Geographical Feature Extraction for Entities in Location-based Social Networks

Daizong Ding[1], Mi Zhang[1], Xudong Pan[1], Duocai Wu[1], Pearl Pu[2]
[1]Shanghai Key Laboratory of Intelligent Information Processing, School of Computer Science, Fudan University, China
{17110240010,mi_zhang,14302010024,14302010040}@fudan.edu.cn
[2]Human Computer Interaction Group, School of Computer and Communication Sciences, Swiss Federal Institude of Technology (EFPL), Switzerland
pearl.pu@efpl.ch

ABSTRACT

Location-based embedding is a fundamental problem to solve in location-based social network (LBSN). In this paper, we propose a geographical convolutional neural tensor network (GeoCNTN) as a generic embedding model. GeoCNTN first takes the raw location data and extracts from it a well-conditioned representation by our proposed Geo-CMeans algorithm. We then use a convolutional neural network (CNN) and an embedding structure to extract individual latent structural patterns from the preprocessed data. Finally, we apply a neural tensor network (NTN) to craft the implicitly related features we have obtained into a unified geographical feature.

The advantages of our GeoCNTN mainly come from its novel neural network structure, which intrinsically offers a mechanism to extract latent structural features from the geographical data, as well as its wide applicability in various LBSN-related tasks. From two case studies, i.e. link prediction and entity classification in user-group LBSN, we evaluate the embedding efficacy of our model. Results show that GeoCNTN significantly performs better on at least two tasks, with improvement by 9% w.r.t. NDCG and 11% w.r.t. F1 score respectively, using the Meetup-USA dataset.

CCS CONCEPTS

• **Social and professional topics → Geographic characteristics**;

KEYWORDS

Location-based Social Networks, Feature Embedding, Deep Learning

1 INTRODUCTION

Location based social network (LBSN) continues to gain popularity. In 2017, there are over 55 million monthly active users and over 12 billion[1] cumulative check-ins in Foursquare[2] alone. Yet there is still more room for these networks grow. LBSNs provide users with a multitude of information and services around their current locations, thus enriching their off-line daily lives. For example, Yelp provides restaurant advice based on a user's daily routine as well as his/her friends' check-ins. Examples of LBSN include Meetup, Yelp, and Facebook[3]. In such networks, various kinds of heterogeneous *entites*, i.e. users, groups, events, etc., interact at a given location in near-real time (Fig. 1). For this study, we take Meetup, a popular off-line group meeting facilitator website, as an illustrative example where users seemingly prefer to participate in meet-up groups not far away from their current residence. This behavioral pattern resonates with Tobler's first law of geography [38] that says near things are more related than distant things.

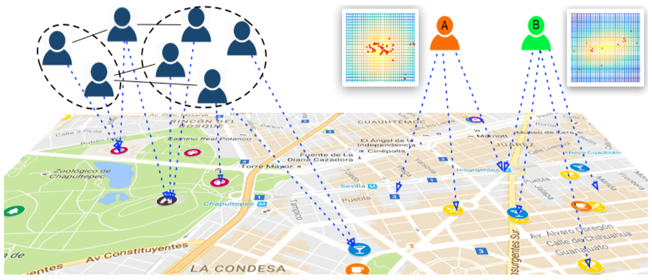


Figure 1: Location-based social networks.

Traditional tasks in social networks, such as link prediction [44, 48] and entity classification [32], are still the focal points in LBSN, while the difference lies in the additional geographical information brought by its innate setting. How to extract useful location-based features from the original geographical information largely determines the final performance of a specific learning model.

From our perspective, the following illustrative examples will present two indispensable aspects of a good geographical feature:

- Users living in Shanghai are considered to be more similar to each other than those living in Beijing, e.g., their preferences for spicy or sweet food. Inspired by Tobler's law, methods such as those presented in [34, 47] exploit such kind of similarity via the assumption that entities with intersecting regions of activities should have more similar features. For example, [23] takes advantage of the geographical neighborhood characteristics for location recommendation.
- Users exhibit different location patterns even though they all live in Shanghai. As is shown on the right part of Fig. 1, user A on the left has a more concentrated location pattern locally than user B's on the right. It seems A prefers to stay home most of the time while B enjoys hanging out in diverse

[1]https://www.statista.com
[2]https://foursquare.com

WWW 2018, April 23–27, 2018, Lyon, France

ACM ISBN 978-1-4503-5639-8/18/04.
https://doi.org/10.1145/3178876.3186131

[3]https://www.meetup.com https://www.yelp.com https://www.facebook.com

locations. It inspires us to obtain more expressive features via considering the characteristics in their location patterns.

In the former example, geographical distance still has a considerable impact on entity's behavior in LBSN. We refer to such a factor as a *global* factor. Moreover, the latter case illustrates that the difference inside location patterns locally may also play an essential role in determining entity's behavior, which is referred to as a *local* factor in this work. To our knowledge, most of the previous works such as [34, 47] are basically focused on exploiting the global geographical information, while, from our perspective, it is critical to combine the local information from location patterns with the global information as a complement.

A similar consideration has been introduced independently in several fields such as image processing [18] and time series analysis [21]. To the best of our knowledge, methods for combining global and local geographical information have not ever been proposed for location-based social networks, probably due to the difficulty of obtaining a well-conditioned representation as input from the raw location data, which is inherently sparse.

In this paper, we present a location-based embedding model called GeoCNTN, which allows us to obtain a unified feature from raw locations for each entity by crafting both global and local geographical information.

A brief description of our proposed method is presented here. Given a set of locations defined by their longitude and latitude for each entity, we first apply our fuzzy-clustering method (Geo-CMeans) on the curved surface of the earth. With fuzziness, we are able to represent *areas* where local patterns can be revealed without *fractures.* We then divide the global geographical coordinate system into grids, obtain the indices of grids where cluster centers are located, and reformulate each grid ID into a corresponding one-hot vector (Fig. 3(b)) as a global representation of locations. Simultaneously, matrices that represent the occurrences of locations in the gridded domain of each truncated cluster (Fig. 3(c)) are obtained as the corresponding representation for each local pattern. We then apply a 3-layer CNN for feature extraction from local patterns, which would otherwise suffer from extreme sparsity without our preprocessing procedures, and project global representations into a vector form as a global feature via embedding the geographical grids into vector space. In the final step, we combine the global and local features into a unified form with a neural tensor network (NTN) [37], which is well-known for its capability to capture higher order correlations among several implicitly related inputs.

Our contributions are three-fold:

- We propose a neural network based embedding structure called GeoCNTN, which adopts embedding table [33] and a convolutional neural network for extracting features from global and local geographical information, respectively. With application of neural tensor network, our model are able to craft the global and local features together, capturing their innate correlations.
- We develop a curvature-sensitive fuzzy clustering method called Geo-CMeans to obtain well-defined global and local representations from the original geographical data.
- Our generic model has a strong applicability for a wide range of LBSN-related tasks.

As a validation for our feature extraction model, we apply our model to both link prediction and entity classification tasks in LBSN. Compared with the state-of-art, our model can perform much better with a net improvement by 9% w.r.t NDCG in the former task and by 11% w.r.t F1 score in the latter one. We have also validated the interpretablility of the obtained features via a series of visualizations.

The rest of the paper is organized as follows. Section 2 describes the related work. In Section 3, we present our approach for obtaining well-conditioned representations of location patterns with our geographical fuzzy-clustering method, Geo-CMeans. In Section 4, we propose our GeoCNTN model for obtaining location-based embedding from global and local representations that we have obtained in Section 3. In Section 5, we present several applications with our generic model and conduct a series of experiments and visualizations. Section 6 concludes our work.

2 RELATED WORK

2.1 Discovering location patterns in LBSN.

Location information is innate in LBSN [42]. However, existing models often simplify a location with an integer ID rather than explicitly exploiting the geographical information, e.g., common places between entities [34, 47]. These methods are unable to analyze the influence of geographical relatedness, which means they in essence could not recognize location pattern for entities.

In fact, location pattern recognition has been a popular topic in recent years. Existing methods can be classified into three categories. The first kind is based on rules *à* priori. Methods such as extracting different statistical features in LBSN [6], e.g., entropy of places, model user's movement patterns based on the assumption that users may stay around their home and work places [5]. This kind of methods can only fit specific datasets that satisfy their prior assumptions, making them relatively weak for generalization. The second kind of methods is based on modeling location patterns with a specific probabilistic distribution on pairwise distances [40, 43] or a two-dimensional Gaussian for locations with coordinate representations [4, 22]. These methods may fail to extract personalized visiting pattern of their entities in LBSN because of a common probabilistic modeling for each user. The third kind argues that patterns of each user can be different [46] and should not be modeled as a common distribution, which leads to the application of the kernel density estimator method (KDE) to model the distribution of pairwise distance with a much more powerful extension into 2-dimension [45]. However, this family of methods may be overly complex and over-fitting can be a problem [20, 45].

As for the clustering of location patterns on a map, K-Means has been proposed to partition user's locations into several clusters [1]. However, as pointed out by [9], hard-clustering method like K-Means may be sensitive to noise points. On the contrary, density-based clustering (fuzzy clustering) can remove noise points by filtering out low probability points under some thresholds. For example, [4] proposes to use multi-center Gaussian model to cluster locations into different areas with soft boundaries. A fuzzy clustering algorithm that is able to find center and radius of clusters for users was proposed by [29]. However these methods did not take the curvature of the earth surface into account, which may led to the imprecision problem with a large geographical range setting.

2.2 Neural Network Architecture

Recently, neural networks have achieved significant success in media processing tasks, such as image labeling, speech recognition [18], as well as obtaining useful embeddings [3, 27, 28, 33]. Furthermore, the use of convolutional neural network has been shown to perform much better than most traditional methods in several tasks [13, 17, 24, 36].

A previously proposed architecture [21] is able to combine global and local information in time series analysis with a CNN for local pattern and long short term memory for global tendency. However, as far as we know, similar consideration has not been attempted for location pattern analysis, probably due to a lack of well-conditioned representations of both global and local location information. For example, a rough preprocessing may bring extremely sparse representations, which is unlikely for CNN to extract any useful features [7]. More specially, several recent studies have already found that neural network structures are very effective in discovering 1-dimensional location pattern. For example, [39] uses recurrent neural network to exploit temporal information of check-ins. However, to our knowledge, work using neural network architecture for extracting feature for entities in LBSN directly from 2-dimensional geographical information has not yet been proposed, unlike those with a similar consideration found in natural language processing (NLP) and image processing.

3 DATA REPRESENTATION

As pointed out by [9], geographical distributions of locations in LBSN tend to have the following characteristics: **(1)** Visited locations of a certain entity tend to cluster in several implicit centers. **(2)** Locations away from any of these cluster centers can be considered as noises.

From our perspective, the *noise* locations could be considered negligible in our context, not only because of their relatively loose connections to each cluster, but also their lack of representativeness w.r.t. the local location distribution patterns. Intuitively, these singular points highly likely exist at the margin of clusters. Such an approximation thus has rather slight effects on our analysis of global and local location distribution patterns for a given entity.

Based on these prior observations, we present an auxiliary concept called *Area* in LBSN, which plays an indispensable role throughout the discussion of our proposed model.

Definition 3.1. An *Area* is a truncated cluster representing the locations of a given entity in LBSN.

For a better understanding our *area* concept, consider a businessman who regularly travels to several cities. Although he makes abrupt visits while he is away, these visits may have little contributions to his global mobility distributions nor his local patterns in the city where he resides. We may thus consider each of these cities for business or every-day life an *area* related with him, except his unexpected adventures.

As a by-product of our definition, the concept of an *area* spontaneously separates information contained in the original set of locations into global and local information. Based on that separation, well-formed representations can be constructed rather naturally by globally *positioning* and locally *zooming*, which is the focal point of 3.2, 3.3.

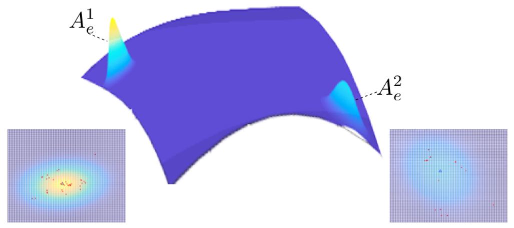


Figure 2: Geo-CMeans algorithm on curved plane.

From now on, the primary problem lies in how to detect a number of *areas* of a given entity without introducing unnecessary fractures of local patterns, while at the same time, filtering out negligible noise locations. As a strong candidate solution for fuzzy clustering on a curved plane, we develop a new algorithm called Geo-CMeans on the surface of the earth introduced by a large geographical range setting. Before an overall discussion of our method, we define the nomenclature.

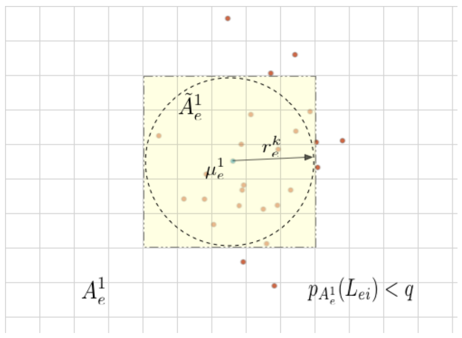
3.1 Notations

In the context of location-based social networks, the set of all entities is denoted as E. An entity $e \in E$ in a general social network should be augmented with additional geographical information, denoted by a set of visited locations $L_e = \{L_e^i\}_{i=1}^{n_e}$, where $L_e^i \doteq (\lambda_e^i, \phi_e^i)$ denotes the longitude and latitude of a certain visited location. Moreover, let [N] denote the set $\{1 \ldots N\}$, C denote the number of clusters, and a representation with footnote e means it belongs to entity e. A_e^k denotes a fuzzy cluster with $p_{A_e^k}$ as its induced probabilistic distribution function (p.d.f). $\tilde{A}_e^k$ is a truncated cluster from A_e^k, as we will call it *area* later. $\tilde{I}_e^k$ denotes the one-hot vector obtained from *area* $\tilde{A}_e^k$, while P_e^k is corresponding local pattern in matrix form. GF_e and LF_e respectively denote the global and local feature, while F_e is a unified feature learned by our model.

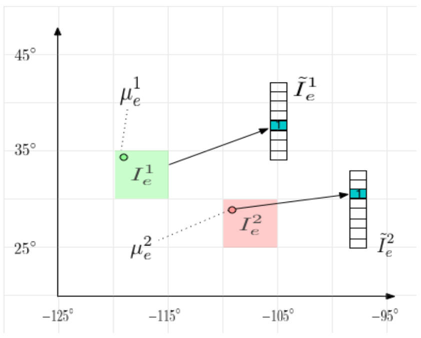
3.2 Geo-CMeans Clustering for Area Detection

In this work, in order to detect the set of areas $\{\tilde{A}_e^k\}_{k=1}^C$ for an entity e, we introduce a geographical fuzzy C-Means algorithm (Geo-CMeans) on the basis of the classical fuzzy clustering method C-Means [8]. As we know, a hard clustering methods, such as K-Means, tends to give rigid boundaries among a set of points, which may make otherwise whole visiting pattern *fractured* in our context. An illustrative example occurs when we apply K-Means with a large k to a small number of concentrated locations. It will then partition the pattern into several pieces, which makes the position of the center lost. With a fuzzy clustering method, we can innately avoid this potential problem.

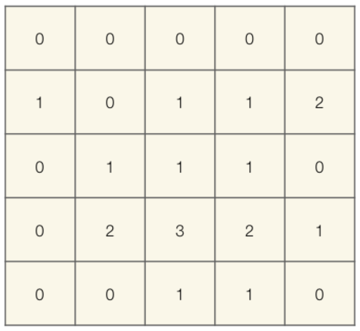
From another aspect, distance between points plays a critical role for any clustering methods. As the underlying geographical range gets larger, the geographical distance between locations cannot be calculated with an original Euclidean distance formula. Instead, we propose to use a common approximation of the intrinsic distance formula (Eq. 3) over the earth. An illustrative result of our algorithm can be found in Fig. 2.



(a) Area detection by Geo-CMeans.



(b) Global representation $\tilde{I}_e$.



0	0	0	0	0
1	0	1	1	2
0	1	1	1	0
0	2	3	2	1
0	0	1	1	0

(c) Local representation P_e.

Figure 3: Obtain global and local data representation of an *area*.

For a certain entity e, its location set is L_e containing n_e points, and a generic location $x_i = (\lambda_i, \phi_i)$. Suppose we want to find C areas for this entity, where each area has its center defined as $\mu_j = (\lambda_j, \phi_j)$, an extended loss function relative to Fuzzy C-Means can be formulated as

$$\min_{p,\mu} \sum_{i=1}^{n_e} \sum_{j=1}^{C} p_{ij}^{\epsilon} \cdot d(x_i, \mu_j), \ s.t. \ \sum_{j=1}^{C} p_{ij} = 1 \tag{1}$$

where p_{ij} represents the probability that point x_i belongs to area j, μ_j the center of area j , and $\epsilon \in (1, +\infty)$ (used for preventing degenerate solutions during optimization). To solve this optimization problem, we adopt Lagrange multiplier method with the Lagrangian defined as

$$\mathcal{L} = \sum_{i=1}^{n_e} \sum_{j=1}^{C} p_{ij}^{\epsilon} \cdot d(x_i, \mu_j) + \sum_{i=1}^{n_e} \eta_i (\sum_{j=1}^{C} p_{ij} - 1) \tag{2}$$

where η_i is the Lagrange multiplier. With an approximate intrinsic distance over sphere $d(x_i, x_j)$ defined as

$$d(x_i, x_j) = R\sqrt{\Big[(\phi_i - \phi_j)^2 + (\lambda_i - \lambda_j)^2 \cdot \cos^2(\frac{\phi_i + \phi_j}{2})\Big]} \tag{3}$$

where R is a positive constant independent of the choice of i, j, we are able to use the alternative direction multiply method (ADMM) to minimize $\mathcal{L}$.

After steps of algebraic manipulation, we obtain the updating rules in closed forms for λ, p as

$$p_{ij} = \Big[\sum_{k=1}^{C} \big(\frac{d(x_i, c_j)}{d(x_i, c_k)}\big)^{\frac{1}{m-1}}\Big]^{-1}, \lambda_i = \frac{\sum_{j=1}^{C} p_{ij}^{\epsilon} \frac{R}{d(x_i, c_j)} \cdot \lambda_j}{\sum_{j=1}^{C} p_{ij}^{\epsilon} \frac{R}{d(x_i, c_j)}} \tag{4}$$

while for ϕ_j, stationary point condition is written as

$$h(\phi_j^o) = \sum_{i=1}^{n_e} p_{ij}^{\epsilon} \frac{R}{2d(x_i, c_j)} \cdot \Big[2(\phi_i - \phi_j^o) + \frac{1}{2}\sin(\phi_i + \phi_j^o)\Big] = 0 \tag{5}$$

Without an explicit closed form, we apply the Newton method to solve this optimization problem with the second-order derivative written as

$$h'(\phi_j^o) = \sum_{i=1}^{n_e} p_{ij}^{\epsilon} \frac{R}{2d(x_i, c_j)} \cdot \Big[-2 + \frac{1}{2}\cos(\phi_i + \phi_j^o)\Big] \tag{6}$$

As a final comment, in order to accelerate the convergence of optimization, we set the initial value of ϕ_j^o as the average of $\{\phi_i\}$ from the observations $\{x_i\}$ as $\tilde{\phi}_j^o = 1/n_e \sum_{k=1}^{n_e} \phi_k$.

3.3 Positioning for Global Grid Representation

Each cluster $\{A_e^k\}_{k=1}^C$ we have obtained (Fig. 2) can thus be represented by each cluster center in the geographical maps, denoted by $\{\mu_e^k\}_{k=1}^C$ with a set of induced probability density function (p.d.f.) $\{p_{A_e^k} : \mathcal{E} \to [0, 1]\}_{k=1}^C$, where $p_{A_e^k}(L_{ei})$ is the probability that location L_{ei} belongs to A_e^k obtained from Geo-CMeans (Fig. 3(a)). Fig. 3(a) serves as a 2-dimensional projection of a specific cluster in Fig. 2, for the sake of better visualization.

With the consideration of the intensively large number of entities and thus clusters in our problem setting, it is intractable if we attempt to identify each one with a unique ID, which may otherwise cause overfitting during learning. A solution lies in the discretization of the whole geographical map into a two-dimensional grid system $[N] \times [N]$, the grid of which is uniquely identified, say with natural number set $[N^2]$, as demonstrated in 3(b). We then assign to each area the ID $I_e^k \in [N^2]$ of the grid where its cluster center lies and obtain the corresponding one-hot representations with length N^2 denoted by $\tilde{I}_e = \{\tilde{I}_e^k\}_{k=1}^C$. The set $\tilde{I}_e$ will be further used as the input of the embedding layer of our model in Section 4 (Fig. 4).

3.4 Zooming for Local Pattern Representation

In addition to the *positioning* practice we apply for the global information representation, we devise a sub-procedure called *zooming* for constructing a well-conditioned representation of local location distribution patterns in A_e^k, which will be discussed as follows.

Although the domain of each fuzzy cluster component A_e^k is the total $\mathcal{E}$, the density of distribution is featured by an extreme sparsity at the margin as shown in Fig. 2 & 3(a). It inspires us to truncate a cluster with a certain probability threshold $q \in (0, 1)$. For each location $L_e^i \in A_e^k$, we exclude it from the belonging component if $p_{A_e^k}(L_{ej}^k) < q$. After such a procedure is carried over all these components, we obtain the set of truncated cluster components as $\{\tilde{A}_e^k\}_{k=1}^C$, i.e. the set of areas for entity e.

In fact, we could interpret the $\tilde{A}_e^k$ that we obtained as a zoomed-in view of the original A_e^k, which ignores the negligible part of the domain. $\tilde{A}_e^k$ magnifies the view around the cluster center, which

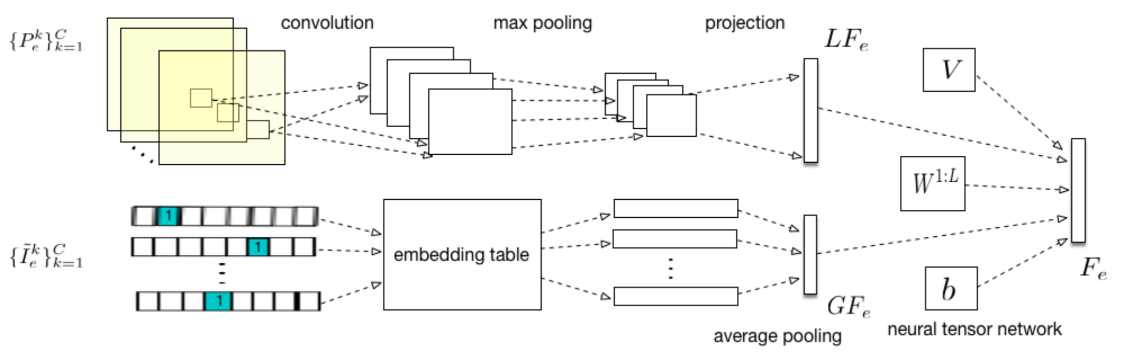


Figure 4: Convolutional Tensor Network for crafting global and local geographical information.

Algorithm 1 Constructing Global & Local Representations

```
Input: Location set $\{L_{ei}\}_{i=1}^{n_e}$ as longitude-latitude pairs for each entity $e$
Output: Extracted one-hot vectors $\{\tilde{I}_e^k\}_{k=1}^C$, matrices $\{P_e\}_{k=1}^C$
Initialize: $C$ areas, threshold $q$, global and local grid size $N, M$
1: Divide map into $N \times N$ grids, assign each grid a unique ID
2: function DATA_REPRESENTATION($L_e$)
3:     Apply Geo-CMeans algorithm in Sec. 3.2
4:     Obtain clusters $\{A_e^k\}_{k=1}^C$, centers $\{\mu_e^k\}_{k=1}^C$, p.d.f. $\{p_{A_e^k}\}_{k=1}^C$
5:     for $k = 1$ to $C$ do
6:         Initialize area $\tilde{A}_e^k$ as $\emptyset$
7:         Get the global grid ID $I_e^k$ where $\mu_e^k$ locates
8:         Convert the ID $I_e^k$ into a one-hot vector $\tilde{I}_e^k$
9:         for all location $L_{ei} \in A_e^k$ do
10:            if $p_{A_e^k}(L_{ei}) > q$ then add $L_{ei}$ to $\tilde{A}_e^k$
11:        Set $r_e^k = \max_{L_{ej} \in \tilde{A}_e^k} d(L_{ej}^k, \mu_e^k)$
12:        Set the center of square as $\mu_e^k$
13:        Set the side length of square as $2r_e^k$
14:        Divide the square into $M \times M$ grids.
15:        Initialize the $M \times M$ local pattern matrix $P_e^k$ as zeros
16:        for all location $L_{ei}$ in area $\tilde{A}_e^k$
17:            Find the grid where $L_{ei}$ is located
18:            Increment the corresponding count in $P_e^k$
19:        Add $\tilde{I}_e^k, P_e^k$ respectively to $\{\tilde{I}_e^m\}_{m=1}^{k-1}, \{P_e^m\}_{m=1}^{k-1}$
20:    return $\{\tilde{I}_e^k\}_{k=1}^C, \{P_e^k\}_{k=1}^C$
21: end function
```

is exactly the representation of an area on the global level of view. We define the radius of each area as $r_e^k = max_{L_e^i \in \tilde{A}_e^k} d(L_e^i, \mu_e^k)$. The outcome of the procedure above is illustrated in Fig.3(a).

We finally discretize the domain of each truncated components again as an integer-valued two-dimensional grid system $[M] \times [M] \rightarrow \mathbb{N}$, where $P_e(i, j)$ gives the count of L_e^i in $\tilde{A}_e^k$, as shown in Fig. 3(c). The center of each grid system P_e^k is aligned with the center of the corresponding area, while the geographical range represented by such a grid system is $2r_e^k \times 2r_e^k$, which corresponds to the common knowledge that the rescaling is independent of a local pattern. The set $\{P_e^k\}_{k=1}^C$ will serve as the input of CNN component in our model (Fig. 4). The procedure discussed in this section as a whole is depicted in Algorithm 1.

4 LOCATION-BASED EMBEDDING MODEL

After constructing a well-conditioned representation of the original set of locations L_e as a set of global one-hot vectors $\{\tilde{I}_e^k\}_{k=1}^C$ and the corresponding relatively dense local pattern matrices $\{P_e^k\}_{k=1}^C$, in this section we propose a neural network based structure (Fig 4) to merge these heterogeneous geographical representations of various shapes, scalings, and even interpretations into a unified vector-valued feature for entity e in a LBSN. As far as we know, our model is the first neural net based structure for capturing the complex entanglement between global and local geographical information.

4.1 Vector-Valued Embedding for Global Feature Modeling

First, let us consider the problem of embedding a set of global one-hot vectors $\{\tilde{I}_e^k\}_{k=1}^C$ into a vector GF_e representing the entity e's global feature. In the literature of modern embedding methods, a generic architecture proposed by [3] has a successful application in NLP for word embedding [27, 28]. Within our context, we were inspired by the representation compression power of such an efficient embedding structure. With the help of such embedding, we could compress the obtained one-hot representations by learning a distributed embedding for each global grid. The technical details are briefly explained as follows.

In the first layer we exploit the classical method proposed by [3] that introduces an embedding table to map the one-hot representation $\tilde{I}_e^k \in \tilde{I}_e$ into a vector, which has an intuitive interpretation as the vector-valued embedding of a global geographical grid. For each entity e, we obtain C vectors as its global embedding of the grids. By the subsequent projection layer, C grid embeddings will be combined into a single vector GF_e using average pooling, which serves as an extracted global feature as shown in the lower part of Fig. 4.

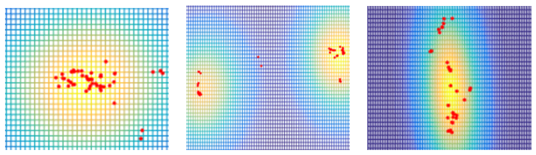
4.2 Convolutional Neural Network for Local Pattern Feature Extraction

Local patterns of different entities over a set of geographically related regions vary a lot [4, 5, 34, 41]. Even the same entity could

Algorithm 2 GeoCNTN for location feature extraction

```
Input: Coordinate set $\{L_{ei}\}_{i=1}^{n_e}$ for each entity $e$
Output: Extracted features $F_e$
Initialize: Parameters $\theta$, regularization $\alpha$, learning rate $\beta$
 1: function EMBEDDING($L_e$)
 2:     $\{\tilde{I}_e^k\}_{k=1}^C, \{P_e^k\}_{k=1}^C$ ←DATA_REPRESENTATION($L_e$)   (Alg. 1)
 3:     Embed $\{\tilde{I}_e^k\}_{k=1}^C$ into vectors through embedding table
 4:     Do average pooling on obtained vectors to extract the global
        feature $GF_e$   (Sec. 4.1)
 5:     $LF_e$ ← CNN($\{P_e^k\}_{k=1}^C$)   (Sec. 4.2)
 6:     $F_e$ ← NTN($GF_e, LF_e$)   (Sec. 4.3, Eq. 7)
 7:     return $F_e$
 8: end function
 9: for all entity $e \in E$ do   (Sec. 4.4)
10:     $F_e$ ←EMBEDDING($L_e$)
11:     Update parameters with observations $X$   (Eq. 8,??)
```

have totally divergent local patterns within distinct areas. A real-world pattern is shown in Fig. 5.



(a) User may often stay around a fixed center. (b) User may prefer to visit fixed places. (c) User may often visit an avenue.

Figure 5: Various local patterns of users from Meetup-USA.

As far as we know, few previous works have successfully applied CNN model to geographical pattern extraction, since a rough preprocessing of geographical features may cause the otherwise powerful convolution operator suffer from the extreme sparsity of the global counter matrix for each entity. Thanks for our locally *zooming method* (Sec. 3.4) applied during the representation construction stage, we are able to alleviate the sparseness and obtain a CNN-friendly relatively dense representation of local patterns.

Technically, we choose a classical 3-layer CNN architecture for local feature extraction, as depicted in the upper part of Fig. 4. The input of our CNN architecture is the C matrices $\{P_s^k\}_{k=1}^C$, representing local patterns for each area $\tilde{A}_u^k$, as shown in Fig. 3(c). Such an input matrix is called a channel, which carries distinct views of information, as suggested by [14, 17, 19]. With a similar consideration, we stack C local patterns into a multi-channel input, actively embracing the diversity of local patterns in distinct areas. A multi-channel input is then forwarded into a convolutional layer, a pooling layer, and finally a projection layer for producing a local feature LF_e for entity e. The feature extraction power of CNN for local visiting patterns will be validated in the following sections via self-comparison experiments.

4.3 Combining Global and Local Features with Neural Tensor Network

After obtaining the global feature GF_e and local feature LF_e for a given entity e, we further our discussion on merging these two kinds of implicitly related feature into a unified form. Previous models such as [31, 35] adopt a naive method by concatenating these vectors into a longer one, which, from our perspective, may fail to model some deeper correlations between features residing on distinct layers of abstraction.

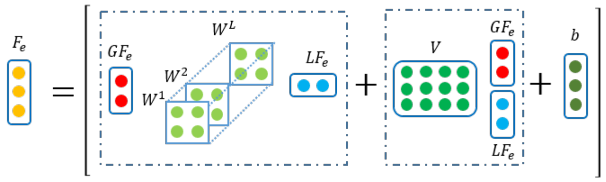


Figure 6: Neural tensor network for a better unifying of global and local features.

In fact, a better neural architecture for this subtask is neural tensor network (NTN) [37], which replaces the traditional structure of a standard linear layer with input on the concatenated vector with a tensor directly operating on several input features. This architecture has the power to explicitly model the interaction of features among the vector-valued cascading dimensions.

A formal description of NTN with our notations for global feature GF_e and local feature LF_e can be given as follows (a schematic diagram with $L = 3$ in Fig. 6)

$$F_e = \tanh\left(GF_e^T \cdot W^{1:L} \cdot LF_e + V \cdot \begin{bmatrix} GF_e \\ LF_e \end{bmatrix} + b\right) \tag{7}$$

where L represents the extra dimension of the tensor W, as compared to an original weight matrix, with V term maintaining a degenerate form as a linear layer and b the bias with dimension L. We will see the advantage of NTN over the original linear structure in our self-comparison experiments as well.

4.4 Optimization

The output of our GeoCNTN model can thus be considered as a unified geographical vector-valued feature F_e for each entity e. In order to refine the features obtained from our model, we introduce a generic form of loss function as follows

$$L(\theta) = L(X|F,\theta) + \alpha\|\theta\|^2 \tag{8}$$

where $L(X|F,\theta)$ is the log likelihood function of given set of features F, with parameters θ and observations X, and the latter term serves for regularization in case of overfitting. Specifically, $\theta = \{\theta_c, \theta_e, \theta_n\}$ are parameters of our model: θ_c for CNN, θ_e for global grid embedding structure and $\theta_n = \{W, V, b\}$ denotes the parameters of NTN.

Then we minimize the loss function in Eq. 8 in order to obtain an optimal θ^*. Specifically we use gradient based method for training, and We adopt Adam algorithm to conduct the learning process [15].

Our GeoCNTN is thus a widely applicable model used with distinct form of observations in a wide range of tasks. Typical real-world case studies in LBSN will be presented in Section 5 with corresponding choices for the generic loss in Eq. 8. As a summary of our model with the devised preprocessing method, the reader can refer to Algorithm 2.

5 APPLICATIONS AND EMPIRICAL RESULTS

In this section, we conduct two typical case studies to validate our location-based embedding model. The former uses heterogeneous

Table 1: Statistics of Datasets for Experiments.

Information \ Datasets	Meetup-USA	Meetup-Europe
Users	35255	21004
Groups	12860	2521
Heterogeneous links	174354	178414
Longitude range	$(-125°, -65°)$	$(-25°, 66°)$
Latitude range	$(23°, 55°)$	$(36°, 82°)$
Average location # per user	13.6	12.3
Average location # per group	26.7	20.3
Linkage density	7.691×10^{-4}	6.739×10^{-3}

social links and the latter experiment uses entities' implicit classification as observations. We compare our model with the state-of-art ones in both tasks. Finally we visualize the features learned by our models for each entity in LBSN, the result shows that our model not only perform well in prediction tasks but also give interpreted features, unifying both global and local geographical information after the supervised learning with observations.

We conduct our experiments on one NVIDIA GTX-1080. As a near-optimal setting of model's hyper-parameter during a large number of experiments, we choose the learning rate as 0.005, regularization parameter as 0.01, grid configuration for the map as 100×100, that for local pattern as 10×10, cluster count to 3 for our Geo-CMeans fuzzy clustering algorithm. In CNN, kernel size is 3×3, together with 2×2 max pooling and the projection layer as a fully connected layer.

5.1 Case Study: Link Prediction

We first use links between heterogeneous entities in a user-group LBSN as observations, specifically we set $E = \{U, G\}$, where U represents the set of users and G the set of groups. Then the objective loss function in Eq. 8 can be written as

$$L(X|F, \theta) = -\sum_{u \in U} \sum_{g \in G} \left[x_{ug} \log \tilde{x}_{ug} + (1 - x_{ug}) \log (1 - \tilde{x}_{ug}) \right] \quad (9)$$

where x_{ug} equals 1 if there is an observed link between u and g, 0 means not, with $\tilde{x}_{ug} = \sigma(F_u^T \cdot F_g)$ the prediction as a probability. Such a task is well defined since intuitively entities with more similar geographical features may have higher similarity in LBSN.

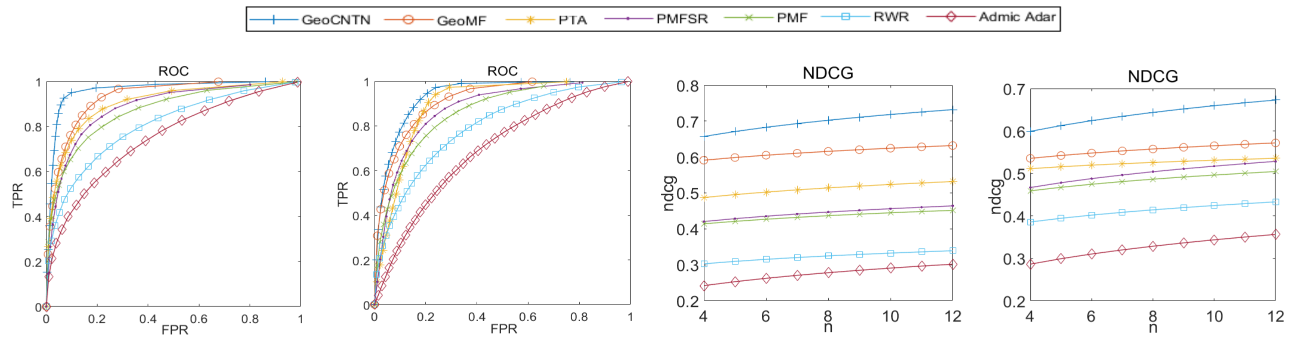
5.1.1 Experiment Settings. In this section we present our detailed experiment settings as follows,

Datasets Metrics We validate our model on two real-world Meetup datasets containing a set of locations for each entity, described by longitude-latitude pairs, together with observed links between heterogeneous entities. We choose two datasets, i.e. Meetup-USA and Meetup-Europe, with the detailed statistics shown in Table 1. We divide the training set and validation set with the ratio of 9:1. In terms of metrics, we adopt normalized discounted cumulative gain for first n items (NDCG@n) [12], receiver operating characteristic (ROC) curve [10] and its corresponding area under curve (AUC) [49] for the performance evaluation. Specifically, we choose NDCG@10 in evaluation.

Comparison Methods We compare GeoCNTN's performance with several state-of-art models in predicting heterogeneous links between users and groups, i.e. two content based methods (A)(B), two collaborative methods (C)(D) and two content-based collaborative methods (E)(F), together with three self-comparison models (G)-(I). A brief review is as follows, **(A) Adamic/Adar**: A metric for calculating similarity between two entities based on their common neighbors in graph [34]. Here we grid the map and count ratio of common grid as similarity. **(B) Random Walk with Restart (RWR)** [2]: It uses geographical similarity and social similarity between users and conduct predicting links between users by a Random Walk with Restart algorithm. **(C) Probabilistic Matrix Factorization (PMF)** [30]: A probabilistic model based on matrix factorization to predict the rating between heterogeneous entities. **(D) Probabilistic Matrix Factorization with Social Regularization (PMFSR)** [25]: It suggests that users have similar social connection should have similar latent factors, which outperform PMF in link prediction. Here, we assume that users in common groups share similar latent factors. **(E) Pairwise Tag-enhanced and feature-based Matrix factorization for group recommendation (PTA)** [47]: It adds similarity-based features between user and group to matrix factorization to conduct group recommendation. Here, we use the minimized distance, common grids of user and group as similarity. **(F) Geographical Matrix Factorization (GeoMF) [20]**: It uses kernel density estimator (KDE) to model user's distribution on global geographical grids and add this feature to matrix factorization to conduct point-of-interest (POI) recommendation for users. Here, we regard group's location set as POI's location. **(G) rawCNN**: Directly application of a CNN of the same architecture to the location counting matrix obtained from the global geographical grids and regard the output as the feature vector. **(H) GeoCNTN-no-local**: We remove the CNN component by injecting random white noise as local pattern **(I) GeoCNTN-no-tensor**: We replace the NTN component with a simply vector concatenation.

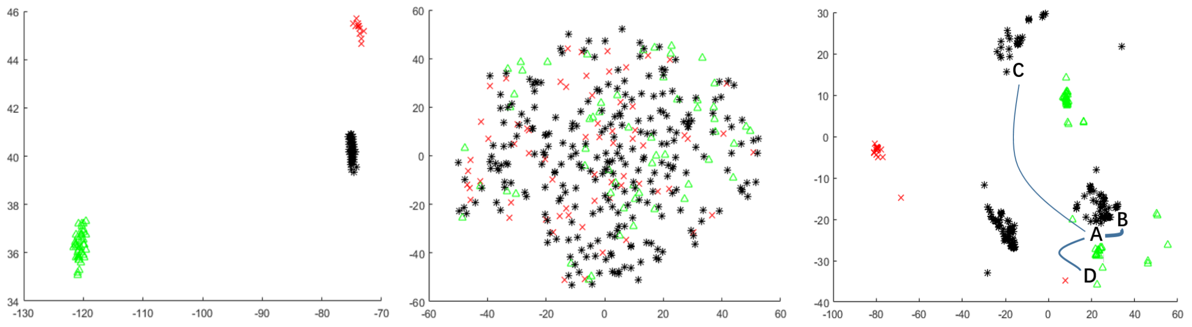
5.1.2 Performance. The NDCG@10 and AUC curves are shown in Table 2, Roc curves and NDCG@n are depicted as Fig. 7. Compared to the state-of-art models, our GeoCNTN brings a huge improvement by 9% in NDCG@10 for both dataset, compared with the best baseline GeoMF.

First comparing two content-based models Adamic/Adar and RWR, we can see that by the additional observations, RWR improved 4% in NDCG, which means it may not be satisfying if we only use location features to predict heterogeneous links in LBSN. For a collaborative method such as PMF and PMFSR, which only uses supervised heterogeneous links for learning and prediction, it performs better than RWR without using any geographical information. Then if we add rough preprocessed geographical information into MF as PTA, we can see that it achieves 7% improvement in NDCG than PMFSR, which states the fact that geographical information is useful in predicting social links in LBSN. By taking advantage of geographical information in a more sophisticated way, we can see that GeoMF made 10% improvement in NDCG than PTA. We infer that 2-dimensional geographical data carries important information in LBSN. AUC states the same result.



(b) Roc curves on Meetup-USA. (c) Roc curves on Meetup-Europe. (d) NDCG@10 on Meetup-USA. (e) NDCG@10 on Meetup-Europe.

Figure 7: The ROC curves and NDCG@10 results of different methods on the two datasets.



(a) Original locations on the map. (b) Features learned by PMFSR. (c) Features learned by our GeoCNTN.

Figure 8: Visualization of original locations (a) and features (b)(c) via t-SNE, where the color of each scatter point annotates the geographical cluster in (a) it comes from.

Table 2: AUC and NDCG@10 of comparison methods.

Results / Methods	Meetup-USA		Meetup-Europe	
	AUC	NDCG@10	AUC	NDCG@10
Adamic/Adar	0.747	0.291	0.691	0.344
RWR	0.805	0.333	0.793	0.386
PMF	0.871	0.445	0.859	0.496
PMFSR	0.885	0.456	0.875	0.517
PTA	0.906	0.524	0.896	0.531
GeoMF	0.933	0.625	0.915	0.565
rawCNN	0.497	0.011	0.496	0.021
GeoCNTN-no-local	0.864	0.435	0.854	0.433
GeoCNTN-no-tensor	0.951	0.664	0.923	0.630
GeoCNTN	**0.962**	**0.718**	**0.937**	**0.660**

Finally our model shows a remarkable advantage over these baselines. Unlike GeoMF, which neglects the local location distribution pattern inside global grids as a conprimise to the complexity of KDE, our model can exploit extra geographical information on a finer level. Especially, we made 9% improvement in NDCG@10, where as n gets larger, the result gets better, which means our model can indeed make less errors in ranking negative-prone links on top of the list. To do further study on the benefits brought by our delicate crafting of global and local information, we conduct several self-comparison models as follows.

5.1.3 GeoCNTN Parts Study. As depicted in Table 2, the comparison between rawCNN and GeoCNTN states the fact that CNN cannot even extract any information from the extreme sparse counting matrix obtained without our locally *zooming* strategy. Furthermore, if we remove the CNN part from our model, the result is similar to PMF because the embedding of global grids only contains latent information rather than geographical information. Finally, we study the effect of replacing the NTN with a naive concatenation, which brought poorer performance compared with our model by about 5% in NDCG. The result shows that the tensor layer indeed discovers the innate correlations between global information and local pattern.

5.2 Case study: Entity Classification

Then we study another task, i.e., use entities' location information to predict their classification. In LBSN, each entity always has a list of tags, e.g., shopping, reading, traveling etc. Entities with irrelevant tags may have different location pattern intuitively. Based on such an observation, we define the loss function as

$$L(X|F,\theta) = -\sum_{e\in E}\left[CEH(X_e,\tilde{X}_e)\right] \tag{10}$$

Table 3: Experiment results (F1 score) for entity classification (Case Study 2) on Meetup-USA.

Methods \ Entities	User	Group
Place Count	0.172	0.142
KDE	0.201	0.168
GeoCNTN	**0.317**	**0.267**

where CEH denotes the cross entropy error and $\tilde{X}_e = softmax(F_e)$ the predicted probability for each class.

5.2.1 *Experiment Settings.*

Here we target on two specific tasks, user classification and group classification in the Meetup-USA dataset. In this experiment, we treat the tags of the users/groups as their class labels. The set of users has 5372 tags in total and for groups, there are 5045 tags annotated on them. For each entity, the average tag number is 6. Considering the large number of tags, in order to reduce the class number, we adopt the method proposed by word2vec [27]. With this method, tags are at first embedded into vectors and then clustered into 20 clusters with K-Means based on their semantic distance. These 20 clusters finally serve as our target classes. Moreover, in terms of the baseline methods, since collaborative filtering methods such as PMF could not be directly applied to solve this problem, we adopt the **(A) place count** and **(B) kernel density estimator (KDE)** as two baselines, where the former one counts the locations in global grids for each entity and the latter applies KDE on these grids, similar to GeoMF [20]. We then train a multi-label support vector machine to predict the class the entity belongs to. To measure the performance, we use F1 score [11] as our metric.

5.2.2 *Performance.*

As shown in Table 3, we can see our model outperforms the best baseline 11% in F1 score. Comparing the baseline methods, we find that although they are both based on the counts on grids, for KDE has more continuous features, it makes a 16% improvement. However, their F1 scores are still low for their lack of information from local patterns, which may lead to the situation that these trained models would give a global class to each grid, e.g., shopping for New York City. GeoCNTN also uses the information from observed local patterns, which may give a much more accurate prediction. Hence it is not surprising that it outperforms the classical feature extraction methods.

Combined with the first case study, our GeoCNTN indeed has the flexibility and extensibility to be applied in a wide range of LBSN-related tasks. Via combining the global and local geographical information in a clever way, it should bring a noticeable improvement with a large probability.

5.3 Visualization of Extracted Features

As a further validation of our GeoCNTN's strength for unifying global and local geographical information with the supervision of observations in LBSN, we conduct a series of visualization experiments with the users' features we have obtained in the first case study. With the aid of t-SNE [26] (an auxiliary method that projects the high-dimensional data to a two-dimensional plane), we plot the the latent factors of users from the PMFSR and the unified features $\{F_e\}_{e \in E}$ learned by GeoCNTN respectively in 8(b) and 8(c), together with the users' original locations in 8(a).

As shown in Fig. 8, it is hard to explain pure latent features (Fig. 8(b)). However, our refined geographical feature is highly interpretable (Fig. 8(c)). The first observed property is that our model roughly preserves the geographical clusters. We say *roughly* because users (see the features in Fig. 8(a)) are divided into three clusters in (c). In order to find out the reasons, we sampled two users A and C, as plotted in Fig. 8(c). We found out that they have no common groups, in contrast to 9 common groups with user B, which means that although A and C are geographically closer, the lack of common social interactions makes their features more distant according to our model. What is more interesting, A and D are much closer w.r.t our features than A and B because they have 1 common group. We further study the relatedness between A and B in Fig. 8(c). When we inspected the local patterns from the preprocessed dataset (Fig. 5(a) & 5(b)), soon it became clear that they have totally different local patterns. Whereas the local patterns of A are relatively concentrated, those for B are scattered. This explains why we can infer that although they are close together in geography and share several existing social links, they are relatively separated in our feature diagram.

To sum up, our GeoCNTN has indeed extracted interpretable features by unifying both global and local geographical information, refined with observed information in LBSN.

6 CONCLUSION AND FUTURE WORK

In this paper, we proposed to study the problem of geographical feature extraction for entities in LBSN and described a novel location-based embedding model with neural network structure called GeoCNTN. It has three major contributions. First, we proposed a curvature-sensitive fuzzy clustering algorithm (Geo-CMeans) to obtain well-defined global and local representations from the original geographical data. We then proposed a neural network based learning model using convolutional neural network and an embedding structure for local pattern and global geographical feature extraction respectively. Finally, we applied a neural tensor network in order to discover higher-order correlations between global and local features. With a 9% improvement in NDCG for link prediction and a 11% improvement in F1 score for entity classification, we have indeed validated the embedding efficacy of our model. Furthermore, we validated the interpretability of the embeddings learned by our model with a series of visualization tasks.

Our work demonstrates how to use deep neural networks to extract geographical features for entities in LBSN. We propose several future directions of this work. First, it may be fruitful to extend the proposed location-based embedding structure towards unsupervised learning as shown in variational autoencoder (VAE) [16]. This can potentially alleviate the rare observation problem in a given LBSN. Another probable direction is to exploit and combine other geographical contexts, such as trajectories and temporal information of entities, with our current GeoCNTN for better embedding performance. Finally, it is interesting to connect the study with machine learning theories to further understand why GeoCNTN can perform better than baseline models in various LBSN-related tasks.

ACKNOWLEDGMENTS

The authors would also like to thank the anonymous referees for their valuable comments and helpful suggestions. This work is funded in part by the National Program on Key Basic Research (NO. 2015CB358800) and R&D Program of STCSM under Grant No.17JC1420200.

REFERENCES

[1] Daniel Ashbrook and Thad Starner. 2003. Using GPS to learn significant locations and predict movement across multiple users. *Personal and Ubiquitous computing* 7, 5 (2003), 275–286.

[2] Hakan Bagci and Pinar Karagoz. 2016. Context-aware friend recommendation for location based social networks using random walk. In *Proceedings of the 25th international conference companion on world wide web*. 531–536.

[3] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research* 3, Feb (2003), 1137–1155.

[4] Chen Cheng, Haiqin Yang, Irwin King, and Michael R Lyu. 2012. Fused Matrix Factorization with Geographical and Social Influence in Location-Based Social Networks.. In *Aaai*, Vol. 12. 17–23.

[5] Eunjoon Cho, Seth A Myers, and Jure Leskovec. 2011. Friendship and mobility: user movement in location-based social networks. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. 1082–1090.

[6] Justin Cranshaw, Eran Toch, Jason Hong, Aniket Kittur, and Norman Sadeh. 2010. Bridging the gap between physical location and online social networks. In *ACM International Conference on Ubiquitous Computing*. 119–128.

[7] Balázs Csanád Csáji. 2001. Approximation with artificial neural networks. *Faculty of Sciences, Etvs Lornd University, Hungary* 24 (2001), 48.

[8] J. C. Dunn. 1974. A fuzzy relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters. *Journal of Cybernetics* 3, 3 (1974), 32–57.

[9] Martin Ester, Hans Peter Kriegel, and Xiaowei Xu. 1996. A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. In *International Conference on Knowledge Discovery and Data Mining*. 226–231.

[10] Tom Fawcett. 2006. An introduction to ROC analysis. *Pattern Recognition Letters* 27, 8 (2006), 861 – 874. https://doi.org/10.1016/j.patrec.2005.10.010 ROC Analysis in Pattern Recognition.

[11] William B Frakes and Ricardo Baeza-Yates. 1992. Information retrieval: data structures and algorithms. (1992).

[12] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems (TOIS)* 20, 4 (2002), 422–446.

[13] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. 2014. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*. 675–678.

[14] Donghyun Kim, Chanyoung Park, Jinoh Oh, Sungyoung Lee, and Hwanjo Yu. 2016. Convolutional matrix factorization for document context-aware recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems*. 233–240.

[15] Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[16] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. 1097–1105.

[18] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521, 7553 (2015), 436–444.

[19] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.

[20] Defu Lian, Cong Zhao, Xing Xie, Guangzhong Sun, Enhong Chen, and Yong Rui. 2014. GeoMF: joint geographical modeling and matrix factorization for point-of-interest recommendation. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 831–840.

[21] Tao Lin, Tian Guo, and Karl Aberer. 2017. Hybrid Neural Networks for Learning the Trend in Time Series. (2017).

[22] Bin Liu, Hui Xiong, Spiros Papadimitriou, Yanjie Fu, and Zijun Yao. 2015. A general geographical probabilistic factor model for point of interest recommendation. *IEEE Transactions on Knowledge and Data Engineering* 27, 5 (2015), 1167–1179.

[23] Yong Liu, Wei Wei, Aixin Sun, and Chunyan Miao. [n. d.]. Exploiting geographical neighborhood characteristics for location recommendation. In *CIKM'14*. 739–748.

[24] Jonathan Long, Evan Shelhamer, and Trevor Darrell. 2015. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3431–3440.

[25] Hao Ma, Dengyong Zhou, Chao Liu, Michael R Lyu, and Irwin King. 2011. Recommender systems with social regularization. In *Proceedings of the fourth ACM international conference on Web search and data mining*. 287–296.

[26] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research* 9, Nov (2008), 2579–2605.

[27] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[28] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.

[29] Boriana L. Milenova and Marcos M. Campos. 2002. O-Cluster: scalable clustering of large high dimensional data sets. In *IEEE International Conference on Data Mining, 2002. ICDM 2003. Proceedings*. 290–297.

[30] Andriy Mnih and Ruslan R Salakhutdinov. 2008. Probabilistic matrix factorization. In *Advances in neural information processing systems*. 1257–1264.

[31] Yafeng Ren, Yue Zhang, Meishan Zhang, and Donghong Ji. 2016. Context-Sensitive Twitter Sentiment Classification Using Neural Network.. In *AAAI*. 215–221.

[32] Luca Rossi and Mirco Musolesi. 2014. It's the way you check-in: identifying users in location-based social networks. In *Proceedings of the second ACM conference on Online social networks*. ACM, 215–226.

[33] Maja Rudolph, Francisco Ruiz, Stephan Mandt, and David Blei. 2016. Exponential family embeddings. In *Advances in Neural Information Processing Systems*. 478–486.

[34] Salvatore Scellato, Anastasios Noulas, and Cecilia Mascolo. [n. d.]. Exploiting place features in link prediction on location-based social networks. In *SIGKDD'11*. 1046–1054.

[35] Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 373–382.

[36] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[37] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in neural information processing systems*. 926–934.

[38] Waldo R Tobler. 1970. A computer movie simulating urban growth in the Detroit region. *Economic geography* 46, sup1 (1970), 234–240.

[39] Cheng Yang, Maosong Sun, Wayne Xin Zhao, Zhiyuan Liu, and Edward Y Chang. 2017. A Neural Network Approach to Jointly Modeling Social Networks and Mobile Trajectories. *ACM Transactions on Information Systems (TOIS)* 35, 4 (2017), 36.

[40] Mao Ye, Peifeng Yin, Wang-Chien Lee, and Dik-Lun Lee. 2011. Exploiting geographical influence for collaborative point-of-interest recommendation. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*. 325–334.

[41] Josh Jia-Ching Ying, Eric Hsueh-Chan Lu, Wang-Chien Lee, Tz-Chiao Weng, and Vincent S Tseng. 2010. Mining user similarity from semantic trajectories. In *Proceedings of the 2nd ACM SIGSPATIAL International Workshop on Location Based Social Networks*. 19–26.

[42] Yonghong Yu and Xingguo Chen. 2015. A survey of point-of-interest recommendation in location-based social networks. In *Workshops at the Twenty-Ninth AAAI Conference on Artificial Intelligence*, Vol. 130.

[43] Quan Yuan, Gao Cong, Zongyang Ma, Aixin Sun, and Nadia Magnenat Thalmann. [n. d.]. Time-aware point-of-interest recommendation. In *NIPS'13*. 363–372.

[44] Jiawei Zhang and S Yu Philip. 2014. Link prediction across heterogeneous social networks: A survey. *SOCIAL NETWORKS* (2014).

[45] Jia-Dong Zhang and Chi-Yin Chow. 2015. CoRe: Exploiting the personalized influence of two-dimensional geographic coordinates for location recommendations. *Information Sciences* 293 (2015), 163–181.

[46] Jia-Dong Zhang, Chi-Yin Chow, and Yanhua Li. 2015. iGeoRec: A personalized and efficient geographical location recommendation framework. *IEEE Transactions on Services Computing* 8, 5 (2015), 701–714.

[47] Wei Zhang, Jianyong Wang, and Wei Feng. 2013. Combining latent factor model with location features for event-based group recommendation. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. 910–918.

[48] Yang Zhang and Jun Pang. 2015. Distance and Friendship: A Distance-Based Model for Link Prediction in Social Networks. (2015).

[49] Kelly H Zou, A James OâĂŹMalley, and Laura Mauri. 2007. Receiver-operating characteristic analysis for evaluating diagnostic tests and predictive models. *Circulation* 115, 5 (2007), 654–657.