Destination prediction by Markov Model based approaches and Multi-layer Perceptron

Yifan Li
Department of Electric Engineering and
Computer Science
York University

Khadijah Alroogi
Department of Electric Engineering and
Computer Science
York University

ABSTRACT

Destination prediction has a great importance in order to serve a variety of location- based applications, for example, analyzing road traffic conditions to help good urban traffic management. Although researchers have done a great amount of work proposing different models, it is always needed to make more accurate prediction. Thus, in this paper, we put forward two models to enhance the accuracy of predicting the destination of a vehicle based on its departure location and zero or more intermediate locations. The first model we propose, Group-level Markov model, is an up-dated version of Markov model which is widely used in location prediction. In our group model, we make use of the similarities between users to help the prediction process for an individual user. We build an iterative framework to train this model which converges with little time. The other model is Multilayer perceptron model with an embedding layer that is employed to combine nominal ID with numerical location information as a flexible and powerful method to simulate complex relations and patterns. Finally, we present our experiments that have been conducted on a real world data set to compare our proposed models with two state-of-art models: PMM and GMM. The results show that Group-level Markov model performs well as the best while the MLP was not so satisfactory. Finally, analyzing the result and future work are discussed.

1. INTRODUCTION

The development of positioning technology nowadays gives arise to plenty of trajectory data sets. For example, social applications like Twitter allow users to post their location in the form of check-in, which usually consists of a user identification, a time-stamp and related text. A series of check-in information forms a trajectory[13]. Moreover, the advanced traffic management system using surveillance infrastructures in highways and city streets makes it easy to track vehicles and pedestrians. In general, a trajectory is a time-ordered locations produced by a specific user.

Also, in modern sedentary environment, everyday people's displacement behaviors and have many similarities. For white-collar workers, they may go to work from home, shop at the nearest supermarket after work, and visiting the popular cinema is likely to appear in such a pattern, too. Those similarities make it possible to mine one's moving pattern based on his or her previous trajectories.

Destination prediction is one of the most discussed problems in the area of trajectory mining. The purpose of destination prediction is to find the possible destinations of a vehicle based on its departure location and zero or more intermediate locations. An efficient and accurate destination prediction algorithm has a great importance for users, owners of the trajectory data, traffic management departments, and advertising platforms. For example, if some advertisement platforms know where a user is going, then they can recommend shops and restaurants near the destination, and by this way they can target potential customers while saving time and expense[5]. Also, traffic management departments can schedule vehicles based on their destinations in advance to prevent traffic jam.

The most challenging part of destination problem is that we need to make prediction as early as possible, which means information is limited - we may only know several starting locations of a long trajectory and are required to make prediction among thousands of possible destinations. Another problem is about sparsity, a specific user may only have visited a small number of places in the city, and we need to react as normal when he visits a new place.

The issue of trajectory based destination prediction is already discussed in many different studies. First, some studies have taken into account a one main idea of using the external information, as vehicle ID or the travel time, to improve the quality of the prediction process. In particular [4] as the winners of the of the Kaggle-ECML/PKDD discovery competition on on taxi destination prediction, have used a multi-layer perceptrons neural network in order to predict the destination of a taxi based on the beginning of its trajectory and other meta-information like departure time, driver id and client information. Although this was the first-place solution in that competition, in general neural network outcomes are hard to interpret and cannot be used for better understanding of the characteristics of the dataset. Moreover,[7, 14] worked on same idea of merging Bayesian methods with some external information such as road status, traffic condition, trajectory length, departure time, and driving habits to predict destinations based historical trajectories.

On the other hand, some other researchers have proposed different new models to get high quality prediction. Particularly, in [8]a T-pattern decision tree was built to extract movement patterns and then predict based on finding the best matching path in the tree. Also, [10] introduced a nearest neighbor trajectory method based on a distance measures to identify the historical trajectory that is most similar to the current partial trajectory. Finally, most of the studies that conducted in the field to solve the problem have used probabilistic procedures to serve the prediction process through constructing probability models. In particular, the Markov model was the most to use in predicting destinations[12, 1, 3]. [12] builds a Markov model for each single user and make prediction for a user based on individual previous trajectories. However, this is only possible when a user seldom visit new place. Also, this model can suffer from great performance descent due to data sparsity, especially when dealing with problems containing large number of locations. [1] proposes a combination of individual trajectories and all-user-trajectories. That's to say, build a Markov model for each user based on this user's individual trajectories (referred as Personal Markov Model or PMM), and build a Markov model based on all trajectories (referred as Global Markov Model or GMM). When making predictions, combine PMM and GMM with some probability and get result. This algorithm, although performs well in most situations, can also be problematic sometimes. For example, when most users share identical pattern, this pattern will have great influence on the final result when making prediction for users in the remaining group who actually have different moving intention. [3] find a way to make prediction for a user with the assistance of similar users. This model considers the skewed data problem, but the performance is heavily depend on the method computing the similarity between different objects, and it's almost impossible to find a judgment method which fits all mobility patterns.
We propose two modified models in this paper, one is inspired by clustering strategy learned in class, and the other one is inspired by Word2Vector method in Natural Language Processing.
The clustering based model is an updated version of PMM and GMM mentioned above. We also consider grouping here, but different from [3], no judgment meth ods are used here. We break the prediction problem into two parts in this model: group-level Markov models learn the moving patterns of all groups, and self-adjustable grouping method to cluster users with similar mobility behavior. We found these two sub-tasks can influence each other. That's to say, well-performed Markov models can help to group users more accurately, and a better grouping can improve the performance of Markov models. Based on this discovery, we solve the problem under an iterative framework which alternates between training Markov models and grouping users.
The embedding based model is similar to Multi-layer Perceptron. In Natural Language Processing, embedding method is used to transfer nominal attibutes[9] to continuous values with the consideration of distance between attributes. Here we used embedding layer to transfer user ID into vector, and this vector is the input of MLP together with location information. One-hot output is supposed to specify the possible destination of this trajectory.
We summarize the main contributions of our work as follows:

- We put forward a group-level method which combine clustering with Markov model to solve destination prediction problem. This method avoids data sparsity while requires no external similarity computation method such as KD-divergence or Jaccard similarity.
- We propose an iterative framework to train Grouping Markov model. In grouping phase, the probability that a specific user belongs to a group will be adjusted according to parameters of Markov model to provide higher within-group consistence. In modeling phase, Markov parameters are updated based on the new grouping condition to give a higher prediction accuracy.
- We introduce an embedding layer to MLP to solve prediction problem with ID. Embedding ID with location vectors here frees us from paying extra attention to group related problems. Also, we can model more complicated relations between trajectory prefix and destination.
- We perform experiments with a real data set and compare the results with existing models PMM and GMM. Models in this paper outperformed baseline by providing more accurate prediction.

The rest of the paper is organized as follows: In Section 2, we give some basic concepts to help understanding the problem and the models, and address the target of this paper;In section 3, we introduce the Markov model; further, in section 4, PMM are discussed, including its principle, advantage, and how to apply it to out data set; Similarly, we introduce GMM in section 5; In Section 6, we introduce how Group-level Markov model works and give its iterative framework. In Section 7, we give the architecture of embedding-combining MLP. In Section 8, the results of experiments and evaluation are given. At last, we conclude the paper in Section 9.

2. PRELIMINARIES

In this section, we first introduce fundamental concepts used in following sections, and then give a brief definition of the problem to be solved in this paper.

Definition 1. **Location**. A location l is defined as a point or a region where the position of a user is recorded. A user may pass through a set of locations and may pass a location for several times.

Definition 2. **Trajectory**. A trajectory is a time-ordered locations sequence. A trajectory is produced by a user within a given time period. A trajectory T is in the form: $(l_1, l_2, ..., l_n)$.

Definition 3. **Prefix**. Given a trajectory $T = (l_1, l_2, ..., l_n)$, a prefix P is the first k locations in T, where $1 \leq k \leq n - 1$. Prefix information if the input when predict destinations. The length of prefix is randomly set by program.

Definition 4. **Candidate Destinations**. A location l_i is a candidate destination give prefix P, if a user can at last arrive at l_i after passing P.

Definition 5. **User group**. A user group contains users who share identical or similar mobility patterns. A user may belong to several groups with different probabilities sum up to 1.

Given the random prefix of a user's trajectory sequence, the aim of destination prediction is to find possible location(s) which the user will arrive at lastly with great possibility.

3. MARKOV MODEL

Markov model is designed to simulate stochastic system and uncover the transition rules. It has been deeply proved that Markov model performs ideally dealing with prediction problem [11, 2]. When predicting possible destinations, Markov model regards the mobility patterns as a discrete stochastic process. In our problem, a state in Markov model corresponds to a specific location, and state transition denotes the process of moving from a location to another. Let T be a trajectory of length n and $p(l_{n+1}|P)$ refers to the probability of arriving at l_{n+1} after prefix P. Every next location is computed with:

$$\begin{aligned} l_{n+1} &= \arg\max_{l \in L} p(l_{n+1} = l|P) \\ &= \arg\max_{l \in L} p(l_{n+1} = l|l_1, l_2, ..., l_n) \end{aligned}$$

where L is the set of all possible locations. This formula requires computing the transition probability from the whole preceding sequence towards the location we want to calculate. However, in reality, next location is only depend on few preceding locations.Intuitively, current location has largest influence the next location. Based on that, we can focus on most closed preceding locations rather than the whole prefix.

$$l_{n+1} = \arg\max_{l \in L} p(l_{n+1} = l|l|l_{n-m-1}, l_{n-m-2}, ..., l_n) \quad (1)$$

where m is called the order of Markov model, i.e. the number of preceding steps we suppose to have influence on the current location. Finally, we can compute the destination based on all previous locations we choose in that trajectory. In order to make prediction with m-ordered Markov model, we need to calculate following conditional probability first:

$$p(l_i|L_n^m) = \frac{\#(L_n^m, l_i)}{\#(L_n^m)} \quad (2)$$

where $\#L_n^m$ denotes the occurrence of L_n^m observed in trajectory set, and $\#(L_n^m, l_i)$ means the times that location l_i is observed right after L_n^m.

4. PERSONAL MARKOV MODEL

Personal Markov Model (PMM) was first proposed in [1]. PMM is a single users trajectory based model. This influenced by that fact that every moving object has its routine and characteristics. For example, people are not likely to drive to a grocery 30 or more kilometers away from their area. Thus, the main focus here is to predict relying on the individual patterns of each moving object using its own previous trajectories. for computing, let $P(l')$ denotes to the discrete probability of a moving object arriving at location of l', where $l_{L'}$ is a set of the distinct locations in the training trajectories, $N(l')$ is the occurrence of the destination l', and l is the possible prefix of l', then we have:

$$P(l') = \frac{N(l')}{\sum_{l \in Ll'} N(l)} \quad (3)$$

Therefore, for the training process, we iteratively train a variable-order Markov model with order m, where $1 \leq m \leq N$using the trajectories of a moving object. First, we sat the prefix set for every moving object using its own trajectories, and then we computed the probability distribution of the final location which is the destination. However, data sparsity is a problem of this model, and this shows why GMM is needed.

5. GLOBAL MARKOV MODEL

Global Markov Model (GMM) is a more generalized model forming collective patterns based on all trajectories [1]. It cares about all the available trajectories on order to find out what the global behaviors of all the moving objects is. This is based on the fact that moving objects are more likely to share similar movement patterns. For example, people who are traveling from location A to location B are often take the same path. Thus, the probability distribution of a destination of a moving object is relying on the immediately preceding N locations that the moving object has arrived at, so we have:

$$P(l'| < l_j, ...l_i >) = P(l'|S_i^N) \quad (4)$$

where S_i^N is the prefix set. Our training process consisted of training an order-N GMM in order to then train a variable-order GMM. Then, we used the longest suffix match method as an approach to utilize the variable-order GMM for prediction.

6. GROUP-LEVEL MARKOV MODEL

The biggest problem of PMM is data sparsity, we can not handle new locations for a single user. But as the development of positioning techniques and the increasing requirement to more accurate prediction, analysis among large area becomes necessary. GMM, although fixed the problem of data sparsity, can also suffer performance descent in skewed data. Therefore, we propose group-level model here, which can group similar users and make prediction in a higher level. The method of group-level Markov model is greatly inspired by the idea of kNN clustering. Here we also divide the whole problem into two subtasks: prediction part and grouping part. In prediction part, we employ Markov model for each group to make predictions for users belong to this group; In grouping part, we adjust group information based on posterior probability. Also, we build an iterative framework to train this model.

6.1 Group method

We group users based on their mobility mode, because mobility mode is a strong reflection of trip purpose. It is obvious that a single users may have several trip purposes. For example, a student may go to school in weekdays and interest club at weekends. Based on this observation, we take advantage of soft-grouping[13] technique which allow users to be classified into more than one group. Assume the trajectory set contains n different users and we assign these users into m groups, then the grouping matrix M can be:

$$M = \begin{pmatrix} p_{11} & p_{12} & ... & p_{1m} \\ p_{21} & p_{22} & ... & p_{2m} \\ ... & ... & ... & ... \\ p_{n1} & p_{n2} & ... & p_{nm} \end{pmatrix} \quad (5)$$

where $p_{ij}, (1 \leq i \leq n, 1 \leq j \leq m)$ denotes the probability of user i belongs to group j. To make a reasonable grouping,

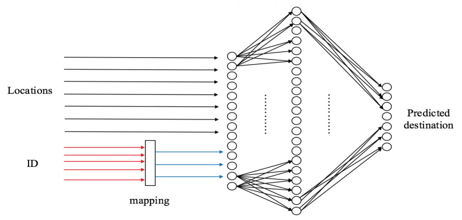


Figure 1: MLP architecture

the following constraints must be obeyed:

$$\sum_{j=1}^{m} p_{ij} = 1, i = 1, 2, ..., n \tag{6}$$

When making prediction to user i with probability array $(p_{i1}, p_{i2}, ..., p_{im})$, all groups will provide part of the final results independently. User k will influence the result in group j with probability $p_{ij} * p_{kj}$.

6.2 Iterative framework

To get a well-performed Group-level Markov predictor, we build an iterative process in which the Markov models and grouping will be enhanced alternately. The iterative process contains five steps generally:

- **Initialization**
  Let U be the set of users and G be the set of groups. $\forall u \in U$, randomly generate the initial probability array w.r.t $\sum_{g \in G} p_g(u) = 1$, where $p_g(u)$ denotes the probability that u belongs to g. Finally we get a probability matrix P.
  $\forall g \in G$, learn Markov model M_g will all trajectories and probability matrix and finally get a list of Markov models M.

- **Updating groups**
  $\forall u \in U$, compute the probability it belongs to every group based on the current Markov model list M and modify the probability matrix P accordingly to get a new probability matrix P^{new}.

- **Updating models**
  $\forall g \in G$, learn Markov models with the updated probability matrix P^{new} and get a new list of models M^{new}

- **Iteration**
  Let $P = P^{new}$, $M = M^{new}$.

- **Termination**
  Terminate when models converge. Else return to step 2.

We give detailed steps below: the area of Natural Language Processing, we come up with a method to transmit ID into continuous vector. The architecture of our model is showed in figure 1.

6.2.1 Markov model training and prediction

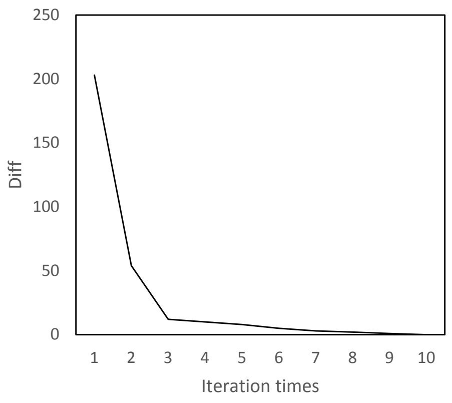


Figure 2: Learning curve

When training Markov model list M with probability matrix P, the Maximum Likelihood Estimation can be rewritten as:

$$p_g(l_i|L_n^m) = \sum_{u \in U} p_g(u) \frac{\#(L_n^m, l_i)^u}{\#(L_n^m)^u} \tag{7}$$

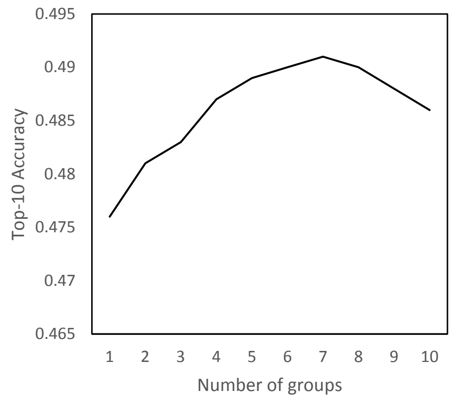
where $p_g(l_i|L_n^m)$ refers to the conditional probability of l_i as destination given L_n^m in group g; $\#(L_n^m, l_i)^u$ is the number of times that l_i as destination with a prefix L_n^m in trajectories produced by user u; and $\#(L_n^m)^u$ refers to the number of times that prefix (L_n^m) appears in trajectories produced by u. It is obvious from this formula that a user u with larger probability to group g will have greater impact on the final result of group g. Once known a prefix sequence L_n^m, the candidates chosen by group g is computed by equation (5). Note the candidate set is allowed to contain more than one location which is reasonable in real life because several trajectories produced by one single user may all cover some popular locations. Models from all groups will provide part of the final destination prediction according the probability matrix. Consider making prediction for user i on a prefix L_n^m. The prediction result is given in the form of a m-order array where each element in the array may be a list of candidate locations:

$$C = (p_{i1} * C(i), p_{i2} * C(2), ..., p_{im} * C(m)) \tag{8}$$

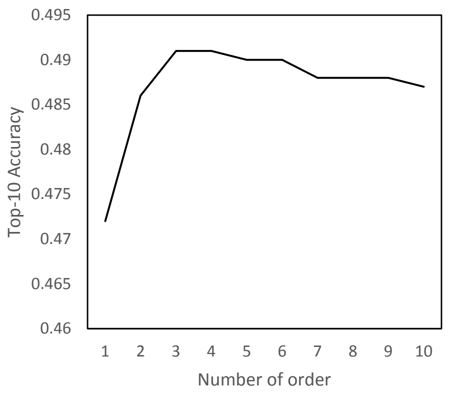
where $C(j)$ refers to the candidate set provided by model j. Note each candidate provided by model j is also marked with a probability, this probability is calculated by above formula. After calculating the prediction array, all locations will be ranked according to the final probability and top-k locations will be chosen as the final candidates. It is straightforward to conclude that, if user i belongs to group j with higher probability, then the candidate set provided by model j will have more chance to appear in the final candidate set.

6.2.2 User grouping

Once obtaining the updated Markov models, we modify grouping parameters accordingly. For every user in U, its probability array is updated so that for every dimendion g of G, $p_g(u)$ is the posterior probability that user u belongs to group g. To calculate this posterior probability, we first need to compute the probability of observing user u in



(a) Accuracy under different group numbers



(b) Accuracy under different order

Figure 3: Accuracy under different setting

group g. Assume the trajectory set of user u is T_u, then the probability of observing u in group g is:

$$p(u|g) = \prod_{t \in T_u} p(t|g) \tag{9}$$

where t is a single trajectory in T_u and $t = l_t^1, l_t^2, ..., l_t^r$. $p(t|g)$ is calculated by (assume the order is 1):

$$p(t|g) = p(l_t^1|g) * p(l_t^2|l_t^1, g) * ... * p(l_t^r|l_t^{r-1}, g) \tag{10}$$

where $p(l_t^1|g)$ and $p(l_t^s|l_t^{s-1}, g)$ can be calculated with equation (2). With Bayes' theorem, we can derive the posterior probability that user u belongs to group g as follows:

$$p(g|u) \propto p(g)p(u|g) \tag{11}$$

where $p(u|g)$ is given by equation (7) and $p(g)$ can be calculated from the previous probability. Therefore we get an updated probability matrix.

7. MULTI-LAYER PERCEPTRON WITH EMBEDDING LAYER

The key issue here ,as we talked about before, is how to combine nominal ID with numerical location. The idea we introduce here is inspired by the idea of Word2Vec[6] which is widely used in Embedding layer (mapping) transmits user ID into vector and this vector composes the input of MLP together with locations. The output of MLP is the predicted destination, which is further compared with the real destination, and the calculated error is back-propagated to weight matrix as well as embedding layer to adjust parameters. For embedding layer, the vector for a user is optimized each time. The working schema can be described as: when trajectory t produced by user i comes to the model, we first transmit all locations contained in the prefix of t (this prefix is randomly generated) into one-hot format and get location representative L, then we map user i with the i-th mapping entry in embedding layer to get a k-D vector V_i, which represents this specific user. L and V_i form the input of MLP. The output of MLP is the predicted destination which is also in one-hot format. Error of this instance is calculated with the output and real destination, and this error helps to adjust all weight matrix and V_i in embedding layer.

In this model, we do not need to pay extra attention to data sparsity problem and grouping problem. Any relations and similarities between users can be modeled automatically. Also, the expressive nature of MLP helps us to find more complicated patterns.

8. PERFORMANCE EVALUATION

We present experiments using real transportation data to evaluate the model. In this section, we first introduce the data set and experiment parameters, then the results of experiment will be given.

8.1 Data and evaluation method

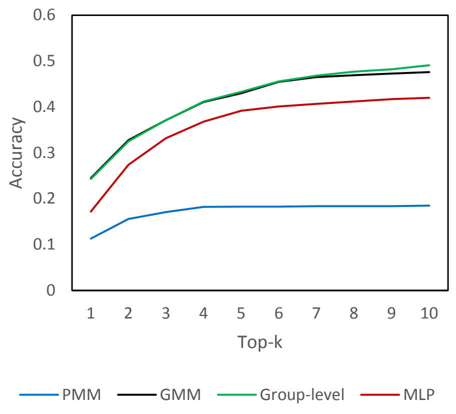
In this study, we exploit a real vehicle passage data set which is collected over the traffic surveillance system on a major metropolitan city. We are provided with $10,344,058$ records from the data center during a period of 31 days. Those trajectories contains 199 different locations and are produced by 437 users. Each record contains a vehicle ID and a location sequence represented by camera IDs. We first pre-process these data to pick up a random prefix. The purpose of pre-processing is to simulate real-world situation in which prediction may be required at any time and any location. After pre-processing, $746,790$ trajectories are employed to train our models, and the rest $104,129$ trajectories are used to test. In order to compare the performance of different models, two kinds of measurements are introduced: accuracy and average precision. Accuracy refers to the ration of correctly-predicted locations in all trajectories. That is:

$$accuracy = \frac{1}{|T'|} * \sum p(l) \tag{12}$$

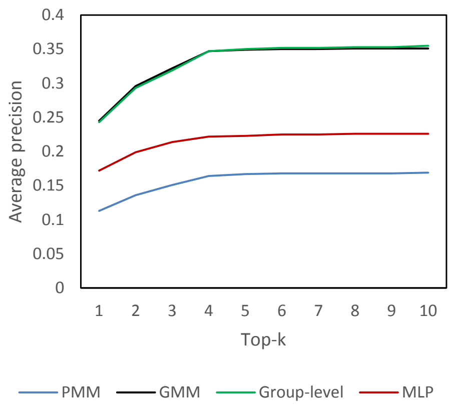
where $|T'|$ is the number of trajectories in test set and $p(l)$ is 1 if l is actually the destination and is 0 otherwise. Average precision is calculated by:

$$ap = \frac{1}{|T'|} * \sum \frac{p(l_i)}{i} \tag{13}$$

where i denotes the length of prefix sequence, and $p(l_i$ takes the value of 1 if the the prediction for this prefix is correct, and is 0 otherwise.



(a) Comparison of accuracy



(b) Comparison of average precision

Figure 4: Performance comparison

8.2 Parameter setting and performance

In group Markov model, the number of groups vary from 1 to 10 to exploit how the number of groups would impact the prediction.The number of iteration round we set is 100 which is big enough to see whether the model will converge or not. We use top-k ranked location to measure the accuracy and average precision of the model, where k ranges from 1 to 10.
In MLP, we use one-hot method which means every single location is represented by a 199-dimension vector. Due to its time-consuming training, we apply it to a small randomly-generated subset of data and record parameters leading to best performance. We use ReLu function for hidden layers and softmax for output layer, and cross-entropy error as penalty function. We adjust the following parameters in experiment: number of hidden layers, number of units in each hidden layer, vector dimension of embedding layer, as well as learning rate. We find the best performance is when MLP has: 1 hidden layer with 78 units, 10-D vector to represent vehicle ID, and learning rate 0.007. All performance showed below is under this setting.
The convergence process of Group-level Markov model is showed in figure 2. X-coordinate represents the iteration times and Y-coordinate refers to the difference of probability matrix between two adjacent iterations. We can see that the curve drops sharply at the first 3 iterations and almost hits 0 after 8 iterations, which proves the correctness of our model from the perspective of convergence.

The influence of group number is then being exploited. We compare top-10 accuracy under all group numbers respectively. The result is showed in figure 3(a). We can see from the figure that the best performance occurred when group number equals to 7. The performance drop slightly when group number continues increasing. This indicates users in our data set roughly follow 7 patterns. This is reasonable in real life because we have limited number of destinations given our departure location and all locations we passed by before current time. Note that mobility patterns which do not conflict with each other is possible to be classified into one group. For example, it is reasonable for pattern $1 : (A - B)$ and pattern $2 : (C - D)$ to be in the same group because pattern 1 will not influence the prediction made with pattern 2, i.e. they do not have locations in common.
The number of orders also has impact on the prediction accuracy in Markov model. We explore how the accuracy changes with the number of order under the setting of 7 groups. The result is showed in figure 3(b). We can see from the figure that the curve almost remains unchanged after the number of orders increases to 3. It proves that, at least in this data set, the next location can be largely decided by previous 3 locations. It also conforms to our knowledge that a trajectory has direction. However, this number may change in different data set due to the sample distance. For example, assume the sample distance of our data set is $2km$, the number is likely to become 6 in a data set with sample distance $1km$.
Then we compare the performance of all models mentioned before under optimized setting. That's PMM, GMM, Group-level Markov model and MLP. We use top-k (k-candidates) accuracy to compare the performance of each model and the result is showed in table 1 and figure 4. We can see from the table and the figure that:

- Accuracy increases in all models as k increases;
- PMM performs the worst because it only considers individual trajectories and suffers greatly from data sparsity;
- Group-level model performs as well as GMM when the candidate set is small and slightly better when candidate set becomes larger. This is because when more candidates are provided, the assistance of similar users will improve the accuracy, while in GMM the prediction for minority users may be influenced by other majority users who share different pattern with prediction target.
- MLP performs better than PMM but worse than GMM and Group-level Markov model. This is because the relative position in a trajectory is less considered in MLP. Markov models treat trajectory as an ordered sequence where the relative position matters, but MLP treats trajectory as a set and mixes all location together. This can be fixed by increase the units in input layer. For example, if we have a length-10 prefix as

(a) Accuracy of different models

Method	top-1	top-2	top-3	top-4	top-5	top-6	top-7	top-8	top-9	top-10
PMM	0.113	0.156	0.171	0.182	0.183	0.183	0.184	0.184	0.184	0.185
GMM	0.245	0.328	0.371	0.411	0.430	0.455	0.465	0.469	0.473	0.476
Group-level	0.243	0.325	0.371	0.412	0.433	0.456	0.468	0.477	0.482	0.491
MLP	0.172	0.274	0.332	0.368	0.392	0.401	0.407	0.412	0.417	0.42

(b) Average precision of different models

Method	top-1	top-2	top-3	top-4	top-5	top-6	top-7	top-8	top-9	top-10
PMM	0.113	0.136	0.151	0.164	0.167	0.168	0.168	0.168	0.168	0.169
GMM	0.245	0.296	0.322	0.347	0.349	0.350	0.350	0.351	0.351	0.351
Group-level	0.243	0.293	0.319	0.347	0.350	0.352	0.352	0.353	0.353	0.355
MLP	0.172	0.199	0.214	0.222	0.223	0.225	0.225	0.226	0.226	0.226

Table 1: Performance comparison

input, we can use $10 * 199 = 1990$ units in input layer and every 199 units represent a single location, rather than sum up those locations and use 199 units only. However, this will dramatically increase the number of parameters and cost much longer time to train. Another problem here is, because the prefix is randomly generated (to simulate real-world condition), different prefix may have different length. We keep this problem and will make further study to it.

9. CONCLUSION AND FUTURE WORK

In this paper, we proposed a Group-level Markov model and an MLP architecture with embedding layer. Group-level Markov model improve Markov model for destination prediction by introducing the idea of group which takes the advantage of similar objects to help the prediction. No external similarity measurement is used in Group-level Markov model, and thus is robust and independent from data distribution. We also built an iterative framework to train this model, which contains two part: training an individual Markov model for each group and classify each user into several groups according to its characteristic and the characteristics of different groups. MLP with embedding layer fixed the problem of data sparsity and can find the similarities between users automatically as well as model different relations and patterns. In MLP, we introduce embedding layer to transmit nominal ID to continuous vector which forms the input of MLP together with locations. We also evaluate our proposed models and two existing models: PMM and G-MM and showed Group-level Markov model out-performed other models. We also gave explanation to the experiment result.
Our future work mainly involves: 1)Optimize MLP to introduce relative position; 2)Find neighborhood relations in our data set and change our evaluation method to "distance with real destination", rather than "match or not"; 3) develop trajectory prediction applications based on best-performed model.

10. REFERENCES

[1] M. Chen, Y. Liu, and X. Yu. Nlpmm: A next location predictor with markov modeling. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 186–197. Springer, 2014.

[2] M. Chen, Y. Liu, and X. Yu. Predicting next locations with object clustering and trajectory clustering. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 344–356. Springer, 2015.

[3] M. Chen, X. Yu, and Y. Liu. Mining object similarity for predicting next locations. *Journal of Computer Science and Technology*, 31(4):649–660, 2016.

[4] A. de Brébisson, É. Simon, A. Auvolat, P. Vincent, and Y. Bengio. Artificial neural networks applied to taxi destination prediction. *arXiv preprint arXiv:1508.00021*, 2015.

[5] H. Gao, J. Tang, X. Hu, and H. Liu. Content-aware point of interest recommendation on location-based social networks. In *AAAI*, pages 1721–1727, 2015.

[6] Y. Goldberg and O. Levy. word2vec explained: Deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*, 2014.

[7] J. Krumm and E. Horvitz. Predestination: Inferring destinations from partial trajectories. In *International Conference on Ubiquitous Computing*, pages 243–260. Springer, 2006.

[8] A. Monreale, F. Pinelli, R. Trasarti, and F. Giannotti. Wherenext: a location predictor on trajectory pattern mining. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 637–646. ACM, 2009.

[9] A. Neelakantan, J. Shankar, A. Passos, and A. McCallum. Efficient non-parametric estimation of multiple embeddings per word in vector space. *arXiv preprint arXiv:1504.06654*, 2015.

[10] D. Tiesyte and C. S. Jensen. Similarity-based prediction of travel times for vehicles traveling on known routes. In *Proceedings of the 16th ACM SIGSPATIAL international conference on Advances in geographic information systems*, page 14. ACM, 2008.

[11] Q. Wu, M. K. Ng, and Y. Ye. Cotransfer learning using coupled markov chains with restart. *IEEE Intelligent Systems*, 29(4):26–33, 2014.

[12] G. Xue, Z. Li, H. Zhu, and Y. Liu. Traffic-known urban vehicular route prediction based on partial mobility patterns. In *Parallel and Distributed Systems (ICPADS), 2009 15th International Conference on*, pages 369–375. IEEE, 2009.

[13] C. Zhang, K. Zhang, Q. Yuan, L. Zhang, T. Hanratty,

and J. Han. Gmove: Group-level mobility modeling using geo-tagged social media. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1305–1314. ACM, 2016.

[14] B. D. Ziebart, A. L. Maas, A. K. Dey, and J. A. Bagnell. Navigate like a cabbie: Probabilistic reasoning from observed context-aware behavior. In *Proceedings of the 10th international conference on Ubiquitous computing*, pages 322–331. ACM, 2008.