller. Thus, the training process of LFBCA can be accelerated much.

The increase in accuracy on LFBCA is the highest because the original LFBCA only considers user’s friendship and ignores geographical information. By clustering the POIs, POIs in a small area are assigned into the same subset, which essentially adds geographical consideration because POIs far away from user’s active area are filtered out. Thus, the accuracy of LFBCA can be increased much.

5. Conclusion

In this study, we accelerated the POI recommendation system's execution speed by proposing the clustering-based method. We proposed a division method to divide the dataset into small subsets and feed subsets into base POI recommendation algorithms. The division process is based on multi-round POI clustering; each round's clustering result is structured to a tree, whose node represents POIs belonging to the same set. The division process continues until the average coverage ratio reaches the threshold. The average coverage ratio is the percentage of covered users' checked POIs under a specified POI division. The users, check-in history, and social relationships are also divided according to the POI division result. The evaluation results show that our proposed method accelerates the base algorithms by 17 to 39 times while keeping accuracy almost the same.

Our future work will include improving the POI division by altering the clustering algorithm or dynamically choosing the number of clusters. We are also seeking to exploit temporal patterns and friendship information to improve the division.

Acknowledgement

I sincerely appreciate Professor Yamana for the guidance on my research. Professor Yamana has frequently communicated with me and provided valuable advice about the research. I also would like to thank all students in Yamana lab, especially senior students and REC group members. They have also helped me a lot.

Reference

[1] I. Portugal, P. Alencar, and D. Cowan, “The use of machine learning algorithms in recommender systems: A systematic review,” Expert Systems with Applications, vol. 97, 2018, pp. 205-227.

[2] X. Amatriain, and J. Basilico, “Recommender systems in Industry: A Netflix case study,” Recommender Systems Handbook, Springer, 2015, pp. 385- 419.

[3] Y. Liu, T. Pham, G. Cong and Q. Yuan, “An experimental evaluation of point-of-interest recommendation in location-based social networks,” In Proceedings of the VLDB Endowment, vol. 10, no. 10, 2017, pp. 1010-1021.

[4] Z. Sun, D. Yu,H. Fang, J. Yang, X. Qu, J. Zhang and C. Geng, “Are we evaluating rigorously? benchmarking recommendation for reproducible evaluation and fair comparison,” In Proceedings of Fourteenth ACM conference on recommender systems, 2020, pp. 23-32.

[5] G. Ference, M. Ye and W. Lee, “Location recommendation for out-of-town users in location-based social networks,” In Proceedings of the 22nd ACM international conference on Information & Knowledge Management, 2013, pp. 721-726.

[6] M. Ye, P. Yin, W. Lee, and D. Lee, “Exploiting geographical influence for collaborative point-of-interest recommendation,” In Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval, 2011, pp. 325-334.

[7] J. Zhang and C. Chow, “Geosoca: Exploiting geographical, social and categorical correlations for point-of-interest recommendations,” In Proceedings of the 38th international ACM SIGIR conference on Research and Development in Information Retrieval, 2015, pp. 443–452.

[8] D. Zhang, M. Li, and C. Wang, “Point of interest recommendation with social and geographical influence,” In Proceedings of 2016 IEEE international conference on big data, 2016, pp. 1070–1075.

[9] H. Wang, M. Terrovitis, and N. Mamoulis, “Location recommendation in location-based social networks using user check-in data,” in Proceedings of the 21st ACM SIGSPATIAL international conference on Advances in Geographic Information Systems, 2013, pp. 374–383.

[10] J. Zhang and C.Y. Chow, “iGSLR: personalized geo-social location recommendation: a kernel density estimation approach,” in Proceedings of the 21st ACM

SIGSPATIAL International Conf. on Advances in Geographic Information Systems, 2013, pp. 334-343.

[11] H.Gao, J.Tang, X.Hu, and H.Liu, "Exploring temporal effects for location recommendation on location-based social networks," In Proceedings of the 7th ACM conference on Recommender systems, 2013, pp. 93-100.

[12] C. Cheng, H. Yang, I. King, and M. R. Lyu, "Fused matrix factorization with geographical and social influence in location-based social networks," In Proceedings of the Twenty-sixth Conference on Artificial Intelligence, 2012, pp. 17-23.

[13] F. Yu, L. Cui, W. Guo, X. Lu, Q. Li, and H. Lu, "A category-aware deep model for successive poi recommendation on sparse check-in data," In Proceedings of the web conference 2020, 2020, pp. 1264-1274.

[14] P. Zhao, H. Zhu,Y.Liu, Z. Li, J. Xu, and V. Sheng, "Where to go next: A spatio-temporal LSTM model for next POI recommendation," In Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence, 2019, pp. 5877-5884.

[15] H. Jiao, F. Mo, and H. Yamana, "Evaluation of POI recommendation system beyond accuracy: Diversity, explainability and computation cost," In Proceedings of 18th Japan data engineering and information management (DEIM) forum, C4-4, 2020.

[16] D. Agarwal and M. Gurevich, "Fast top-k retrieval for model based recommendation," In Proceedings of the fifth ACM international conference on Web search and data mining, 2012, pp. 484-492.

[17] X. Jiao, Y. Xiao, W. Zheng, H. Wang, and Y. Jin, "R2SIGTP: A novel real-time recommendation system with integration of geography and temporal preference for next point-of-interest," In Proceedings of the World Wide Web conference, 2019, pp. 3560-3563.

[18] Q. Fan, L. Jiao, C. Dai, Z. Deng, and R. Zhang. "Golang-based POI discovery and recommendation in real time," In Proceedings of the 2019 20th IEEE International Conference on Mobile Data Management (MDM), 2019, pp. 527-532.

[19] Q. Wang, T. Chen, Z. Huang, and H. Wang, "Next point-of-interest recommendation on resource-constrained mobile devices," In Proceedings of the Web conference 2020, 2020, pp. 906–916.

[20] Y. Si, F. Zhang, and W.Liu, "CTF-ARA: An adaptive method for POI recommendation based on check-in and temporal features," Knowledge-Based Systems, vol.128, 2017, pp.59-70.

[21] J. Zhu, C.Wang, X. Guo, Q. Ming, J. Li, and Y. Liu, “Friend and POI recommendation based on social trust cluster in location-based social networks,” EURASIP Journal on Wireless Communications and Networking, 2019, vol 2019, no. 1:89, pp. 1-12.

[22] D. Massimo and F. Ricci, “Clustering users’ POIs visit trajectories for next-POI recommendation,” Information and Communication Technologies in Tourism 2019, 2019, Springer, pp. 3-14.

Publications

As first author:

Huida Jiao, Fan Mo, and Hayato Yamana, “Evaluation of POI recommendation system beyond accuracy: Diversity, explainability and computation cost,” In Proceedings of 18th Japan data engineering and information management (DEIM) forum, 2020, C4-4.

Huida Jiao, Fan Mo, and Hayato Yamana, “Point of Interest Acceleration using Clustering,” In Proceedings of 6th IEEE International Conference on Big Data Analytics (ICBDA 2021), 2021. *(accepted)*

As co-author:

Fan Mo, Huida Jiao, and Hayato Yamana, “Time Distribution based Diversified Point of Interest Recommendation,” In Proceedings of 5th IEEE International Conference on Cloud Computing and Big Data Analytics (ICCCBDA), 2020, pp. 37-44.

Fan Mo, Huida Jiao, Shun Morisawa, Makoto Nakamura, Koichi Kimura, Hisanori Fujisawa, Masafumi Ohtsuka, and Hayato Yamana, “Real-Time Periodic Advertisement Recommendation Optimization using Ising Machine,” In Proceedings of 2020 IEEE International Conference on Big Data (Big Data), 2020. *(accepted)*