Social Community Recommendation based on Large-scale Semantic Trajectory Analysis Using Deep Learning

Chaoquan Cai
Vanderbilt University
USA
chaoquan.cai@vanderbilt.edu

Wei Jiang
Oracle Labs
USA
wei.wj.jiang@oracle.com

Dan Lin
Vanderbilt University
USA
dan.lin@vanderbilt.edu

ABSTRACT

The widespread use of smart mobile devices has resulted in a massive accumulation of trajectory data by service providers. The analysis of human trajectories, particularly semantic location information, has opened up avenues for discovering common social behavior and enhancing social connections, leading to a range of applications such as friend recommendations and product suggestions. However, the exponential growth of trajectory information generated every day presents significant challenges for existing trajectory analysis algorithms, which are no longer capable of delivering timely analysis results. To address this issue, we propose a highly efficient algorithm that can recommend social communities for new users in real time by leveraging knowledge gained from large-scale semantic trajectories. Specifically, we develop a novel two-branch deep neural network model that extracts semantic meanings at different levels of granularity from human trajectories and uncovers the hidden relationship between trajectories and social communities. We then utilize this model to perform instant social community recommendations. Our experimental results have demonstrated that our approach is not only significantly faster than traditional trajectory analysis algorithms in terms of social community recommendation, but also preserves high prediction accuracy with F1-score above 97%.

KEYWORDS

Social Community Recommendation, Deep Learning, Semantic Trajectory Analysis, Convolutional Neural Network

ACM Reference Format:

Chaoquan Cai, Wei Jiang, and Dan Lin. 2023. Social Community Recommendation based on Large-scale Semantic Trajectory Analysis Using Deep Learning. In *Proceedings of 18th International Symposium on Spatial and Temporal Data (SSTD).* ACM, Canada, 10 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

As the Internet of Things continues to develop and society becomes increasingly digitized, service providers are collecting more and more human trajectory data through smart devices, smartphones, and vehicles equipped with GPS. By analyzing human trajectories, particularly semantic location information, we are gaining a better understanding of common social behavior, which has led to the development of a variety of interesting applications such as friend recommendations, topic recommendations, product recommendations, and location-based advertising. The foundation of all these recommendations is to identify a person's preferences in different settings, which can be effectively achieved by identifying the social communities a person belongs to. A social community is a group of people who share common interests. For example, as illustrated in Figure 1, Alice and Bob are both business travelers who enjoy seafood; Alice, Bob, and Carl all work for IT companies; Daisy and Ethan are tourists. People who are in different countries and have different GPS trajectories may belong to the same social groups, and a single person may also join multiple social groups. Thus, identifying the potential social communities that a new user may be interested in would be beneficial for both the user and the service providers. In the past, such social community recommendation was typically conducted based on the new user's profile, which, however, was usually not very accurate as most users are not willing to spend time filling out lengthy questionnaires to report all of their preferences. Thus, recent research efforts [5, 20, 26, 28] have focused on using semantic trajectory information to automate social community identification.

A semantic trajectory differs from traditional coordinate-based trajectories. While coordinate-based trajectories simply record the exact locations a person has visited, semantic trajectories record the types of places a person has visited. What is fascinating is that people who have no intersection of their coordinate-based trajectories may still possess similar social behavior patterns. Take Alice and Bob, for example, who live in two different cities and have nothing in common in terms of their daily coordinate-based trajectories. However, from their semantic trajectories, we can infer that both of them may share an interest in Italian restaurants, as they have frequently visited them. This demonstrates that analyzing semantic trajectories is much more useful for inferring one's preferences and interests than traditional coordinate-based trajectories. With the increasing prevalence of smart devices and GPS, semantic trajectory analysis has the potential to unlock a wealth of information about human behavior and preferences.

Social community recommendation based on semantic trajectory analysis is a very challenging research question. Despite the potential benefits of this approach, there are few studies in this field [5, 20, 26, 28], as it presents a myriad of challenges that are yet to be addressed. With the rapid growth of trajectory data generated daily, existing trajectory analysis algorithms are struggling to keep up with the pace required to provide timely analysis results for large-scale datasets. Most existing approaches rely on clustering

SSTD, 23-25 August, 2023, Calgary, Alberta, Canada

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

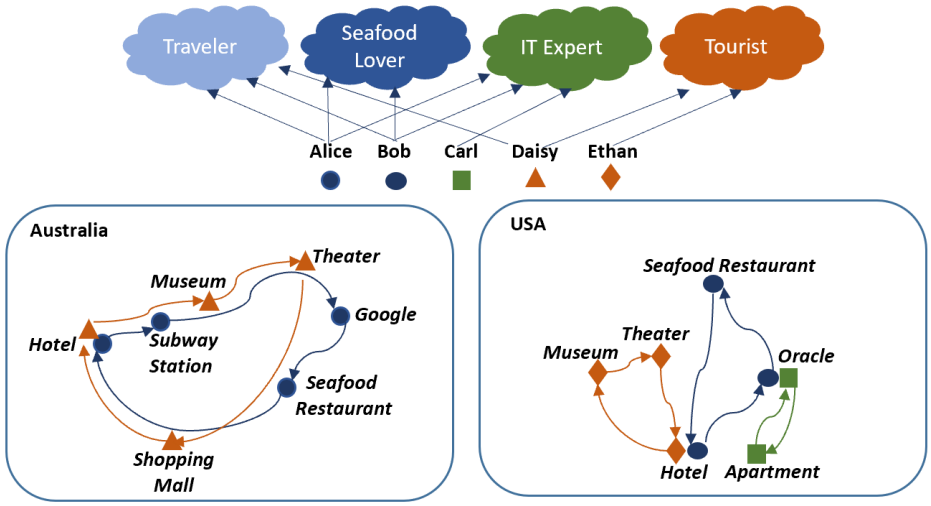

Figure 1: Social Community Recommendations Based on Semantic Trajectories

similar semantic trajectories into social groups, which is a daunting task given the sheer volume of trajectory data. As a result, identifying a new user's social groups requires comparing their trajectory with a massive number of existing trajectories, which is extremely time-consuming. Even the most recent distributed Spark-based algorithm [5] requires a whopping 6.3 hours for a single user social community recommendation on a dataset with 1 million trajectories. Addressing these challenges and developing more efficient algorithms for social community recommendation based on semantic trajectory analysis represents an exciting and important research direction.

We have developed a novel solution to tackle the aforementioned ever-growing scalability issue that accompanies the proliferation of newly connected data. Our approach is capable of recommending social communities for new users in real time, irrespective of the semantic trajectory dataset's size. Our strategy is a significant departure from traditional approaches that rely heavily on trajectory comparisons. Instead, we have harnessed the power of deep learning to construct a sustainable model that captures the hidden relationships between semantic trajectories and their corresponding social communities.

Our approach is speedy, requiring a mere 0.3 milliseconds to complete recommendations. To accomplish this, we designed a two-branch deep neural network, known as the Gated Recurrent Unit with K-Sequential Shingling (GRU-KSS), which extracts multi-level semantic meanings from semantic trajectories, paving the way for community recommendations. To the best of our knowledge, this is the first instance where a deep neural network model is proposed to handle social community recommendation. We summarize our contributions as follows:

- Our proposed GRU-KSS network represents a new way towards understanding the intricate relationship between input semantic trajectories and output social communities. We have developed a novel encoding method that efficiently converts semantic trajectories into effective vector inputs for the deep neural network. Furthermore, our two-branch structure captures and combines the semantic similarity at different granularity, resulting in a sophisticated model capable of revealing previously unrecognized connections between semantic trajectories and social communities.
- Our model is also dynamic and can handle the situations where new social communities are added. This flexibility allows our model to adapt to the ever-evolving landscape of social communities, ensuring that users are always receiving the most up-to-date and relevant recommendations.
- Our approach delivers incredible scalability, efficiency, and accuracy. Unlike traditional approaches that suffer from processing delays that increase exponentially with larger trajectory datasets, our approach ensures constant processing times for social community predictions. Furthermore, our model outperforms existing deep neural network models, achieving an impressive accuracy rate of 97% as confirmed by extensive experimental studies

The remainder of the paper is organized as follows. Section 2 reviews the related work. Section 3 presents the problem statement. Section 4 presents our community recommendation algorithms. Section 5 reports the experimental results. Finally, Section 6 concludes the paper and outlines future research directions.

2 RELATED WORK

As our work is related to both trajectory similarity analysis and social community recommendation, we review these two lines of works in the following.

2.1 Trajectory Similarity Analysis

There are two main representations of trajectories: (i) coordinate-based trajectories and (ii) semantic trajectories. As our work falls under the second category, we will briefly review the coordinate-based trajectory analysis and then focus our discussion on semantic trajectory analysis.

A coordinate-based trajectory is a sequence of recorded GPS locations of a person, while a semantic trajectory is a sequence of place names visited by a person. There have been a large body of works on analyzing and clustering coordinate-based trajectories [1, 10, 23, 27, 29, 36, 37]. The similarity between coordiante-based trajectories are typically measured using Euclidean distance [17, 18] or road network distance [15, 16], along with some other types of measures such as Hausdorff and Frechet distance, angular metric and edit distance [1, 8, 10, 27].

Unlike coordinate-based trajectory analysis which is focused on location proximity and bound by their geographic regions (e.g., within same cities), semantic trajectory analysis opens up opportunities to explore similar social behavior patterns in trajectories that may or may not be geographically similar and even across the geographic boundaries. As illustrated in Figure 1, similar coordinate-based trajectories may have totally different underlying semantic meanings, whereas dissimilar coordinate-based trajectories may actually exhibit similar social behavior patterns. Various approaches have been proposed to identify similarity between semantic trajectories. As for semantic trajectories, the commonly used similarity measure is the longest common subsequence. Zhang et al. [35] and Choi et al. [12] propose mining algorithms to detect sequential patterns in semantic trajectories, which are able to group users who visited similar types of places rather than just nearby locations. Wan et al. [32] take into account both coordinate-based distance and semantic-based distance, and define a new concept of semantic-geographic similar trajectories. Celik and Dokuz [7] introduces a new notion of socially important locations by integrating the frequency of the locations visited on social media platforms. Gao et al. [13] employ clustering approaches to identify regions of interests and then represent trajectories in multi-resolutions. Majority of existing works on semantic trajectory similarity analysis are centralized approaches which are not scalable to deal with the exponential collection of data. Therefore, the most recent work started looking into distributed algorithms. For example, Chen et al. [9] propose to group trajectories based on semantic meanings first and then spatial proximity. As a result, they will obtain smaller datasets to process the trajectory comparison in parallel. However, this distributed algorithm is not suitable for identifying semantically similar trajectories which do not reside in same geographic region as shown in our examples. Most recently, Cai et al. [5] propose the AnotherMe algorithm which leverages the Spark platform to dramatically improve the efficiency of trajectory similarity calculation compared to the previously centralized approaches. However, given a new user, these distributed algorithms still need to compare the new user's trajectory against all the existing users, which could be millions or billions of users, to find out their similarity, and hence they are inevitably computational expensive. Our deep-learning-based approach is much more resilient to the growth of the data collection and has the advantage of providing instant response in real time by simply feeding the new user's information through the trained network. We will demonstrate the superiority of our approach later in the experiments.

2.2 Social Community Recommendation

Social community recommendation or friend recommendation is a popular feature on the fast growing online social community. Recommendations can be based on various types of user information such as users' profiles [31] and users' trajectories [2, 5, 6, 26]. As our work is using trajectories for recommendation, we review the related work as follows.

Recently, researchers have begun to leverage the deep learning models for trajectory classification to make social community recommendations. Many of them utilize Rrecurrent Neural Networks (RNN) [3, 14, 19, 30, 34]. Besides RNN-based approaches, Liu et al. [25] develop a GRU-based trajectory classifier by introducing an additional temporal gate to better capture the spatial and temporal information in trajectories. Kontopoulos et al. [21, 22] convert the coordinate-based trajectories into images and then employ Convolutional Neural Networks (CNN) to classify the trajectories. However, as later shown in our experiments, these RNN, GRU or CNN based models all have lower F1 score than our approach when applied to semantic trajectories. This is because these existing approaches are mainly for coordinate-based trajectories and have not taken into considerations of non-contiguous transitions of locations in semantic trajectories. Some recent works [4, 20, 28] consider both semantic and geographic similarity between trajectories for classification. However, these approaches are not suitable for our scenarios where users with totally different coordinate-based trajectories may still be grouped into the same social community if their semantic trajectories are similar.

3 PROBLEM STATEMENT

In this section, we formally define the social community identification problem. As we leverage semantic trajectories to detect social communities, we first introduce the common definition of the semantic trajectory [5].

Definition 3.1. (Semantic Trajectory) A semantic trajectory of a user u is in the form of T_u=[$loc_1 \rightarrow loc_2 \rightarrow ... \rightarrow loc_n$], where loc_i ($1 \leq i \leq n$) denotes the name of the place where user u stayed for more than τ time (a time parameter used to distinguish passing points from stay points), and the arrows indicate the visiting order.

The following are some examples of semantic trajectories.

T_{Alice} = [Bridge Hotel→Sydney Station → Google Australia → Wharf Seafood →Bridge Hotel].

T_{Bob} = [Courtyard→ Oracle → Lobster Chef →Courtyard].

T_{Carl} = [Doolittle Apartment→ Oracle→ Doolittle Apartment].

We adopt the same definition of similarity between a pair of semantic trajectories as that in the most recent work [5].

Definition 3.2. (Semantic Trajectory Similarity) Given a pair of semantic trajectories T_{u_1} and T_{u_2}, the semantic trajectory similarity between them is calculated as follows:

$ST(T_{u_1}, T_{u_2}) = \sum_{i=1}^{n} [w_i \cdot S(T^i_{u_1}, T^i_{u_2})]$,

where $T^i_{u_1}$ and $T^i_{u_2}$ represents the trajectory location representation at level i in a semantic forest, w_i is the corresponding weight with $\sum_{i=1}^{n} w_i = 1$, and $S(T^i_{u_1}, T^i_{u_2})$ denote the maximum matching subsequence of location representations at level i.

The semantic forest used in the above definition is a hierarchical structure that organizes location information according to their

semantic relationship. The root of each tree corresponds to the highest level of abstraction of a location, such as transportation, entertainment, health, etc. The lower the level in the hierarchy, the more specific category the location belongs to. For example, the leaf level of the tree stores the exact address of each location; the parent of the leaf level stores the type of the locations such as bus station, 3-star hotel, children hospital. The number of levels in the semantic forest is adjustable based on the specific application domain, and it is treated as background knowledge for the semantic trajectory comparison. For example, given the semantic forest shown in Figure 2, the 2nd level representation of the above three example trajectories are the following.

T^2_{Alice} = [3-star Hotel→Subway Station → IT Company → Seafood Restaurant →3-star Hotel].

T^2_{Bob} = [3-star Hotel→ IT Company → Seafood Restaurant →3-star Hotel].

T^2_{Carl} = [Apartment→ IT Company→ Apartment].

The adoption of such hierarchical representation of a trajectory allow us to capture the in-depth semantic similarity since people who did not visit the exactly same places may still share commonality of their travel patterns. From the example, we can observe that although Alice and Bob are in two different countries, their trajectory representations at the 2nd level of the semantic tree exhibit high level of similarities and we may even speculate that they were likely on business trips to IT companies and both preferred seafood. Also, the similarity score between Alice and Bob is likely higher than that between Bob and Carl. Even though Bob and Carl visited the same company, the maximum common subsequences in each level of their trajectories' representations are as short as one.

Given the above definition of semantic trajectory similarity, we are now ready to define the notion of social community. The goal of finding a social community is to help people connect with other who may share common interests with them. In this work, we achieve this by identifying people who are highly similar to each other in terms of their semantic trajectories. The formal definition of the social community is as follows.

Definition 3.3. (Social Community) Given a set of semantic trajectories SC= $T_{u_1}, T_{u_2}, ... T_{u_k}$, they form a social community if any of two trajectories have the semantic similarity score above a threshold ρ, i.e., $\forall T_{u_i}, T_{u_j} \in SC, ST(T_{u_i}, T_{u_j}) \geq \rho$.

It is worth noting that a person may belong to multiple social communities. For example, {Alice, Bob, Carl} could be in the same IT community; {Alice, Bob} may also be in the seafood lover community.

4 OUR ALGORITHM

As discussed earlier, recommending social communities for a user using traditional approaches will require the calculation of the mutual semantic similarity between the user and all the other users on the social network. Considering the billions of users on social network nowadays, this task is an undoubtedly computational expensive. Thus, the goal of our work is to dramatically improve the efficiency of social community recommendation via deep learning techniques. Our key idea is to build a sophisticated deep neural network that is capable of modeling the convoluted relationship between semantic trajectories and social communities. Once the model is constructed, it will take the semantic trajectories of a user as input, and quickly predict the social communities that the user may be interested in joining.

There are two major challenges to conquer in the model design. First, we need an effective mapping algorithm which can convert the multi-level representations of a semantic trajectory into a single embedding that can be processed by a neural network. Second, we need to figure out appropriate types of layers, the number and size of the layers that best capture the meaning of semantic trajectories and yield high prediction accuracy. In what follows, we elaborate the detailed algorithms with respect to these two aspects.

4.1 Generation of Trajectory Embedding

In the raw dataset, a semantic trajectory is a sequence of place names. We convert each semantic trajectory into two types of embeddings: (i) hierarchical embedding; (ii) k-sequential shingle embedding. The hierarchical embedding corresponds to the representations of the semantic trajectory at different levels of the semantic forest. The k-sequential shingle embedding aims to capture the non-contiguous transition relationship among places. The detailed algorithms to generating the embeddings are the following.

There are four steps to generate the hierarchical embedding. First, we create a vocabulary of all the location names in the entire semantic forest as shown in Figure 2, and assign each name a unique integer index. Note that even though a location name may consist of multiple words, the name of the location is considered as a whole when assigning the integer index. This is because when we compare addresses or types of the locations, we are matching the entire addresses and types. Then, for each input semantic trajectory, we map the trajectory to all the levels in the semantic forest, and convert the locations to the corresponding indexes in the vocabulary. Next, we add a special symbol $'OOV'$ to the end of each trajectory to make its length equal to the maximum possible length (denoted as N) of trajectories being considered. Finally, we concatenate the N-dimensional index representations at all the L semantic levels into a $(N \times L)$-dimensional vector to form the hierarchical embedding. The format of the hierarchical embedding of a trajectory after multi-level semantic mapping and vectorization is as follows: $E_h = \begin{bmatrix} V^1 \\ V^2 \\ . \\ . \\ . \\ V^L \end{bmatrix}$ where $V^l = [v^l_1, v^l_2, ..., v^l_N]$, $1 \leq l \leq L$.

Reconsidering the example of Alice's trajectory, her corresponding hierarchical encoding is shown in Figure 3 calculated based on the semantic forest in Figure 2.

The hierarchical embedding preserves the contiguous transition relationship between the locations that are next to each other in a trajectory. For example, Alice transited from her hotel to a subway station, which is a contiguous transition. Alice later visited an IT company. Her transition from her hotel to the IT company is an example of non-contiguous transition. Observing Bob's trajectory,

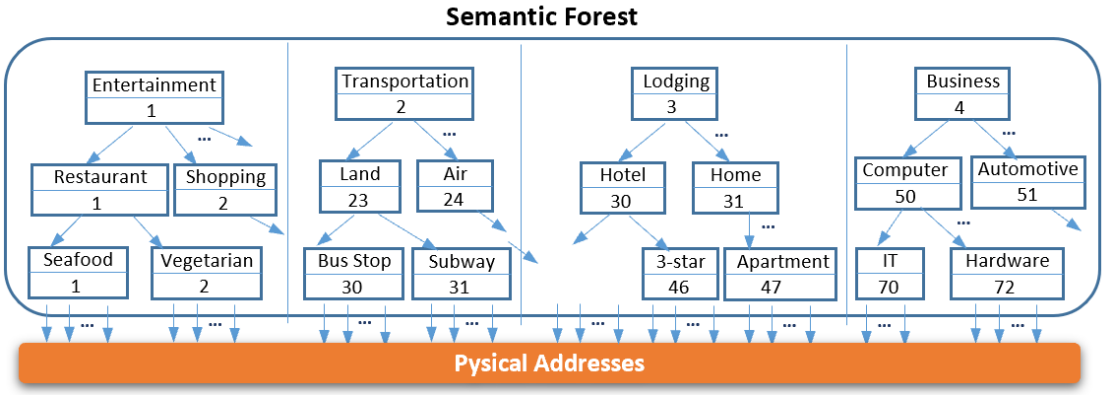

Figure 2: An Example of the Semantic Forest

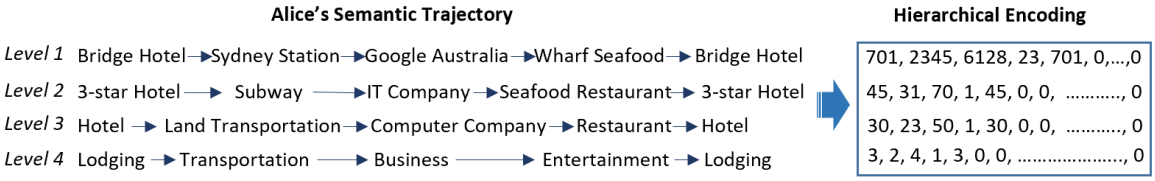

Figure 3: The Hierarchical Encoding of Alice's Trajectory

we may notice that Bob has a contiguous transition from his hotel to an IT company. If we only compare the hierarchical embeddings of Alice's and Bob's trajectories, we may miss the matching between contiguous transitions and non-contiguous transitions. Therefore, we introduce the k-sequential shingle embedding. A k-sequential shingle represents a combination of k locations in the trajectory. The locations in the k-sequential shingle are in the same order as that in the original trajectory but they may be contiguous or non-contiguous. Its formal definition is below.

Definition 4.1. (K-sequential Shingle) Given a trajectory $T=[loc_1 \rightarrow loc_2 \rightarrow ... \rightarrow loc_n]$*, its k-sequential shingle is a set of* m *distinct shingles* $G = \{g_1, g_2, ..., g_m\}$*, where each shingle is formed by* k *contiguous or non-contiguous locations in* T *in the same order as they occur in the trajectory, i.e.,* $g_i=[loc_{i_1} \rightarrow loc_{i_2} ... \rightarrow loc_{i_k}]$*, and* $1 \leq i \leq m$*,* $m \leq \binom{n}{k}$.

Given Alice's and Bob's trajectory representations at the 2nd level of the semantic forest, we may obtain the following two sets of 3-sequential shingles for them. From these k-sequential shingles, we can observe several exact matches: $g_{A2} = g_{B1}, g_{A3} = g_{B2}, g_{A4} = g_{B3}$, $g_{A5} = g_{B4}$. These indicate that Alice and Bob have highly similar moving patterns.

$G_{Alice} = \{g_{A1}, g_{A2}, g_{A3}, ..., g_{A10}\}$
g_{A1}= ⟨3-star hotel, subway station, IT company⟩
g_{A2}= ⟨3-star hotel, IT company, seafood restaurant⟩
g_{A3} = ⟨3-star hotel, IT company, 3-star hotel⟩
g_{A4}= ⟨3-star hotel, seafood restaurant, 3-star hotel⟩
g_{A5} = ⟨IT company, seafood restaurant, 3-star hotel⟩
...

$G_{Bob} = \{g_{B1}, g_{B2}, g_{B3}, g_{B4}\}$
g_{B1}= ⟨3-star hotel, IT company, seafood restaurant⟩
g_{B2} = ⟨3-star hotel, IT company, 3-star hotel⟩
g_{B3} = ⟨3-star hotel, seafood restaurant, 3-star hotel⟩
g_{B4} = ⟨IT company, seafood restaurant, 3-star hotel⟩

Actually for each semantic level of the trajectory representation, there is a corresponding set of k-sequential shingles. If we consider the shingles at all the semantic levels, it would be inevitably computational and storage expensive. As a tradeoff, we only generate the shingles at the second level of the semantic forest. This level contains fewer location categories and hence helps reduce the total number of shingles to be calculated for each trajectory, while this level still captures high level similarity of trajectories. To generate embeddings for the k-sequential shingles at the second level, we first create a shingle vocabulary for location categories at this level. Specifically, we generate all possible combinations of k-sequential shingles using the keywords at this semantic level, and assign a unique integer index to each shingle. Let M denote the total number of location categories in this level. The total number of combinations k-sequential shingles is M^k. In the real world scenario, the number of location categories at high semantic level would not be too large. Assuming there are 100 location categories, the size of the 3-sequential-shingle vocabulary will have 1 million integers which can be easily retrieved with the aid of a hash map to generate embeddings. Given the maximum trajectory length of N, the number of distinct shingles per trajectory will not exceed $\binom{N}{k}$. For example, when a trajectory consists of 10 location categories, the total number of 3-sequential shingle for this trajectory

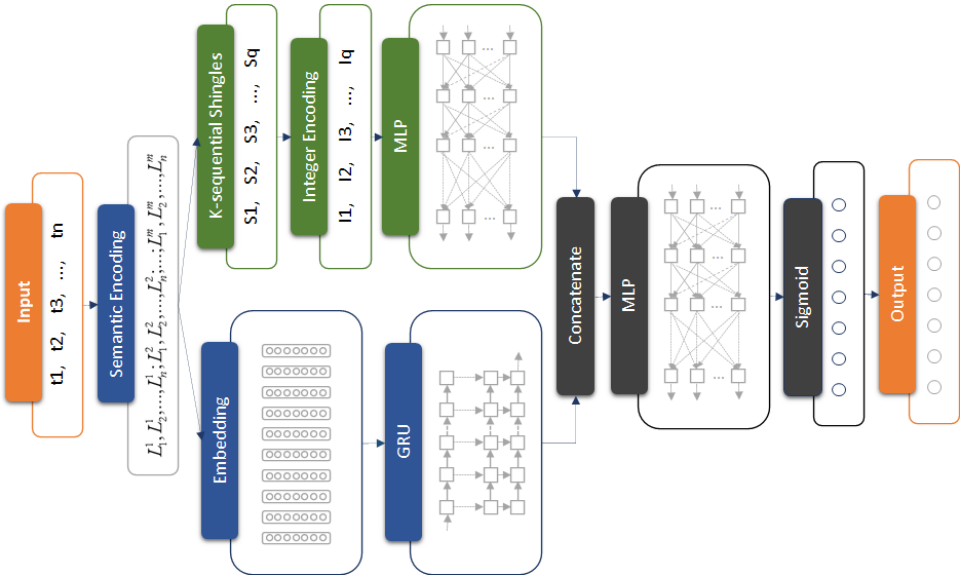

Figure 4: Overview of the Proposed GRU-KSS Model

is $\binom{10}{3}$=120. That means the corresponding 3-sequential shingle embedding will be a 120-dimensional vector of shingle indexes. As for a single trajectory to include 10 different location categories is already quite rare, we use this as the upper bound for padding the shingle embeddings. Specifically, if a trajectory's shingle embedding contains less than 120 indexes, we will add special symbols to the end of the embedding to make it 120-dimensional. The format of the k-sequential-shingle embedding of a trajectory is as follows: $E_s = [I_1, I_2, ..., I_q]$, where q is the maximum number of possible k-sequential shingles in a trajectory.

4.2 GRU-KSS Model

The goal of our model is to recommend social communities for users by analyzing their semantic trajectories. As a user may belong to multiple social communities which could be correlated to different portions of his/her trajectory, we are dealing with a multi-class classification problem. As the transition order of places being visited may indicate different types of user behavior or preferences, our model is expected to process data sequences. Taking into account these factors, we develop a Recurrent Neural Network (RNN) based model as shown in Figure4.

Our model is called GRU-KSS (Gated Recurrent Units & K-Sequential Shingling) model which consists of two branches. One branch aims to analyze contiguous location transitions in the semantic trajectories and it leverages the GRU [11] to handle the hierarchical embeddings of semantic trajectories. The other branch aims to analyze the non-contiguous location transition information and it takes the k-sequential-shingle embeddings as input. We elaborate the mechanism in each branch as follows.

GRU is a gating mechanism in recurrent neural networks. The GRU is like a long short-term memory (LSTM) with a forget gate, but has fewer parameters than LSTM and also perform better than LSTM for our problem. In our model, the GRU component is comprised of one layer with 256 nodes. It takes the hierarchical embedding of a semantic trajectory as input and produces a 256-dimensional feature vector denoted as f_c. Specifically, given a $(N \times L)$-dimensional hierarchical embedding E_h=[e_h^1, e_h^2, ..., e_h^{NL}], each node at the first hidden layer will compute the following, and the nodes at the later layers will conduct similar calculation by taking the input from the previous layer.

$$z_t = \sigma(w_z e_h^i + u_z h_{t-1} + b_z)$$
$$r_t = \sigma(w_r e_h^i + u_r h_{t-1} + b_r)$$
$$\tilde{h_t} = \tanh(w_h e_h^i + u_h(h_{t-1} \odot r_t))$$
$$h_t = (1 - z_t) \odot \tilde{h_t} + z_t \odot h_{t-1}$$

On the other side is the KSS branch which is a Multi-layer Perceptron (MLP). It contains two fully connected layers with 256 number of nodes per layer. It is fed with the k-sequential-shingle embedding and outputs a 256-dimensional feature vector denoted as f_n.

Next, we concatenate the obtained two feature vectors f_c and f_n and feed it to two fully connected layers. The intuition is to merge the features obtained from the contiguous location representation and the non-contiguous location representation to better characterize the user's behavior. In particular, the first fully connected layer contains 1024 nodes. The second fully connected layer contains the number of nodes equal to the maximum number of possible social communities in our experiments.

As we are predicting multiple social communities for each user, we apply the Sigmod activation function to produce the probability of each social community that a user may belong to. To measure the correctness of the prediction, we adopt the Binary Cross Entropy shown in Equation 1, where C is the total number of social communities, $p(y_i)$ denotes the predicted probability of the i^{th} social community that the user may be interested in, and y_i is the true

label which indicates whether the user really belongs to this social community. The true label y_i is set to 1 if the user is in the i^{th} community.

$$Loss = -\frac{1}{C}\sum_{1}^{C}(y_i \log p(y_i)) + ((1-y_i)\log p(1-y_i)) \quad (1)$$

5 EXPERIMENTAL STUDIES

Our GRU-KSS model is implemented by Keras and ran on NVIDIA GeForce RTX 2060 GPU. Our computer has 12 CPUs with Intel Core i7-9750H CPU(2.60GHz), 64G of memory and 1T disk space available. We compare our approach with a variety of existing algorithms as elaborated in the following. As we are predicting multiple social communities for each user, all of the models contain an output layer which has the number of nodes equivalent to the total number of social communities (denoted N_c) and uses the Sigmoid activation function at the end. Unless mentioned, otherwise the ReLu activation function is used for middle layers.

- MLP (Multi-Layer Perceptron): This is a simple fully-connected network that consists of two hidden layers and each hidden layer has 256 nodes. It takes only the hierarchical embedding as introduced in Section 4.1.
- RNN (Recurrent Neural Network): This is a vanilla RNN with a hidden layer that has 256 nodes and an output layer with N_c nodes. It also takes only the hierarchical embedding that represents contiguous location transition.
- RNN-KSS: This is a variant of our GRU-KSS model. We replace the GRU with the above vanilla RNN and keep the KSS branch the same.
- LSTM (Long Short-Term Memory): This LSTM has a hidden layer with 256 nodes. It takes only the one-dimensional hierarchical embeddings.
- LSTM-KSS: This is another variant of our model, whereby we replace the GRU with the LSTM.
- GRU (Gated Recurrent Unit): This GRU contains a single hidden layer with 256 units, and its input is the one-dimensional hierarchical embeddings.
- VGG16-1D: We modify the original VGG16 model by replacing its 2-dimensional convolutional kernel with a 1-dimensional convolutional kernel to take the one-dimensional hierarchical embedding. The stride is set to 1 and the size of kernel is 3.
- VGG16-2D: In this model, we turn the one-dimensional hierarchical embedding to a matrix to match the input format of the VGG16-2D. The stride is also set to 1 and the size of kernel is 3.
- VGG19-1D: Similar to the VGG16-1D, we modify the input layer in this model to take 1-dimensional hierarchical embedding.
- VGG19-2D: This model takes the same format of input as VGG16-2D.
- ResNet16-1D: This is not a standard ResNet network. In order to compare with the VGG network, we keep the main network structure the same as VGG16-1D and add skip connections between adjacent blocks in the VGG.
- ResNet16-2D: This is a variant of VGG16-2D but with skip connections.
- ResNet19-1D: This is a variant of VGG19-1D but with skip connections.
- ResNet19-2D: This is a variant of VGG19-2D but with skip connections.

In the experiments, we adopt both synthetic and real datasets. The synthetic dataset has 1 million trajectories. The length of each trajectory, i.e., the number of semantic locations visited, is set to 5 to 10 in order to match the common people's daily behavior patterns. The default semantic forest has 3 levels, including 10,000 semantic locations at the lowest level, 30 and 10 categories on the 2nd and root levels, respectively. As for the real dataset, we select the current largest real trajectory dataset – GeoLife [36, 38, 39]. The GeoLife dataset has 17,621 trajectories generated by 182 users. The trajectories in GeoLife are represented as a sequence of GPS coordinates. We convert these coordinates to semantic locations following the same steps as that in [5]. First, we detect stay points in the GPS trajectories based on the algorithm in [24, 33]. The stay points are the places where a person lingered for a period of time. Then we map the stay points to semantic location names to obtain corresponding semantic trajectories. Finally, we assign each converted semantic trajectory a new ID. For each dataset, we use 90% for training and 10% for testing.

Moreover, to be consistent with the latest distributed algorithm, AnotherMe, which is used as the baseline for comparison, we adopt the same setting for the selection of sequential shingles whereby 3-Sequential Shingling is used.

Let C denote the total possible number of social communities. We treat this multi-label classification task as a C binary classification task. The social communities generated by the traditional non-deep-learning approach [5] are used as the ground truth for comparison. The performance of all the models is evaluated using five commonly adopted metrics: CPU time, accuracy, precision, recall and F_1 score as defined below, where TP_i, TN_i, FP_i, and FN_i denote the total number of true positives, true negatives, false positives, false negatives of the i^{th} user, respectively. Each model is constructed 10 times on each training dataset. Each data point on the experimental figures represents the average performance of the 10 rounds.

$$Accuracy = \frac{1}{n}\sum_{i=1}^{n}\frac{TP_i + TN_i}{FP_i + FN_i + TP_i + TN_i} \quad (2)$$

$$Precision = \frac{1}{n}\sum_{i=1}^{n}\frac{TP_i}{FP_i + TP_i} \quad (3)$$

$$Recall = \frac{1}{n}\sum_{i=1}^{n}\frac{TP_i}{FN_i + TP_i} \quad (4)$$

$$F_1 = 2 * \frac{Precision * Recall}{Precision + Recall} \quad (5)$$

5.1 Comparison with Distributed Trajectory Analysis Algorithms

In the first round of experiments, we compare the performance of our deep-learning-based approach with the recent semantic trajectory analysis algorithm called AnotherMe [5] by varying the total

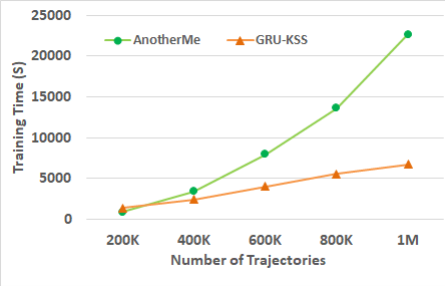

Figure 5: Training Time Comparison

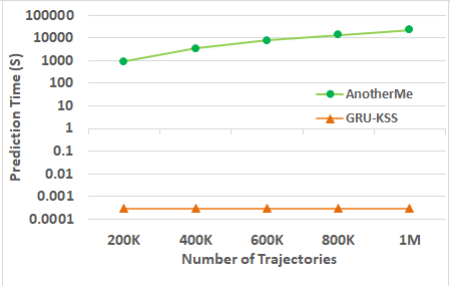

Figure 6: Prediction Time Comparison

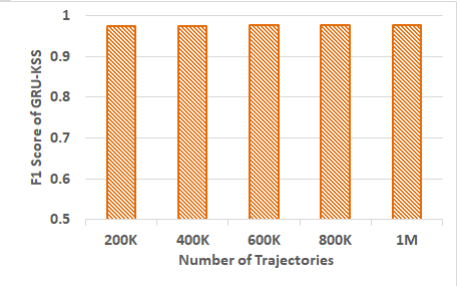

Figure 7: F1 Score Comparison

number of trajectories from 200K to 1M in the synthetic dataset. Recall that AnotherMe claims to be the up-to-date most efficient algorithm as it leverages Spark to conduct parallel trajectory similarity analysis and then clusters similar trajectories to form social communities. Figure 5 shows the construction time of the AnotherMe approach and the training time of our GRU-KSS model. Observe that the construction time of the AnotherMe approach increases rapidly with the growth of the total number of trajectories, and quickly surpasses the training time of the GRU-KSS model. Specifically, for the 1M trajectory dataset, our GRU-KSS model is more than 3 times faster than the AnotherMe approach.

Next, we examine the efficiency when using these two algorithms to predict social communities for new users. As shown in Figure 6, our GRU-KSS model significantly outperforms the existing distributed algorithm, i.e., AnotherMe. Our model outputs the social community recommendation less than a millisecond whereas the AnotherMe approach could take hours for a single recommendation. This is because given a new user's trajectory, AnotherMe needs to calculate the similarity between the new trajectory with all the existing trajectories to find the social communities that contain all the trajectories with similarity scores above a threshold. Such comparisons are extremely time consuming especially when the total number of trajectories increases. As for our GRU-KSS model, once it is constructed, the social community prediction is highly efficient by feeding the new trajectory through the model once.

To further validate the efficacy of our model, we also check the prediction accuracy by comparing the output social communities from our model and the AnotherMe algorithm. The social communities found by the AnotherMe algorithm are treated as ground truth. Figure 7 reports the F1-score of our model. Observe that the F1-score of our model is above 97% in all cases, which indicates the high prediction accuracy and recall of our model.

5.2 Comparisons with Existing Deep-Learning Models

After observing the dramatic prediction time reduction of our deep-learning model against distributed algorithms, we are interested in finding out how other existing deep-learning models perform in the same settings.

First, we evaluate the effect of our proposed hierarchical encoding when comparing with various deep learning models. Table 1 shows the results on 1M synthetic trajectories when the input to our model is hierarchical encoding whereas the input to other deep learning models is simply one level of encoding. Observe that existing models that take the single-level encoding of trajectories yield very low precision, recall and F1 scores. Our model significantly outperforms the existing models with the aid of the proposed hierarchical encoding.

Table 1: Results of Different Models on 1M Synthetic Trajectories without Hierarchical Encodings

Model	Accuracy	Precision	Recall	F1 Score
RNN	0.9768	0.0807	0.1618	0.1077
LSTM	0.9823	0.1887	0.4044	0.2573
GRU	0.9814	0.2098	0.4050	0.2764
VGG16-1D	0.9834	0.2706	0.5430	0.3612
VGG19-1D	0.9836	0.2599	0.4781	0.3368
ResNet16-1D	0.9819	0.2801	0.4530	0.3461
ResNet19-1D	0.9833	0.2448	0.4891	0.3263
GRU-KSS	**0.9993**	**0.9676**	**0.9804**	**0.9740**

Table 2: Results of Different Models on 1M Synthetic Trajectories with Hierarchical Encodings

Model	Accuracy	Precision	Recall	F1 Score
MLP	0.9906	0.7028	0.8350	0.7632
RNN	0.9850	0.3233	0.7035	0.4430
RNN-KSS	0.9854	0.4423	0.7860	0.5661
LSTM	0.9976	0.9328	0.9515	0.9421
LSTM-KSS	0.9984	0.9572	0.9635	0.9604
GRU	0.9981	0.9527	0.9539	0.9533
GRU-KSS	**0.9993**	**0.9676**	**0.9804**	**0.9740**
VGG16-1D	0.9965	0.8879	0.9194	0.9034
VGG16-2D	0.9952	0.8603	0.8770	0.8686
VGG19-1D	0.9952	0.8578	0.8779	0.8677
VGG19-2D	0.9973	0.9185	0.9354	0.9269
ResNet16-1D	0.9969	0.9112	0.9213	0.9162
ResNet16-2D	0.9964	0.8795	0.9201	0.8994
ResNet19-1D	0.9972	0.9110	0.9357	0.9232
ResNet19-2D	0.9967	0.8993	0.9221	0.9105

After observing the benefits of the hierarchical encoding on our model, we are interested in investigating whether the hierarchical encoding could help improve other existing models. Table 2 shows the comparison results between our GRU-KSS model and 14 other

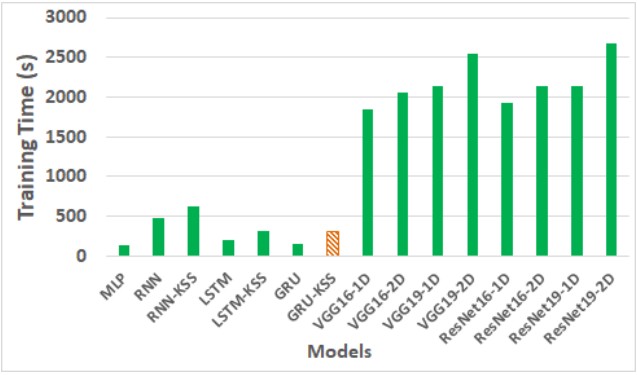

Figure 8: Training Time on 1M Synthetic Trajectories with Hierarchical Encodings

Table 3: Results of Different Models on Real Dataset with Hierarchical Encodings

Model	Accuracy	Precision	Recall	F1 Score
MLP	0.9948	0.6088	0.8436	0.7072
RNN	0.9921	0.4313	0.6932	0.5318
RNN-KSS	0.9945	0.5686	0.8532	0.6824
LSTM	0.9983	0.9250	0.9505	0.9376
LSTM-KSS	0.9988	0.9461	0.9732	0.9595
GRU	0.9979	0.9456	0.9525	0.9490
GRU-KSS	**0.9988**	**0.9690**	**0.9749**	**0.9719**
ResNet16-1D	0.9976	0.8366	0.9218	0.8771
ResNet16-2D	0.9977	0.8368	0.9355	0.8834
ResNet19-1D	0.9978	0.9256	0.9543	0.9397
ResNet19-2D	0.9983	0.8963	0.9381	0.9167
VGG16-1D	0.9966	0.7761	0.8833	0.8262
VGG16-2D	0.9967	0.7789	0.8929	0.8320
VGG19-1D	0.9975	0.8181	0.9276	0.8694
VGG19-2D	0.9979	0.8627	0.9251	0.8928

models (described in the experimental setting) on the synthetic dataset with 1 million trajectories. This time, all the models take the same hierarchical encodings of trajectories. Observe that our GRU-KSS model still performs the best in terms all the four metrics. Specifically, the F1 score of GRU-KSS is above 97%, which indicates the high prediction accuracy achieved by our model. Among the sequential models, the simple RNN model is least accurate and even worse than the MLP model, while LSTM and GRU perform similarly. After adding the KSS branch to these models, all of their F1 scores have been improved which could be up to 27%. This phenomenon validates the efficacy of the KSS branch which considers the non-contiguous location transmissions. As for the convolutional models, they are not as good as sequential models like LSTM and GRU. In general, the 2D versions perform better than the 1D versions. This is probably because 2D convolutional layers can better capture the correlations among different levels of semantic representations. Also, it is not surprising to see that adding the identity functions from ResNet helps further improve the overall accuracy.

Figure 8 compares the training time of all the models. Observe that the sequential models can be trained more than 10 times faster than the convolutional models, and our GRU-KSS model is among the fastest. This is because the convolutional models are deeper, and hence need more time to train.

Next, we also evaluate all the models using the real trajectory set. As shown in Table 3, again our GRU-KSS model yields the highest prediction accuracy among all. The training time also demonstrates the similar patterns as that in the synthetic dataset. For the sake of space, we omitted the figures here.

5.3 GRU-KSS

Finally, we evaluate how robust our model is under various scenarios such as different trajectory lengths, different semantic levels and different numbers of distinct locations. Figure 9 shows the F1 score of GRU-KSS when varying the average length of trajectories, i.e., average number of places in each trajectory from 8 t0 13. Observe that the changes in the trajectory length have very little impact on the model's prediction accuracy which stays around 97% in all cases.

Figure 10 shows the effect of the number of semantic levels. Observe that our model achieves similarly high prediction accuracy when the semantic levels are less than 5. When the semantic levels further increase to 6, the F1 score of our model dip below 90%. The possible reason is that when there are too many semantic levels, the representation of a single trajectory is further complicated as it is the combination of all the level presentations, which in turn increase the learning complexity of our model. Fortunately, in the real world applications, 3 or 4 levels of semantic hierarchy are more common and easier for adoption.

Figure 11 studies how the total number of locations may affect our model's performance. For that, we vary the distinct names of locations from 200K to 1M. As shown in the figure, the F1 score of our model is consistently above 97% regardless the number of distinct locations contained by all the trajectories.

To sum up, our GRU-KSS model can be trained very fast to achieve high prediction accuracy. It is robust under different scenarios and its prediction time meets the stringent real-time applications' needs.

6 CONCLUSION

In this paper, we present a novel approach to social community recommendation using a deep neural network model called Gated Recurrent Unit with K-Sequential Shingling (GRU-KSS). Our model uncovers previously hidden relationships between large-scale human semantic trajectories and corresponding social behavior, revolutionizing the way we understand and utilize trajectory data. GRU-KSS is designed with two branches that model contiguous and non-contiguous location transmissions, respectively. This innovative approach sets our model apart from traditional trajectory analysis algorithms, enabling us to deliver real-time social community recommendations with unparalleled accuracy and efficiency. Extensive experimental studies have confirmed the superiority of our model compared to adaptations of other deep learning models. With our model, users can expect to receive more personalized and relevant recommendations, making it easier than ever to connect with the communities that matter most to them.

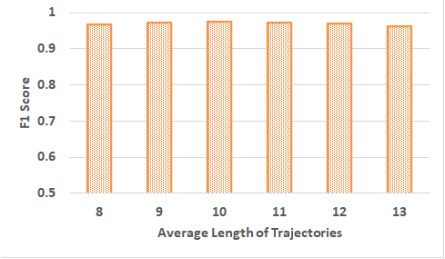

Figure 9: Varying the Average Length of Trajectories

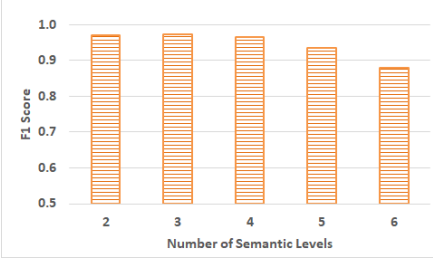

Figure 10: Varying Semantic Tree Levels

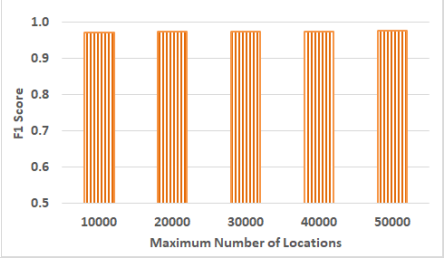

Figure 11: Varying Maximum Number of Locations

REFERENCES

[1] Helmut Alt. 2009. The computational geometry of comparing shapes. In *Efficient Algorithms*. Springer, 235–248.

[2] Sotiris Angelis, Konstantinos Kotis, and Dimitris Spiliotopoulos. 2021. Semantic Trajectory Analytics and Recommender Systems in Cultural Spaces. *Big Data and Cognitive Computing* 5, 4 (2021).

[3] Keywoong Bae, Suan Lee, and Wookey Lee. 2022. Transformer Networks for Trajectory Classification. In *2022 IEEE International Conference on Big Data and Smart Computing (BigComp)*. IEEE, 331–333.

[4] Lei Bao. 2020. A Trajectory Classification Model Using Grammar Parsing. *IEEE Access* 8 (2020), 218416–218423.

[5] Chaoquan Cai and Dan Lin. 2022. Find Another Me Across the World–Large-scale Semantic Trajectory Analysis Using Spark. *arXiv preprint arXiv:2204.00878* (2022).

[6] Yang Cao, Fei Xue, Yuanying Chi, Zhiming Ding, Limin Guo, Zhi Cai, and Hengliang Tang. 2020. Effective spatio-temporal semantic trajectory generation for similar pattern group identification. *International Journal of Machine Learning and Cybernetics* 11, 2 (2020), 287–300.

[7] Mete Celik and Ahmet Sakir Dokuz. 2018. Discovering socially similar users in social media datasets based on their socially important locations. *Information Processing & Management* 54, 6 (2018), 1154–1168.

[8] Lei Chen, M Tamer Özsu, and Vincent Oria. 2004. Symbolic representation and retrieval of moving object trajectories. In *Proceedings of the 6th ACM SIGMM international workshop on Multimedia information retrieval*. 227–234.

[9] Lisi Chen, Shuo Shang, Christian S Jensen, Bin Yao, and Panos Kalnis. 2020. Parallel Semantic Trajectory Similarity Join. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 997–1008.

[10] Yueguo Chen, Mario A Nascimento, Beng Chin Ooi, and Anthony KH Tung. 2007. Spade: On shape-based pattern detection in streaming time series. In *2007 IEEE 23rd International Conference on Data Engineering*. IEEE, 786–795.

[11] KyungHyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. *CoRR* abs/1409.1259 (2014). http://arxiv.org/abs/1409.1259

[12] Dong-Wan Choi, Jian Pei, and Thomas Heinis. 2017. Efficient mining of regional movement patterns in semantic trajectories. *Proceedings of the VLDB Endowment* 10, 13 (2017), 2073–2084.

[13] Chongming Gao, Zhong Zhang, Chen Huang, Hongzhi Yin, Qinli Yang, and Junming Shao. 2020. Semantic trajectory representation and retrieval via hierarchical embedding. *Information Sciences* 538 (2020), 176–192.

[14] Qiang Gao, Fan Zhou, Kunpeng Zhang, Goce Trajcevski, Xucheng Luo, and Fengli Zhang. 2017. Identifying Human Mobility via Trajectory Embeddings.. In *IJCAI*, Vol. 17. 1689–1695.

[15] Lasanthi Heendaliya, Dan Lin, and Ali Hurson. 2015. *Continuous Predictive Line Queries for On-the-Go Traffic Estimation*. 80–114.

[16] Lasanthi Heendaliya, Michael Wisely, Dan Lin, Sahra Sedigh Sarvestani, and Ali Hurson. 2015. Influence-Aware Predictive Density Queries Under Road-Network Constraints. In *Advances in Spatial and Temporal Databases*.

[17] C.S. Jensen, D. Lin, Beng Chin Ooi, and Rui Zhang. 2006. Effective Density Queries on ContinuouslyMoving Objects. In *22nd International Conference on Data Engineering (ICDE'06)*. 71–71.

[18] Christian S Jensen, Dan Lin, and Beng Chin Ooi. 2007. Continuous clustering of moving objects. *IEEE transactions on knowledge and data engineering* 19, 9 (2007), 1161–1174.

[19] Xiang Jiang, Erico N de Souza, Ahmad Pesaranghader, Baifan Hu, Daniel L Silver, and Stan Matwin. 2017. Trajectorynet: An embedded gps trajectory representation for point-based classification using recurrent neural networks. *arXiv preprint arXiv:1705.02636* (2017).

[20] Canghong Jin, Ting Tao, Xianzhe Luo, Zemin Liu, and Minghui Wu. 2020. S2N2: An Interpretive Semantic Structure Attention Neural Network for Trajectory Classification. *IEEE Access* 8 (2020), 58763–58773.

[21] Ioannis Kontopoulos, Antonios Makris, and Konstantinos Tserpes. 2021. A deep learning streaming methodology for trajectory classification. *ISPRS International Journal of Geo-Information* 10, 4 (2021), 250.

[22] Ioannis Kontopoulos, Antonios Makris, Dimitris Zissis, and Konstantinos Tserpes. 2021. A computer vision approach for trajectory classification. In *2021 22nd IEEE International Conference on Mobile Data Management (MDM)*. IEEE, 163–168.

[23] Jae-Gil Lee, Jiawei Han, Xiaolei Li, and Hector Gonzalez. 2008. TraClass: trajectory classification using hierarchical region-based and trajectory-based clustering. *Proceedings of the VLDB Endowment* 1, 1 (2008), 1081–1094.

[24] Quannan Li, Yu Zheng, Xing Xie, Yukun Chen, Wenyu Liu, and Wei-Ying Ma. 2008. Mining user similarity based on location history. In *Proceedings of the 16th ACM SIGSPATIAL international conference on Advances in geographic information systems*. 1–10.

[25] Hongbin Liu, Hao Wu, Weiwei Sun, and Ickjai Lee. 2019. Spatio-temporal GRU for trajectory classification. In *2019 IEEE International Conference on Data Mining (ICDM)*. IEEE, 1228–1233.

[26] Siyuan Liu and Shuhui Wang. 2016. Trajectory community discovery and recommendation by multi-source diffusion modeling. *IEEE Transactions on Knowledge and Data Engineering* 29, 4 (2016), 898–911.

[27] Tetsuya Nakamura, Keishi Taki, Hiroki Nomiya, Kazuhiro Seki, and Kuniaki Uehara. 2013. A shape-based similarity measure for time series data with ensemble learning. *Pattern Analysis and Applications* 16, 4 (2013), 535–548.

[28] Dianfeng Qiao, Yan Liang, Chaoxiong Ma, and Huixia Zhang. 2021. Semantic Trajectory Clustering via Improved Label Propagation With Core Structure. *IEEE Sensors Journal* 22, 1 (2021), 639–650.

[29] Amir Salarpour and Hassan Khotanlou. 2019. Direction-based similarity measure to trajectory clustering. *IET Signal Processing* 13, 1 (2019), 70–76.

[30] Rajiv Shah and Rob Romijnders. 2016. Applying deep learning to basketball trajectories. *arXiv preprint arXiv:1608.03793* (2016).

[31] Anna Squicciarini, Dan Lin, Sushama Karumanchi, and Nicole DeSisto. 2012. Automatic social group organization and privacy management. In *8th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom)*. 89–96.

[32] You Wan, Chenghu Zhou, and Tao Pei. 2017. Semantic-geographic trajectory pattern mining based on a new similarity measurement. *ISPRS International Journal of Geo-Information* 6, 7 (2017), 212.

[33] Xiangye Xiao, Yu Zheng, Qiong Luo, and Xing Xie. 2010. Finding similar users using category-based location history. In *Proceedings of the 18th SIGSPATIAL international conference on advances in geographic information systems*. 442–445.

[34] Di Yao, Chao Zhang, Zhihua Zhu, Jianhui Huang, and Jingping Bi. 2017. Trajectory clustering via deep representation learning. In *2017 international joint conference on neural networks (IJCNN)*. IEEE, 3880–3887.

[35] Chao Zhang, Jiawei Han, Lidan Shou, Jiajun Lu, and Thomas La Porta. 2014. Splitter: Mining fine-grained sequential patterns in semantic trajectories. *Proceedings of the VLDB* 7, 9 (2014), 769–780.

[36] Yu Zheng, Quannan Li, Yukun Chen, Xing Xie, and Wei-Ying Ma. 2008. Understanding mobility based on GPS data. In *Proceedings of the 10th international conference on Ubiquitous computing*. ACM, 312–321.

[37] Yu Zheng, Like Liu, Longhao Wang, and Xing Xie. 2008. Learning transportation mode from raw gps data for geographic applications on the web. In *Proceedings of the 17th international conference on World Wide Web*. ACM, 247–256.

[38] Yu Zheng, Xing Xie, Wei-Ying Ma, et al. 2010. GeoLife: A collaborative social networking service among user, location and trajectory. *IEEE Data Eng. Bull.* 33, 2 (2010), 32–39.

[39] Yu Zheng, Lizhu Zhang, Xing Xie, and Wei-Ying Ma. 2009. Mining interesting locations and travel sequences from GPS trajectories. In *Proceedings of the 18th international conference on World wide web*. 791–800.