HisRect

Features from Historical Visits and Recent Tweet for Co-Location Judgement

Li, Pengfei; Lu, Hua; Zheng, Qian; Li, Shijian; Pan, Gang

Published in:
I E E E Transactions on Knowledge & Data Engineering

DOI (link to publication from Publisher):
10.1109/TKDE.2019.2934686

Publication date:
2021

Document Version
Accepted author manuscript, peer reviewed version

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2934686, IEEE Transactions on Knowledge and Data Engineering

HisRect: Features from Historical Visits and Recent Tweet for Co-Location Judgement

Pengfei Li, Hua Lu, *Senior Member, IEEE,* Qian Zheng, Shijian Li, and Gang Pan

Abstract—Enabled by smartphones, social media users are increasingly going mobile. This trend fosters various location based services on social media platforms (*e.g.*, Twitter). Many services like friends notification and community detection benefit from co-location judgement, *i.e.*, to decide whether two Twitter users are co-located in some point-of-interest (POI). This problem is challenging due to the limited information in tweets and the lack of explicit geo-tags in tweets that can be used as labeled data. Our approach to this problem is based on a novel concept of *HisRect* features extracted from users' historical visits and recent tweets: The former has impacts on where a user visits in general, whereas the latter gives more hints about where a user is currently. In practice, labeled data is scarce. Therefore, we design a semi-supervised learning (SSL) framework that leverages unlabeled data to extract HisRect features. Moreover, we employ an embedding neural network layer to process HisRect features of two users, which decides co-location based on the embedding difference between the two features. Our model is extensively evaluated on two large sets of real Twitter data from more than one million users. The experimental results demonstrate that our HisRect features and SSL framework are highly effective at deciding co-locations. In terms of multiple metrics, our approach clearly outperforms alternative approaches using state-of-the-art techniques.

Index Terms—Twitter, POI, Co-Location Judgement, Semi-Supervised Learning

1 Introduction

Driven by smartphones and mobile Internet, social media users are increasingly going mobile [1], [2]. For example, Twitter had approximately 257 million mobile active users monthly as per the first quarter in 2016[1]. Along with this trend is the emergence of location based services deployed on social media. Most of such location based services require accurate or coarse user locations to decide the service results for users. However, many people actually do not share their precise locations in their social posts [3], [4]. It is thus necessary to bridge this gap in order for the relevant location based services to take full effect.

In this work, we study a co-location judgement problem, *i.e.*, to judge if two Twitter users are at the same point-of-interest (POI) in a period of time. We stipulate that two tweets are sent in the same time period if their time difference is less than a threshold Δt. Many important applications today need to know how to acquire the information about "if two users stay together" or "who are in the same place among a group of users". For example, friends notification [5] is a very popular service on social media, which notifies a user that one of his/her friends is also present at the same POI in the same time. As another example, many social network platforms also offer local people recommendation [6], [7], which can recommend users who are close to and share the same interest with a user in need. Furthermore, "followship" measurement in the real world [8] investigates when a person visits a POI due to the influence of another person. Other examples include community detection and group analysis [9], [10], [11] that aim to find users sharing interests and appear in the same place at the same time. Such people may form communities in an online-to-offline fashion to fulfill different purposes. All aforementioned applications can clearly benefit from co-location judgement. Existing works usually requires the input data to be geotagged, *i.e.*, tweets must be associated with coordinates or place names. However, geo-tagged tweets only occupy a small fraction in all tweets [3]. Solving the problem of co-location judgement provides a better way to enable these applications as it can deal with non-geotagged tweets. Traditional location inference approaches [12] also work in these tasks—we just infer the location of every user and then judge if users in question are in the same location. However, as our experiments show, such a method results in low inference accuracy whereas our co-location judgement approach achieves high judgement accuracy.

The problem of co-location judgement on Twitter data is challenging. On the one hand, the content length of each tweet is limited to 140 characters and thus a tweet conveys little information in general. As Twitter users often use non-standard and shorthand terms, tweets are often vague and noisy. Thus, it is difficult to find location clues from short, noisy tweets. On the other hand, as tweets are mostly not geo-tagged [13], and even fewer tweets are explicitly associated to POIs, the labeled data across each POI, on average, is scarce.

Since only a small fraction of tweets are geo-tagged, it is not possible to directly know if two Twitter users are co-located or not. We have to make use of other information available for the users. Fortunately, people often send location-related tweets when they get to new POIs. Also, the places a user has ever visited tend

- *P. Li is with Department of Computer Science, Zhejiang University, China. E-mail: pfl@zju.edu.cn*
- *H. Lu is with the Department of Computer Science, Aalborg University, Denmark. E-mail: luhua@cs.aau.dk*
- *Q. Zheng is with Department of Computer Science, Zhejiang University, China. E-mail: csqianzheng@gmail.com*
- *S. Li is with Department of Computer Science, Zhejiang University, China. E-mail: shijianli@zju.edu.cn*
- *G. Pan is with the State Key Lab of CAD&CG and Department of Computer Science, Zhejiang University, Hangzhou 310027, China. E-mail: gpan@zju.edu.cn.*
- *(Corresponding authors: Shijian Li and Hua Lu)*

1. http://expandedramblings.com/index.php/twitter-mobile-statistics/

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2934686, IEEE Transactions on Knowledge and Data Engineering

to have an impact on where they are now or where they are going. Therefore, our approach to the co-location judgement problem is based on historical visits and recent tweet contents.

To address the challenges, our solution integrates users' visit history and recent tweets in feature selection. A user's historical visits can be regarded as a kind of prior information, whereas her/his recent tweet is often related to where she/he is (or heading to). We deliberately extract features from historical visits and recent tweets (HisRect for short), which combine Twitter user visit history and recent tweet contents. Furthermore, we decide if two users are co-located based on the difference between their HisRect features.

To alleviate the issue of data scarcity, we further propose a semi-supervised learning (SSL) framework [14] that leverages unlabeled data, those geo-tagged tweets which are not explicitly associated to any POI, to train the HisRect featurizer. Subsequently, we feed the HisRect features of two users to another embedding neural network layer and calculate the difference of the two embeddings. Finally, we construct our co-location judger as a feed-forward neural network that only takes the embedding difference as input.

Our HisRect featurizer and co-location judgement are experimentally evaluated on large real datasets. The experimental results demonstrate that our HisRect based approach is effective at finding co-located users and it clearly outperforms alternative approaches. Furthermore, our HisRect features result in more accurate POI inference than the state-of-the-art techniques.

Our contributions are summarized as follows.

- We formulate the problem of co-location judgement on Twitter data. To the best of our knowledge, this is the first work to address this problem.
- We design HisRect features which quantify the spatial and temporal aspects in Twitter users' visit history and address the local features in recent tweets.
- We develop a novel semi-supervised embedding learning framework to train the HisRect featurizer such that unlabeled data can be exploited in our solution. The framework includes a carefully-designed affinity graph that considers both spatial and temporal distances between HisRect feature instances. Relevant experimental results demonstrate the effectiveness of the proposed semi-supervised method.
- We conduct extensive performance evaluation using real-world Twitter data. The results verify the effectiveness of our HisRect features and semi-supervised framework.

The rest of this paper is organized as follows. Section 2 reviews the related work. Section 3 formulates the research problem. Section 4 describes how to build HisRect features and presents the semi-supervised learning framework for training the featurizer. Section 5 details our approach for co-location judgement. Section 6 reports the experimental results. Finally, Section 7 concludes the paper and points to future work directions.

2 Related Work

To the best of our knowledge, there exists no work on the co-location judgement problem on Twitter data. A straightforward solution works as follow. For two twitter users who sent tweets in the same period of time, we infer two respective locations based on their tweet contents or their historical visits and check if the two inferred locations are the same. Therefore, we briefly review location inference or recommendation methods in Section 2.1. Also, we review semi-supervised learning techniques briefly in Section 2.2.

2.1 Location Inference or Recommendation

To infer locations for individual tweets, most existing approaches rely on tweet contents. Kinsella *et al.* [15] create language models of locations using geo-tagged tweets, measure the Kullback-Leibler divergence between such a model and a tweet, and infer the tweet location at the neighborhood and city level. Doran *et al.* [16] build smoothed language models to estimate tweet locations at the neighborhood level. Priedhorsky *et al.* [17] propose a two-dimensional Gaussian Mixture Model to infer the city of a tweet. Zubiaga *et al.* [13] extract features from user profiles and tweet contents, following a weighted maximum entropy classifier to determine the country for a tweet in real-time. Flatou *et al.* [18] use a Gaussian model to capture the location distributions of n-grams and associate geographic scope to such n-grams.

Some studies consider not only content and metadata in user profiles but also temporal information. Yuan *et al.* [19] propose a probabilistic topic model to exploit micro-blogging data to detect spatio-temporal topics, and then they use the topics to model user mobility behavior and infer tweet locations. Dredze *et al.* [20] also consider the impact of time on tweet locations, taking time as a feature and using a linear classifier trained on geo-tagged tweets to infer tweet locations at the city level. Palpanas and Paraskevopoulos [21], [22] exploit the similarities in the contents between a tweet and a set of geo-tagged tweets posted at the same time in order to decide if the given tweet is from the same location as others. Besides, Noulas *et al.* [23] and Ryoo *et al.* [24] exploit users mobility to predict their next places. McGee *et al.* [25], [26], Yamaguchi *et al.* [27] and Kong *et al.* [28] consider temporal spatial aspects when making location inference.

Some recent POI/location recommendation models are able to predict the location or POI for a given user at a given time period. Recent studies [29], [30], [31], [32], [33] focuses on leveraging the geographical and social influences to improve recommendation accuracy. Studies [34], [35], [36] make use of temporal cyclic patterns and temporal sequential patterns. Besides, semantic information is adopted by many recommendation approaches to alleviate the data sparsity problem. Yin *et al.* propose LCA-LDA [37] and Geo-SAGE [38] models to exploit the content information of checked-in POIs to infer both personal interests and local preferences. Also, SPORE [39] fuses sequential influence with personal interests in the latent and exponential space, TPM [40] utilizes cyclic patterns, and GE [41] embeds four corresponding relational graphs into a shared low dimensional space to capture the sequential effect, geographical influence, temporal cyclic effect and semantic effect. ST-LDA [42] learns region-dependent personal interests according to the contents of their checked-in POIs at each region. SH-CDL [43] jointly performs deep representation learning for POIs from heterogeneous features and hierarchically additive representation learning for spatial-aware personal preferences. However, unlike our HisRect-based approach, these models requires a user's historical data in the training process in order to make recommendations for the user. That said, these models can hardly deal with new users, *i.e.*, those users whose historical data is not available in the training dataset. Furthermore, they require semantic information about POIs (such as tags or descriptions). Moreover, most of these models focus on building static features for users and POIs/items. Thus, they will predict the locations of a users at two time periods to be the same. Although some models consider the temporal information when making predictions, they only concern time, days or hours, *e.g.*, if a user posts two tweets of 2 o'clock but on two different Saturdays, the two times are regarded as the same and the user will most likely receive the same POI recommendations. Also, they do not utilize the textual information of relevant tweets when predicting the locations of users. In summary, those recommendation methods fall short for our co-location judgment problem as their design characteristics are different from what is needed by our problem..

2.2 Graph-based Semi-Supervised Learning

Semi-supervised learning (SSL) aims to leverage unlabeled data to improve performance when labeled data is scarce. Graph-based SSL uses a matrix to describe the similarities between any two instances. Let L and U be the numbers of labeled and unlabeled instances, respectively. Let $\mathbf{x}_{1:L}$ and $\mathbf{x}_{L+1:L+U}$ denote the input vectors of labeled and unlabeled instances, respectively, and $\mathrm{y}_{1:L}$ are the labels of $\mathbf{x}_{1:L}$. Graph-based SSL learns a classifier $f : \mathbf{x} \rightarrow \mathrm{y}$. The mainstream approaches usually need an affinity graph, which is a $(L + U) \times (L + U)$ matrix $\boldsymbol{A}$. Each entry a_{ij} indicates the similarity between instances i and j that are either labeled or unlabeled. The matrix can be derived from distances between instances [14], [44], [45], [46], or from external data such as knowledge graphs [47], document citation networks [48] and social networks [49]. In this paper, our matrix $\boldsymbol{A}$ is constructed based on the spatial and temporal distances between instances.

Generally, the loss function of graph-based SSL can be written as follows [14]:

$$\mathcal{L} = \sum_{i=1}^{l} \ell_l(f(\mathbf{x}_i), \mathrm{y}_i) + \sum_{i,j=1}^{l+u} \ell_u(f(\mathbf{x}_i), f(\mathbf{x}_j), a_{ij})$$

In particular, $f(\cdot)$ is the prediction function to be learned that maps input x to labels y, ℓ_l is some proper loss function for supervised loss on labeled data, and and ℓ_u for unsupervised loss on both labeled and unlabeled data. Specifically, ℓ_u incurs a large penalty when similar instances with a large a_{ij} are predicted to have different labels.

Different graph-based SSL algorithms define unsupervised loss ℓ_u in different ways. Zhu *et al.* [45] use label propagation to force f to agree with $y_{1:L}$, where f is a label lookup table for unlabeled instances in the graph and can be obtained with a closed-form solution. Talukdar *et al.* [49] propose a variant of label propagation called modified adsorption that allows prediction on labeled instances to vary and incorporates node uncertainty. Zhou *et al.* [44] define ℓ_u as squared loss. Belkin *et al.* [14] use the Laplacian Eigenmaps regularizer and parameterize f in the Reproducing Kernel Hilbert Space with ℓ_u being squared loss or hinge loss.

Different from aforementioned approaches, Yang *et al.* [50] present an SSL framework based on graph embeddings. Weston *et al.* [46] propose to learn an embedding function instead of a prediction function. In their proposal, $f(\mathbf{x}) \in \mathbb{R}^d$ is the embedding for a given instance $\mathbf{x} \in \mathbb{R}^n$. In this case, ℓ_u is defined as $a_{ij}\|f(\mathbf{x_i}) - f(\mathbf{x_j})\|_2$ for any instances $\mathbf{x_i}$ and $\mathbf{x_j}$.

In this paper, we adopt the idea of semi-supervised embedding. Nevertheless, we use a neural network to fit the embedding function f because neural networks have powerful expression ability [51]. As the dimension of the embeddings is high (up to 512) in our setting, the Euclidean distance is not a feasible measure for similarity [52]. Instead, we use the cosine distance to calculate ℓ_u.

3 Problem Formulation and Framework

This section formulates the research problem and gives our solution framework. .

3.1 Notations and Problem Formulation

It is very difficult to directly know if two Twitter users are co-located as the geo-tagged tweets are scarce. Thus, we consider utilizing other information available to solve the problem. According to our observations, the places a user has ever visited tend to have impact on where they are now or where they are going. Also, people usually send location-related tweets contents when they get to new POIs. In this study, we make use of geo-tagged tweets that are posted with geo-locations captured as latitude and longitude. Such a tweet implies a visit of a place by the corresponding Twitter user. With the help of an appropriate geographic information service like OpenStreetMap[2], we are able to decide if a geo-tagged tweet was posted in a POI. In addition, we also consider the contents of the most recent tweets of Twitter users. By taking into account the visit histories and the recent tweet contents, we expect to identify if two Twitter users are currently located in the same POI or not.

The notations used throughout the paper are given in Table 1.

TABLE 1: Notations

P	Set of POIs
R_L	Labeled profiles
R_U	Unlabeled profiles
R_L^{train}	Labeled profiles of training dataset
R_L^{test}	Labeled profiles of testing dataset
Γ_L	Labeled pairs
Γ_L^+, Γ_L^-	Positive and negative pairs with $\Gamma_L^+ \bigcup \Gamma_L^-$=$\Gamma_L$
Γ_L^{train}	Labeled pairs of training dataset
Γ_L^{test}	Labeled pairs of testing dataset
Γ_U	Unlabeled pairs
Γ_U^{train}	Unlabeled pairs of training dataset
Δt	The time period

Definition 1 (POI). A POI p is a 4-tuple $p = (pid, bp, lat, lon)$, where pid is p's identifier, bp is p's bounding polygon formed by connecting N coordinate points, and (lat, lon) is the central point of the polygon bp.

We use $(lat, lon) \in p.bp$ to denote that a point (lat, lon) is inside the POI p.

Definition 2 (Tweet). A tweet t is a 4-tuple t = (ts, $content$, lat, lon) where ts is the timestamp when t was posted, and $content$ is the content of t. If t is a geo-tagged tweet, lat and lon represents the latitude and longitude, respectively. Otherwise, both lat and lon are set to null.

Let P be the set of all POIs. A tweet t is a *POI tweet* if there is a POI $p \in P$ such that $(t.lat, t.lon) \in p.bp$, *i.e.*, tweet t was posted when the user was at a POI in P.

A geo-tagged tweet implies a visit as follows.

Definition 3 (Visit). A visit v is a 3-tuple v = (ts, lat, lon), meaning a user visited location (lat, lon) at time ts.

2. https://www.openstreetmap.org/

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2934686, IEEE Transactions on Knowledge and Data Engineering

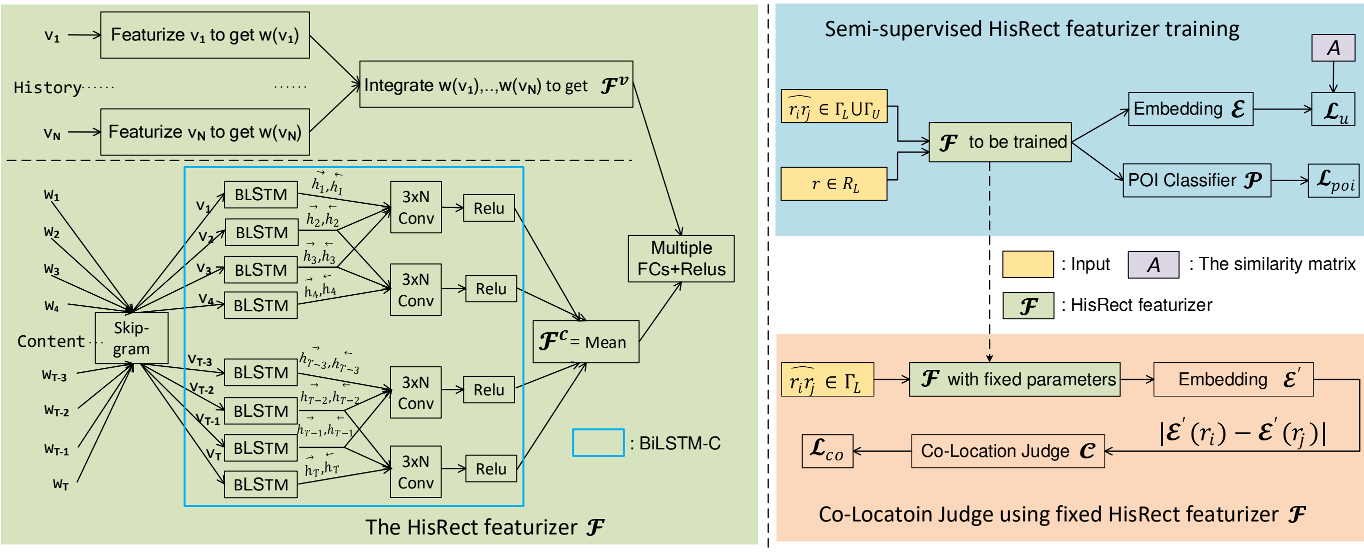


Fig. 1: The framework of building and training HisRect features and co-location judgement with HisRect featurizer

In the training phase, we extract ts, lat and lon from all geo-tagged tweets in a user's timeline, which forms a complete sequence of visits of the user. Such a sequence is called the user's *visit history*. Furthermore, we build user profiles that combine user visit history and recent tweet.

Definition 4 (Profile). A user profile r is a 4-tuple $r = (uid, t, v\text{-}history, pid)$, where uid identifies the user who sent the recent tweet t, $v\text{-}history$ is the user's visit history before $t.ts$, and pid is a POI identifier.

For convenience, we use $r.ts$, $r.lat$, $r.lon$ and $r.content$ to denote $r.t.ts$, $r.t.lat$, $r.t.lon$ and $r.t.content$, respectively. If $r.t$ is a POI tweet at p, $r.pid$ is set to $p.pid$ and r is regarded as *labeled*; otherwise, $r.pid$ is set to null and r is *unlabeled*.

In the co-location judgement, we consider a pair of users.

Definition 5 (Pair). A pair is a 3-tuple $\widehat{r_ir_j} = (r_i, r_j, co\text{-}label)$ where r_i and r_j are two profiles, $r_i.uid \neq r_j.uid$, and $|r_i.ts - r_j.ts| < \Delta t$.

A pair is *unlabeled* and its $co\text{-}label$ is set to null, if either of its profiles is unlabeled. Otherwise, the pair is *labeled*. In a labeled pair, $co\text{-}label$ is set to 1 in the training data if the two users are co-located at the same POI. Such a pair is *positive*. Otherwise, $co\text{-}label$ is set to 0 and the pair is *negative*. We use Γ_L^+ and Γ_L^- to denote the sets of positive and negative pairs, respectively.

In the rest of this paper, we use $d(a, b)$ to denote the spatial distance between two objects with location connotations. Specifically, a or b can be a profile, a visit or a POI. Besides, we define the spatial distance $d(r, P)$ between a profile r and the POI set P as the lower bound distance between r and all POIs in P, *i.e.*, $d(r, P) = \min\{d(r, p) \mid p \in P\}$.

Our research problem is formulated as follows:

Co-Location Judgement: Given two **profiles** r_i and r_j generated by two new users such that $r_i.uid \neq r_j.uid$, and $|r_i.ts - r_j.ts| < \Delta t$, *i.e.*, r_i and r_j form a **pair**, suppose that the two profiles' $pids$ are unknown, decide if they are co-located, *i.e.*, if they are from the same POI.

3.2 Solution Framework

Fig. 1 shows the framework of our solution to the co-location judgement problem. The core is to extract features from historical visits and recent tweets in profiles. The left part illustrates how it works. The feature, called 'HisRect', $\mathcal{F}(r)$ of the profile r includes two parts: the fixed dimensional feature $\mathcal{F}^v(r)$ which quantifies the spatial and temporal aspects of historical visits, and $\mathcal{F}^c(r)$ which extracts the location clues from the raw recent tweet contents and is also fixed dimensional. To get $\mathcal{F}^c(r)$, we adopt the skip-gram algorithm [53] to build word vectors first. Then BiLSTM-C [54], an LSTM variant, is used to convert the word vector sequence of the tweet content in a profile into the fixed dimensional feature $\mathcal{F}^c(r)$. We detail the procedure of building HisRect feature of profiles from Section 4.1 to 4.3. To train this featurizer, we propose an graph-based semi-supervised learning approach, as shown in right upper part in Fig. 1. The SSL framework takes pairs and labeled profiles as input. It needs a similarity matrix A that measures the 'distance' of both profiles in a pair. To create A, we take the spatial and temporal distance of two profiles into consideration (Section 4.4). After that, each profile is expressed as a fixed dimensional feature vector. Finally, the right lower part shows the procedure of judging if a pair $\widehat{r_ir_j}$ is co-located or not using the featurizer $\mathcal{F}$. We calculate the difference of features $\mathcal{F}(r_i)$ and $\mathcal{F}(r_j)$ and feed it into a binary classifier composed of fully-connected layers and activation functions only. It is noteworthy that the parameters of $\mathcal{F}$ are fixed during this stage. Section 5 describes the approach in details.

4 HisRect Feature

The HisRect $\mathcal{F}(r)$ of a profile r includes two parts: $\mathcal{F}^v(r)$ featurizes $r.v\text{-}history$; $\mathcal{F}^c(r)$ is the embedding of $r.content$. The left part of Fig. 1 shows how to build HisRect feature of a profile.

4.1 Feature of Historical Visits

A straightforward way to featurize a user's visit history is to use a one-hot encoding vector representing the POI identities [33]. However, this way fails to take into account when the visits to those POIs took place and neither can this way utilize those visits whose coordinates are not inside any POI. Thus, we need to find a better way to model users' visit histories.

In a short period of time, a user is unlikely to move from one POI to another that is far away. Therefore, a user's historical visits have an impact, to some extent, on her/his current locations. Intuitively, more recent visits tend to have a higher impact. It is thus beneficial to utilize a user's visit history by considering both temporal and spatial aspects. More specifically, we want to know where the users have been before, when they went to those places, how far the two places are with subsequent POI candidates. To quantify these aspects, we propose a featurizing method, shown in the left upper part of Fig. 1, that considers all of these aspects:

$$\boldsymbol{w}(v) = [\frac{\epsilon_d}{\epsilon_d + d(v, p_1)}, ..., \frac{\epsilon_d}{\epsilon_d + d(v, p_{|P|})}] \tag{1}$$

$$\boldsymbol{\mathcal{F}}^v(r) = \ell^2\text{-}norm\big(\sum_{v \in r.v\text{-}history} \frac{\epsilon_t}{\epsilon_t + r.ts - v.ts} \boldsymbol{w}(v)\big) \tag{2}$$

Above, $d(v, p_i)$ is the spatial distance between the visit v and the i-th POI p_i. The i-th item in $\boldsymbol{w}(v)$ is a spatial relevance measure between v and p_i: the value of ith item in $\boldsymbol{w}(v)$ almost grows in inverse proportion with the distance between POIs of v and p_i. The coefficient $\frac{\epsilon_t}{\epsilon_t + r.ts - v.ts}$ is an influential factor of the timestamp of v, which implies that more recent visits have a larger impact on the user's current POI. Parameters ϵ_d and ϵ_t are distance-concerned and time-concerned smoothing factors, respectively. It is evident from the definition of $\boldsymbol{\mathcal{F}}^v(r)$ that if the user have ever been to some places close to some POI p_i, the ith weight in $\boldsymbol{\mathcal{F}}^v(r)$ will be large.

On the one hand, by using $\boldsymbol{\mathcal{F}}^v(r)$ to quantitatively describe the temporal and spatial distances between users' visits and every POI, we obtain a kind of prior information about a user's current location. On the other hand, $\boldsymbol{\mathcal{F}}^c(r)$ extracts posterior information from users' recent tweets that contain more clues about where they are now.

If a profile r contains no historical visit, we just set the value of every dimension of $\boldsymbol{\mathcal{F}}^v(r)$ to be the same, i.e., $\boldsymbol{\mathcal{F}}^v(r) = \ell^2\text{-}norm\big([1, .., 1]\big)$. Therefore, our model is able to deal with the user timelines that contain no POI tweet.

We compare our approach with that using one-hot encoding and find our approach performs much better. These experimental results are reported in Section 6.

4.2 Feature of Recent Tweet

Generally, $r.content$ can be defined as a sequence of words $r.content$ = $(w_1, ..., w_T)$ with length of T. Since processing words directly is difficult, we convert $r.content$ into a sequence of fixed-dimensional vectors. In particular, we extract the content of all tweets of each timeline in our training data C_{train} and use the skip-gram algorithm [53] to train these word vectors. Consequently, each word is expressed as an M-dimensional vector of float numbers, where M is an empirical value that has little impact on the overall model performance. It is set to 512 in our experiments. Subsequently, we express the word sequence of $r.content$ as a word vector sequence $X = (x_1, ..., x_T)$. This X serves as the vectorization for $r.content$. Each x_t in X is the word w_t's M-dimensional vector and the length of the sequence X, i.e., $|X|$, is T. This way, we convert $r.content$ into X.

Bidirectional Long short-term memory (BLSTM) [55] is specialized for sequential data. It takes X as input and computes the N-dimensional hidden state sequences $\overrightarrow{\mathcal{H}}$ = $(\overrightarrow{h_0}, ..., \overrightarrow{h_T})$ and $\overleftarrow{\mathcal{H}}$ = $(\overleftarrow{h_0}, ..., \overleftarrow{h_T})$ bidirectionally. However, BLSTM take the sequence of individual word vectors as input. Sometimes, an individual word cannot give clear location clues but word groups or phrases have close ties with some particular locations. For example, "statue" or "liberty" can be used everywhere, whereas "Statue of Liberty" is the landmark of New York City. Motivated as such, we combine word groups in the hope of extracting such local features that are more powerful for location inference.

To address this idea, we use $BiLSTM\text{-}C$ [54], a variant of LSTM which adds a convolution layer above the BLSTM layer, to exploit the local features inside the word groups. It concatenates every vector in $\overrightarrow{\mathcal{H}}$ and $\overleftarrow{\mathcal{H}}$ and converts the combination of them into a $T \times N \times 2$-dimensional tensor $\boldsymbol{H}$ that can be viewed as a 2-channel image with height T and width N. Using one filter $\boldsymbol{K} \in \mathbb{R}^{3 \times N}$ to convolute $\boldsymbol{H}$, followed by a nonlinear rectified linear unit (Relu) operation, $BiLSTM\text{-}C$ gets a $(T\text{-}2) \times N$-dimensional output "feature map". By computing the mean of elements in this "feature map" across the first dimension, we get the fixed N-dimensional feature $\boldsymbol{\mathcal{F}}^{\boldsymbol{c}}(r)$ of $r.content$:

$$\boldsymbol{H}_{::0} = \begin{pmatrix} \overrightarrow{h}_0 \\ \overrightarrow{h}_1 \\ \vdots \\ \overrightarrow{h}_T \end{pmatrix} \qquad \boldsymbol{H}_{::1} = \begin{pmatrix} \overleftarrow{h}_0 \\ \overleftarrow{h}_1 \\ \vdots \\ \overleftarrow{h}_T \end{pmatrix}$$

$$\boldsymbol{\mathcal{F}}^{\boldsymbol{c}}(r) = \boldsymbol{Mean}\big(\boldsymbol{Relu}\left(\boldsymbol{K} * \boldsymbol{H}\right)\big) \tag{3}$$

The architecture of $BiLSTM\text{-}C$ is shown in the bottom-left part of Fig. 1.

4.3 Combination of $\boldsymbol{\mathcal{F}}^v(r)$ and $\boldsymbol{\mathcal{F}}^c(r)$

To obtain the final HisRect $\boldsymbol{\mathcal{F}}(r)$, we merge $\boldsymbol{\mathcal{F}}^v(r)$ and $\boldsymbol{\mathcal{F}}^c(r)$ through vector concatenation and feed $[\boldsymbol{\mathcal{F}}^v(r), \boldsymbol{\mathcal{F}}^c(r)]$ to a feed-forward neural network which is composed of some stacked fully connected layers followed by nonlinear rectified linear units ($Relu(x) = max(0, x)$). Combining Eq. (2) and (3), we obtain the representation of HisRect feature $\boldsymbol{F}(r)$:

$$\boldsymbol{\mathcal{F}}(r) = h^{Q_f}\Big(...h^2\big(h^1([\boldsymbol{\mathcal{F}}^v(r), \boldsymbol{\mathcal{F}}^c(r)])\big)\Big),$$

where Q_f is the total number of fully connected layers in $\boldsymbol{\mathcal{F}}$.

4.4 Semi-Supervised HisRect Training

The HisRect feature is the bridge between raw profiles and POIs. It is natural to train $\boldsymbol{\mathcal{F}}$ by feeding HisRect features to a POI inference classifier. As we have a labeled profile set R_L, a straightforward way is to use a supervised learning method. Such a method estimates the probabilities of a profile located in every POI and builds a supervised loss function based on the probabilities and their actual labels. However, a supervised method falls short when there lacks sufficient training data. In our case, out of the total 1,904,227 profiles, only 533,400 are associated to a POI. In other words, the amount of unlabeled data is almost three time larger than that of the labeled data. This motivates us to employ a graph-based semi-supervised learning (SSL) method to leverage unlabeled data to generate better features.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2934686, IEEE Transactions on Knowledge and Data Engineering

Our SSL approach aims to solve the following optimization problem:

$$f^* = \arg\min_{f \in \mathscr{N}} \mathcal{L}_{poi} + \mathcal{L}_u \tag{4}$$

$$\text{where } \mathcal{L}_u = \sum_{i,j=1}^{l+u} a_{ij} \Big(1 - \langle \boldsymbol{\mathcal{E}}(\boldsymbol{\mathcal{F}}(r_i)), \boldsymbol{\mathcal{E}}(\boldsymbol{\mathcal{F}}(r_j)) \rangle \Big)$$

$$\text{with } \boldsymbol{\mathcal{E}}(\mathbf{x}) = normalize\Big(h^{Q_e}\Big(...h^2\big(h^1(\mathbf{x})\big)\Big)\Big)$$

Above, f is the objective function, $\mathscr{N}$ is a normalized vector function space, and $\mathcal{L}_{poi}$ is the supervised loss of the POI classifier $\boldsymbol{\mathcal{P}}$, a feed-forward neural network taking the labeled profiles set R_L as input which is just the cross entropy in our settings. $\boldsymbol{\mathcal{E}}$ is a normalized feed-forward network to embed the HisRect features of profiles into $\mathbb{R}^E$ space in which E is the dimensionality of embeddings and Q_e is the number of fully connected layers. The unsupervised loss $\mathcal{L}_u$ takes the labeled and unlabeled pairs sets Γ_L and Γ_U as input, and a_{ij} is the similarity between profiles r_i and r_j. $\mathcal{L}_u$ gives a large penalty when the cosine distance[3] is large between the embeddings of two similar profiles. In Fig. 1, the top-right part illustrates the joint training of the semi-supervised HisRect featurizer $\boldsymbol{\mathcal{F}}$ and the POI classifier $\boldsymbol{\mathcal{P}}$.

The most important thing is how to measure the similarity between two profiles. As each profile contains latitude and longitude, it is natural to use the spatial distance between two profiles to compute their similarity. Also, the time dimension should be taken into consideration as two profiles tend to be less similar if they were posted at different times. By considering the spatial and temporal aspects, we derive the similarity matrix $\boldsymbol{A}$ as follows:

$$a_{ij} = \begin{cases} 1, & \text{if } \widehat{r_i r_j} \in \Gamma_L^+, \\ -1, & \text{if } \widehat{r_i r_j} \in \Gamma_L^-, \\ & \text{if } \widehat{r_i r_j} \in \Gamma_U \text{ and } d(r_i, r_j) < \rho \\ \frac{\epsilon'_d}{\epsilon'_d + d(r_i, r_j)}, & \text{and } d(r_i, P) < \rho \text{ and } d(r_j, P) < \rho \\ & \text{and } |r_i.ts - r_j.ts| < \Delta t, \\ 0, & \text{otherwise.} \end{cases}$$

In particular, ρ and Δt are the thresholds for spatial and temporal distances, respectively; ϵ'_d is another smoothing factor.

Two profiles should be similar if they are associated to the same POI in the same time period. In contrast, two profiles in a negative pair are supposedly different, as they are associated to different POIs. Accordingly, a penalty should be given if the distance is large between features of profiles in a positive pair. Otherwise, it gives a reward, *i.e.*, a negative penalty. Therefore, we set the similarity item a_{ij} for profiles in positive and negative pairs to be 1 and -1, respectively.

When a pair of profiles are not from the labeled set Γ_L, we cannot measure their similarity in the aforementioned way. For profile pairs from Γ_L, we utilize the lat, lon and ts in the profiles. Intuitively, the shorter the spatial distance between two profiles, the larger the similarity between them is. Given a profile pair, if their spatial distance is larger than ρ or their temporal distance is larger than Δt, we set their similarity to be 0. Furthermore, if a profile r was posted in a place close to no POI, *i.e.*, $d(r, P) > \rho$, we think it offers little useful information and set all the similarities involving r to be 0. Those relevant profiles are unlabeled and we are less confident to say that their features are totally different. Therefore, we do not set them to -1.

We train $\boldsymbol{\mathcal{F}}$, $\boldsymbol{\mathcal{E}}$ and $\boldsymbol{\mathcal{P}}$ as follows. First, we build the sets Γ_L and Γ_U from labeled profile set R_L and unlabeled profile set R_U, respectively, and calculate the similarity matrix $\boldsymbol{A}$. For $\boldsymbol{A}$, we only need to consider those pairs of profiles in $R_L \bigcup R_U$, as the weights of the pairs not in $\Gamma_L \bigcup \Gamma_U$ are all 0. These pairs have no impact on the penalty $\mathcal{L}_u$. After we obtain the aforementioned data, batches are sampled from R_L and $\Gamma_L \bigcup \Gamma_U$ according to the proportion of $R_L : \Gamma_L \bigcup \Gamma_U$. Subsequently, we feed the samples to the network to calculate $\mathcal{L}_{poi}$ and $\mathcal{L}_u$. Finally, by updating $\Theta_{\mathcal{F}}$, $\Theta_{\mathcal{P}}$ and $\Theta_{\mathcal{E}}$—the parameters of $\boldsymbol{\mathcal{F}}$, $\boldsymbol{\mathcal{E}}$ and $\boldsymbol{\mathcal{P}}$, respectively, we perform a stochastic gradient descent step with mini-batch Adam [56] to optimize the supervised loss $\mathcal{L}_{poi}$ and the unsupervised loss $\mathcal{L}_u$. The whole procedure is repeated for a number of iterations until $\mathcal{L}_{poi}$ and $\mathcal{L}_u$ are convergent. The whole training process is formalized in Algorithm 1.

Algorithm 1 Semi-supervised HisRect feature training

Input: $R_L, \Gamma_L, \Gamma_U, \boldsymbol{A}, B$

1: set $\Omega = |R_L| + |\Gamma_L \bigcup \Gamma_U|$
2: set $\gamma_{poi} = \frac{|R_L|}{\Omega}, \gamma_u = \frac{|\Gamma_L \bigcup \Gamma_U|}{\Omega}$
3: **repeat**
4: generate a random number $\gamma \in [0, 1)$
5: **if** $\gamma < \gamma_{poi}$ **then**
6: sample a batch of labeled profiles $\mathcal{B}_r \in R_L$ of size B
7: $\mathcal{L}_{poi} = -\frac{1}{B} \sum_{r \in \mathcal{B}_r} log(p_{poi}[r.pid])$
8: Take a gradient step to optimize $\mathcal{L}_{poi}$, update $\Theta_{\mathcal{F}}$ and $\Theta_{\mathcal{P}}$
9: **else**
10: sample a batch of pairs $\mathcal{B}_p \in \Gamma_L \bigcup \Gamma_U$ of size B
11: $\mathcal{L}_u = -\frac{1}{B} \sum_{\widehat{r_i r_j} \in \mathcal{B}_p} a_{ij} \|\boldsymbol{\mathcal{E}}\big(\boldsymbol{H}(r_i)\big), \boldsymbol{\mathcal{E}}\big(\boldsymbol{H}(r_j)\big)\|_2^2$
12: Take a gradient step to optimize $\mathcal{L}_u$, update $\Theta_{\mathcal{F}}$ and $\Theta_{\mathcal{E}}$
13: **until** $\mathcal{L}_{poi}$ and $\mathcal{L}_u$ all converge or are sufficiently small

5 HisRect-based Co-Location Judgement

The HisRect feature is suitable to solve the problem of co-location judgement. If two users have similar historical visits or send tweets from the same POI, they are more likely to be co-located. On the contrary, two users with very different visits histories are very likely to be in different places.

In order to judge if two profiles r_i and r_j are co-located, a simple method called $Comp2Loc$ uses the classifier $\boldsymbol{\mathcal{P}}$ to infer the POIs for both profiles and see if the two inferred POIs are identical. However, this method only considers part of the original HisRect featurse and utilizes the features of r_i and r_j separately. It lacks insight on the intrinsic properties of co-located pairs.

A more sophisticated method should consider the difference between the features of r_i and r_j and thus capture the intrinsic relationship between r_i and r_j. The bottom-right part of Fig. 1 shows the framework of our approach to the co-location judgement problem. We use an embedding layer $\boldsymbol{\mathcal{E}'}$ to embed the HisRect features of r_i and r_j. Also, we construct a feed-forward neural network $\boldsymbol{\mathcal{C}}$ whose input is the difference vector between the two embeddings. On top of $\boldsymbol{\mathcal{C}}$ follows a binary softmax layer. It is basically a logistic regression with sigmoid function and the corresponding cross entropy is reducible to a log loss function. The formula of $\boldsymbol{\mathcal{E}'}$, $\boldsymbol{\mathcal{C}}$, co-location probability estimate p_{co}, and the supervised loss are shown as follows.

$$\boldsymbol{\mathcal{E}'}(\mathbf{x}) = h^{Q'_e}\Big(...h^2\big(h^1(\mathbf{x})\big)\Big); \boldsymbol{\mathcal{C}}(\mathbf{x}) = h^{Q_c}\Big(...h^2\big(h^1(\mathbf{x})\big)\Big)$$

$$p_{co}(\widehat{r_i r_j}) = \boldsymbol{\sigma}\bigg(\boldsymbol{\mathcal{C}}\Big(\big|\boldsymbol{\mathcal{E}'}\big(\boldsymbol{\mathcal{F}}(r_i)\big) - \boldsymbol{\mathcal{E}'}\big(\boldsymbol{\mathcal{F}}(r_j)\big)\big|\Big)\bigg)$$

3. As $\boldsymbol{\mathcal{E}}\big(\boldsymbol{\mathcal{F}}(r_i)\big)$ and $\boldsymbol{\mathcal{E}}\big(\boldsymbol{\mathcal{F}}(r_j)\big)$ are normalized, we do not need to divide the product of their ℓ^2-norm.

$$\mathcal{L}_{co} = -\sum_{\widehat{r_i r_j} \in \Gamma_L^+} log(p_{co}(\widehat{r_i r_j})) - \sum_{\widehat{r_i r_j} \in \Gamma_L^-} log(1 - p_{co}(\widehat{r_i r_j}))$$

Above, Q'_e and Q_c are the numbers of fully connected layers in $\boldsymbol{\mathcal{E}'}$ and $\boldsymbol{\mathcal{C}}$, respectively. When $p_{co}(\widehat{r_i r_j})$ is larger than a threshold (it is set to 0.5 generally), the profiles r_i and r_j are regarded as co-located.

To train $\boldsymbol{\mathcal{E}'}$ and $\boldsymbol{\mathcal{C}}$, we only need the labeled pairs set Γ_L. In each training iteration, we sample batches from Γ_L, calculate corresponding $\mathcal{L}_{co}$, and take a gradient decent step until $\mathcal{L}_{co}$ converges. Note that the parameters $\Theta_{\mathcal{F}}$ of $\boldsymbol{\mathcal{F}}$ are fixed at this stage.

We can also connect the HisRect featurizer $\boldsymbol{\mathcal{F}}$ with $\boldsymbol{\mathcal{E}'}$ and $\boldsymbol{\mathcal{C}}$ directly and take $\mathcal{L}_{co}$ as loss objective to train the parameters $\Theta_{\mathcal{F}}$, $\Theta_{\mathcal{E}'}$ and $\Theta_{\mathcal{C}}$ jointly using labeled pairs Γ_L. This approach, called *One-phase*, omits the process of HisRect feature training. As some profiles in R_L may not show in any pair in Γ_L, One-phase may fail to exploit useful information. Also, One-phase cannot be trained in a semi-supervised way since it does not use unlabeled data. Experiments show that our approach outperforms One-phase.

Our proposed co-location approach can be easily wrapped into an efficient clustering algorithm. Given N profiles, we can get an $N \times N$ probability matrix S with each item $S_{i,j}$ representing the similarity of profiles r_i and r_j. By taking each profile as a node and linking an edge between node i and j if $S_{i,j}$ is larger than a threshold (in general, it is set to 0.5), this matrix is converted into an undirected graph. Consequently, the clusters are just the connected components of the graph. We do not even need to designate the number of clusters. The experimental results demonstrate that our approach works well on clustering tasks.

6 EXPERIMENTS

6.1 Experimental Settings

6.1.1 Datasets

We use the open-source library *twitter4j*[4] to access Twitter's open API to crawl data. We crawl timelines of Twitter users whose profile location is in one of New York's five boroughs[5] (NYC for short) or Clark County (including Las Vegas, LV for short) in Nevada. Totally, there are 892,172 NYC and 207,682 LV Twitter user timelines, involving 992,390,010 and 148,021,872 tweets, respectively. Only 2.2 percent of NYC tweets and 2.0 percent of LV tweets are geo-tagged. In addition, we download NYC and LV OpenStreetMap data dump[6] and extract all POI bounding polygons. By checking the coordinates in those geo-tagged tweets against the POI bounding polygons, we identify all POI tweets, *i.e.*, those sent in a POI. In the data we use, most POIs involve no tweets. Therefore, we only consider the top 1000 POIs in NYC and top 250 POIs in LV that have the most tweets. We filter out the user timelines that contain no POI tweet and obtain 58,966 and 10,844 user timelines in NYC and LV, respectively. We randomly select $\frac{1}{5}$ of these timelines to form the testing dataset. The remaining timelines are split into training and validation data in the ratio of $9:1$.

We obtain labeled profiles set R_L^{train}, labeled pairs set Γ_L^{train}, and unlabeled pairs set Γ_U^{train} for the training dataset. In the testing or validation dataset, only the labeled profiles set R_L^{test} and labeled pairs set Γ_L^{test} are needed. More details about the datasets are given in Table 2.

6.1.2 Training and Implementation

Each stopword[7] in the content of all profiles is replaced with a </s> symbol firstly. Since the ratio between the numbers of negative/unlabeled and positive pairs is very large, we use $\frac{1}{10}$ of negative and unlabeled pairs only in every training epoch, i.e., negative and unlabeled pairs can be gone through in every 10 epochs.

It is noteworthy that our approach design is independent of the Δt in the problem setting. We conduct some preliminary experiments using different Δt values. The performance results are very stable despite the varying Δt. Therefore, for both training and testing datasets, we set Δt to be 1 hour in subsequent experiments.

Other important details are as follows:

- We replace each stop word with a </s> and only consider those words appearing more than 10 times when training word embeddings.
- The parameters of the LSTM and all the fully connected layers are initialized with Gaussian noise with mean being 0 and standard deviation set to be 0.01.
- We initialize the initial state of the LSTM with 0.
- To avoid exploding gradient problem, we enforced a hard constraint on the norm of gradient by scaling it when its norm exceeds a threshold [0, 5].
- We use dropout and the configuration is set to keep probability of 0.8 [57] at the LSTM layer and before every fully connected layer during training. These are not involved when applying the model on testing dataset.
- We use three Adam optimizers to minimize $\mathcal{L}_{poi}$, $\mathcal{L}_u$ and $\mathcal{L}_{co}$ respectively. To avoid overfitting, we add a l^2-regularization term on these three loss functions.
- We perform SGD with mini-batch Adam, started with learning rates of 0.01 for all the three optimizers. The coefficients of learning rates and l^2-regularization terms all decrease with the number of training iterations increasing.
- We set $\Delta t = 1h$, $\epsilon_d = 1000m$, $\epsilon'_d = 50m$ and $\rho = 1000m$, respectively.

6.1.3 Evaluation Metrics and Approaches

For performance comparison with other co-location judgement approaches, we apply four widely-used metrics, *i.e.*, Acc (accuracy), Rec (recall), Pre (precision) and F_1 (F1-score, $F_1 = \frac{2}{\frac{1}{Rec} + \frac{1}{Pre}}$).

Table 3 summarizes the differences among all approaches in our experiments. HV is short for historical visits. Our proposed approach is called *HisRect*; *HisRect-SL* is the same but uses the supervised HisRect training only. An approach is FF (short for Feature-first) if it extracts features for both profiles in a pair first, followed by using the features to make the judgement. *One-phase* is not a FF approach as it does not use an explicit step to extract features.

A Naive approach infers the locations of two profiles and checks if the two locations are identical. In our experiments, we implement three naive approaches: $Comp2Loc$ and two exiting tweet location inference approaches called $TG\text{-}TI\text{-}C$ [22] and

4. http://twitter4j.org
5. The five boroughs are: Manhattan, Brooklyn, The Bronx, Staten Island and Queens.
6. https://wiki.openstreetmap.org/wiki/Planet.osm
7. https://www.ranks.nl/stopwords

TABLE 2: Dataset statistics

Dataset		#timeline	#labeled profiles	#avg visits/profile	#pos-pairs	#neg-pairs	#unlabeled pairs
NYC	Training	42,454	480,646	42.94	243,080	5,817,634	4,432,984
	Validation	4,718	52,754	43.56	2,867	69,313	None
	Testing	11,794	129,003	44.16	18,219	418,091	None
LV	Training	7,835	130,455	24.46	25,711	366,423	163,160
	Validation	871	12,225	23.64	270	2,935	None
	Testing	2,177	36,989	23.35	2,211	28,596	None

TABLE 3: Eleven co-location judgement approaches

Approach	HV	Tweet	SSL	FF	Naive
N-Gram-Gauss	×	✓	×	×	✓
TG-TI-C	×	✓	×	×	✓
Comp2Loc	✓	✓	✓	✓	✓
One-phase	✓	✓	×	×	×
History-only	✓	×	✓	✓	×
Tweet-only	×	✓	✓	✓	×
HisRect-SL	✓	✓	×	✓	×
One-hot	✓	✓	✓	✓	×
BLSTM	✓	✓	✓	✓	×
ConvLSTM	✓	✓	✓	✓	×
HisRect	✓	✓	✓	✓	×

N-$Gram$-$Gauss$ [18]. In particular, TG-TI-C infers tweet locations using similarity comparison between a tweet and a set of geo-tagged tweets, whereas N-$Gram$-$Gauss$ trains a Gaussian model for the distribution of geo-specific n-grams and uses that model to discover the geographic scope for a given n-gram.

Also, to investigate if our design of HisRect is effective at capturing historical visits and recent tweet contents, we build another kind of feature which uses a one-hot encoding to model a user's visit history and featurizes the recent tweet in the same way with HisRect.

In addition, to study the effect of the convolution layer of the architecture $BiLSTM$-C, we build another neural network model, named $BLSTM$, that only uses bidirectional LSTM and omits the convolution layer from $BiLSTM$-C. Moreover, we implement $ConvLSTM$ [58] which uses convolutional structures instead of fully-connected layers in both the input-to-state and state-to-state transitions. Unlike our HisRect, these two approaches use bidirectional LSTM or ConvLSTM without the following convolution layer when extracting features of tweet contents.

The original testing set contains significantly more negative pairs than positive pairs. In order to have clear comparison, we split the negative pairs into 10 parts, merge each of them with the positive pairs to form 10 testing sets instead. The reported results of each approach are the average over the 10 testing sets.

6.2 Experimental Results on Co-Location Judgement

Table 4 reports the overall performance results of all the eleven approaches. The results show that our HisRect approach is overall the best in terms of all metrics. Moreover, the ROC-curves of all approaches but the three naive ones are presented in Figure 2. The naive approaches are excluded because it is impossible to set the thresholds of the false positive rates for them. All of the tested approaches are trained with the same training dataset and their parameters are tuned to the best separately. It is also evident from Figure 2 that our HisRect performs best. Its AUC values are 0.974 and 0.957 in NYC and LV datasets, respectively. In Co-location judgement problems, judging the co-located pairs rightly matters more, *i.e.*, we hope to get a higher recall on the prerequisite of a relatively high accuracy. Considering the low rate of positive pairs, we use $\frac{1}{10}$ of negative and unlabeled pairs only in every training epoch to increase the proportion of labeled pairs. Similarly, we split the negative pairs into 10 parts and merge each of them with the positive pairs to form 10 testing sets in order to have clear comparison (Section 6.1.1). Overall, the approaches that based on "historical visits + tweet contents" type feature almost get high AUC values and performs much better than other methods. Our HisRect outperforms the state-of-the-art alternatives. Subsequently, we further compare the performance differences and disclose the reasons behind.

6.2.1 Comparison with Existing Approaches

HisRect performs much better than TG-TI-C and N-Gram-Gauss on all of the three metrics. Even Comp2Loc, another naive approach, also outperforms them. Compared with HisRect, Comp2Loc yields worse results on these metrics except Pre. Comp2Loc judges a pair to be co-located only when $\mathcal{P}$ classifies both profiles in the pair into the same POI. In this case, the HisRect features of both profiles are very similar and thus Comp2Loc achieves high Pre by using the features. However, the HisRect features of the two profiles which are involved in the same POI may focus on different aspects. As a result, Comp2Loc cannot understand the intrinsic relationships. Therefore, it performs worse in terms of Acc, Rec and F_1.

6.2.2 The Effect of HisRect Features

Compared to History-only and Tweet-only, HisRect-SL and HisRect clearly improve the performance of co-location judgement. These results indicate that our HisRect features are more powerful than those features that only consider either visit history or recent tweets. Besides, HisRect outperforms One-hot. Thus, it is reasonable to say that HisRect utilizes the historical visits in a better way. Moreover, HisRect achieves better performance than BLSTM and ConvLSTM. This shows that the $BiLSTM$-C structure in HisRect features is more suitable for extracting features of tweet contents and our complete HisRect features model the historical visits more effectively.

6.2.3 The Effect of Semi-supervised Training

Both HisRect-SL and One-phase are inferior to HisRect on these four metrics. Such performance differences clearly demonstrate the power of semi-supervised learning framework that leverages unlabeled data in our approach.

6.3 The Power of HisRect Features

To understand the power of HisRect features in different settings, we also design more experiments and report relevant results.

6.3.1 Historical Visits or Tweet Contents Are Not Available

In order to verify the power of HisRect features, we investigate whether HisRect can work well if only historical visits or tweet contents are used in the features. We carry out experiments with variants of relevant approaches. We remove the visit history of

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2934686, IEEE Transactions on Knowledge and Data Engineering

TABLE 4: Performance of different approaches

Approach	NYC Dataset				LV Dataset			
	Acc	Rec	Pre	F_1	Acc	Rec	Pre	F_1
TG-TI-C	0.7367	0.4388	0.5980	0.5062	0.6707	0.4374	0.6944	0.5367
N-Gram-Gauss	0.7769	0.5110	0.6751	0.5817	0.7102	0.4826	0.7663	0.5922
Comp2Loc	0.9106	0.7274	0.9709	0.8317	0.8283	0.6196	0.9791	0.7590
History-only	0.7942	0.5366	0.7143	0.6128	0.7525	0.6139	0.7721	0.6840
Tweet-only	0.8735	0.7316	0.8314	0.7783	0.8037	0.6482	0.8685	0.7423
One-phase	0.9017	0.8045	0.8622	0.8324	0.8414	0.7227	0.8932	0.7989
HisRect-SL	0.9222	0.8446	0.9018	0.8669	0.8781	0.8061	0.9040	0.8522
One-hot	0.8805	0.7424	0.8450	0.7904	0.8011	0.6812	0.8324	0.7493
BLSTM	0.9186	0.8252	0.8985	0.8603	0.8762	0.7908	0.9143	0.8481
ConvLSTM	0.9135	0.8266	0.8811	0.8530	0.8662	0.7849	0.8955	0.8366
HisRect	**0.9341**	**0.8618**	**0.9162**	**0.8881**	**0.8981**	**0.8348**	**0.9242**	**0.8772**

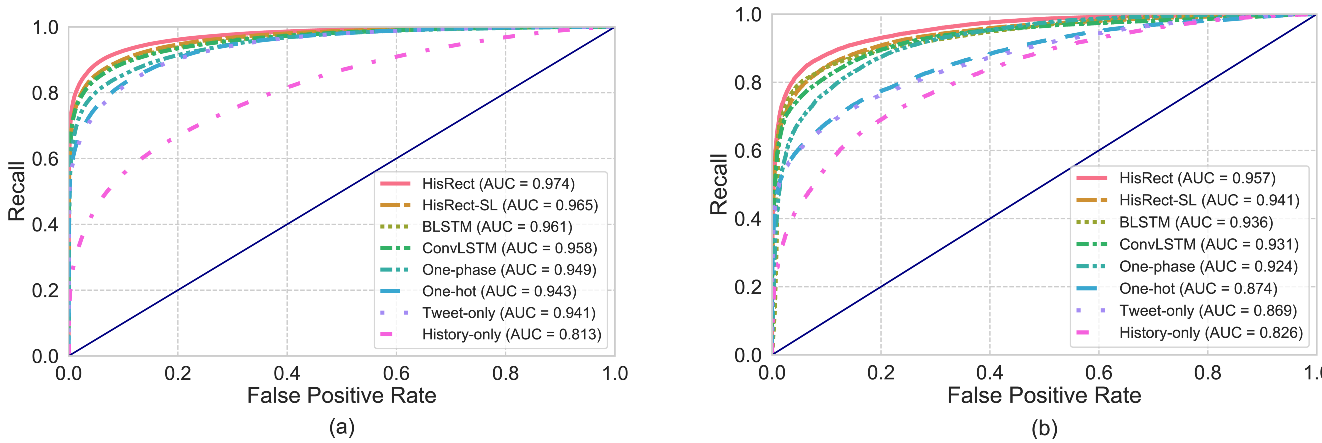


Fig. 2: The ROC-curve and AUC of different approaches. (a): NYC dataset; (b): LV dataset

every profile in Γ_L^{test} to obtain a new dataset $\Gamma_L^{test}\backslash H$. Also, by replacing each word in the tweet contents of every profile in Γ_L^{test} with the symbol </s>, we obtain another dataset $\Gamma_L^{test}\backslash T$. We test our well-trained HisRect model on these two datasets. For convenience, the model on $\Gamma_L^{test}\backslash H$ and $\Gamma_L^{test}\backslash T$ are called $HisRect\backslash H$ and $HisRect\backslash T$, respectively. Table 5 reports the comparative results of three HisRect-based approaches with History-only and Tweet-only.

TABLE 5: Comparison among HisRect-based approaches, History-only and Tweet-only on NYC dataset

Approach	Acc	Rec	Pre	F_1
HisRect\T	0.7607	0.4495	0.6539	0.5328
HisRect\H	0.8319	0.7721	0.7032	0.7361
History-only	0.7942	0.5366	0.7143	0.6128
Tweet-only	0.8735	0.7316	0.8314	0.7783
HisRect	**0.9341**	**0.8618**	**0.9162**	**0.8881**

It is apparent that HisRect performs badly if the testing dataset is composed by historical visits or tweet contents only. Its results are even worse than that of History-only when tweet contents are missing. Without historical visits in Γ_L^{test}, Tweet-only outperforms HisRect. However, HisRect is clearly the best when the dataset is complete, which indicates that HisRect is able to establish useful linkages between historical visits and tweet contents. As most profiles' $v\text{-}history$ are not empty in real word data, our proposed approach can work well in real world applications.

6.3.2 HisRect Visualization

In order to better understand why HisRect works well, we leverage labeled and unlabeled data together to generate HisRect features and observe if such features are good expressions of raw data or not. We get the HisRect features $\mathcal{F}(r)$ for every profile r in the top-5 POIs in the testing dataset. Due to the high number of dimensions, we use t-SNE [59] transformation to visualize HisRect features, as shown in Figure 3.

It can be seen from Figure 3 that only a small central part mixes many different POIs and results in chaos. According to our observation, the contents in the profiles displayed in the center either have no tie with any POIs or are noises, and the corresponding visit histories offer little information. Local clues can be hardly found in these profiles. Except for the slight chaos, most profiles that come from the same POIs are adjacent to each other and form clusters as shown in Figure 3. Therefore, it is reasonable to say that HisRect featurizer $\mathcal{F}$ are able to extract good features of profiles.

6.3.3 POI Inference Based on HisRect

We also investigate the performance of HisRect features on the POI inference problem, i.e., to infer the POI of a tweet without geo-tag. We compare the inference accuracy of HisRect with the following approaches:

- *History-only* only utilizes visit histories in profiles. The rest is the same with HisRect.
- *Tweet-only* only utilizes the contents of tweets in profiles. The rest is the same with HisRect.
- *One-hot* model the visit histories using one-hot encoding vectors.
- *HisRect-SL* does not leverage unlabeled pairs when training HisRect features. The rest is the same with HisRect.
- *BLSTM* uses bidirectional LSTM when training features of tweet contents. The rest is the same with HisRect.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2934686, IEEE Transactions on Knowledge and Data Engineering

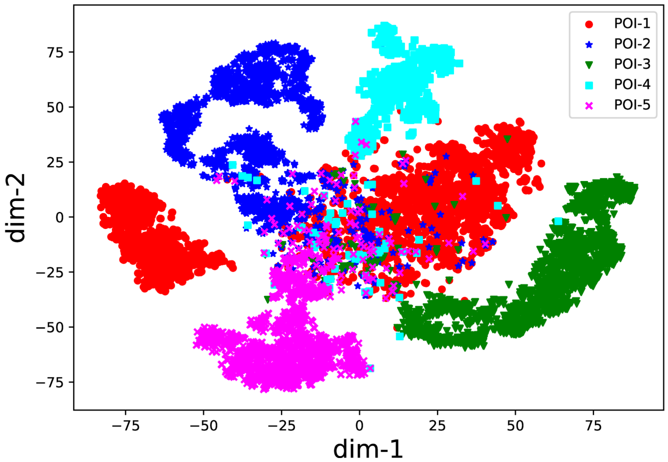


Fig. 3: 2-dimensional t-SNE projection of HisRect features

- *ConvLSTM* uses ConvLSTM [58] when training features of tweet contents. The rest is the same with HisRect.
- *N-Gram-Gauss* [18] learns a Gaussian model for the distribution of geo-specific n-grams and applies that model to discover the geographic scope for a given n-gram.
- *TG-TI-C* [22] infers tweet locations using similarity comparison between a tweet and a given set of geo-tagged tweets. It exploits the contents and time-evolution characteristics.

We use the metric $Acc@K$ to evaluate the performance of these approaches on NYC and LV datasets, where K is the number of POI candidates each model infers. Figure 4 show the performance of all the nine approaches. It is clear that our HisRect performs the best. It is the only one that achieves accuracy higher than 64% and 70% on NYC and LV datasets, respectively.

Furthermore, we split every R_L^{test} of two datasets into two parts: TR_L^{test} contains those profiles that either History-only or Tweet-only can infer correctly, and FR_L^{test} is the set of the opposite cases. Table 6 reports the accuracy of HisRect inferring the POIs on both datasets. For NYC dataset, 92.09% of profiles in TR_L^{test} can be inferred correctly by HisRect. It shows that the combination of visit history and recent tweet is able to capture the typical features of either information source. Even when neither History-only nor Tweet-only can infer POIs of any profiles in FR_L^{test} correctly, HisRect is still able to achieve an accuracy of more than 26%. The accuracies on LV dataset are similar with that of NYC dataset. Again, these results demonstrate the power of HisRect features.

TABLE 6: Accuracy of HisRect on TR_L^{test} and FR_L^{test}

Dataset	TR_L^{test}		FR_L^{test}	
	Number	*Acc*	Number	*Acc*
NYC	76,735	0.9209	52,468	0.2635
LV	26,653	0.9112	10,336	0.3247

6.4 Parameters Study

6.4.1 Effect of Training Dataset Size

In order to investigate how the amount of training data affects the performance, we design an experiment as follows. We randomly pick up 10%, 20%, .., 90% and 100% of the user timelines in the NYC training dataset, obtaining corresponding labeled profiles, labeled and unlabeled pairs to train all approaches. Figure 5 illustrates the comparative recalls of the ten approaches with varying amount of training timelines. The figure also reports the ratios between positive, negative, unlabeled pairs and labeled profiles of varying amount of training timelines with respect to the corresponding statistics.

Clearly, all the ten approaches work better with more training data. More training data enable them to extract better features that in turn result in more accurate identifications. Nevertheless, HisRect can achieve good performance even when the amount of training dataset is very small. This shows that our model is equipped with a more powerful expression ability and less sensitive to the amount of training data.

6.4.2 Effect of Deep Learning Parameters

Since deep learning is used widely, we want to explore if deeper architectures can bring about improvements to the resolving of the research problem in this work. We vary the number of fully connected layers (Q_f) and that of stacked bidirectional LSTM layers in HisRect featurizer $\mathcal{F}$ (Q_l). Table 7 shows the performance with different neural architectures.

TABLE 7: Recall and accuracy in different settings

Rec	$Q_l = 1$	$Q_l = 2$	$Q_l = 3$	$Q_l = 4$
$Q_f = 1$	0.8384	0.8400	0.8557	0.8394
$Q_f = 2$	0.8307	0.8583	**0.8618**	0.8546
$Q_f = 3$	0.8354	0.8452	0.8484	0.8461
Acc	$Q_l = 1$	$Q_l = 2$	$Q_l = 3$	$Q_l = 4$
$Q_f = 1$	0.9160	0.9201	0.9241	0.9209
$Q_f = 2$	0.9175	0.9290	**0.9341**	0.9259
$Q_f = 3$	0.9125	0.9247	0.9264	0.9166

From Table 7, we find that a deeper architecture does not necessarily improve the performance. When $Q_f = 2$ and $Q_l = 3$, the corresponding recall and accuracy are the highest. This observation is also seen in the experiments with different Q_c, Q_e and $Q_{e'}$, where the optimal parameter values are 3, 2 and 2, respectively.

6.4.3 Comparison with Other SSL Alternatives

We alter the cosine distance with ℓ^2 norm of two embeddings' difference in calculating the unsupervised loss, which is the case in [46]. The best accuracy and recall on NYC dataset are 0.9232 and 0.8453, respectively. If we remove the embedding $\mathcal{E}$ in the formula of the unsupervised loss, the corresponding accuracy and recall are 0.9237 and 0.8515, respectively. These results are worse than the performance of HisRect with the best configurations. Therefore, our HisRect has clear advantages over other SSL approach in the problem of co-location judgement.

6.4.4 Computation Time and Scalability

Once the HisRect featurizer $\mathcal{F}$ and the co-location judge $\mathcal{C}$ are trained, the HisRect feature construction and co-location judgement can both be finished in 1 *ms* for a given profile pair. Besides, the computation time of converting raw tweets and timelines into profiles and pairs is also very short (less than 1 *ms* per tweet). Thus, our approach can work in online scenarios.

We also investigate the scalability of our training procedures. We randomly pick up 10%, 20%, .., 90% and 100% of the user timelines in the NYC training dataset. Figure 6 reports the average training time of two models (the HisRect Featuizer and the HisRect model for co-location judgement) per sample in an episode with respect to different amounts of timelines. Here, a sample

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2934686, IEEE Transactions on Knowledge and Data Engineering

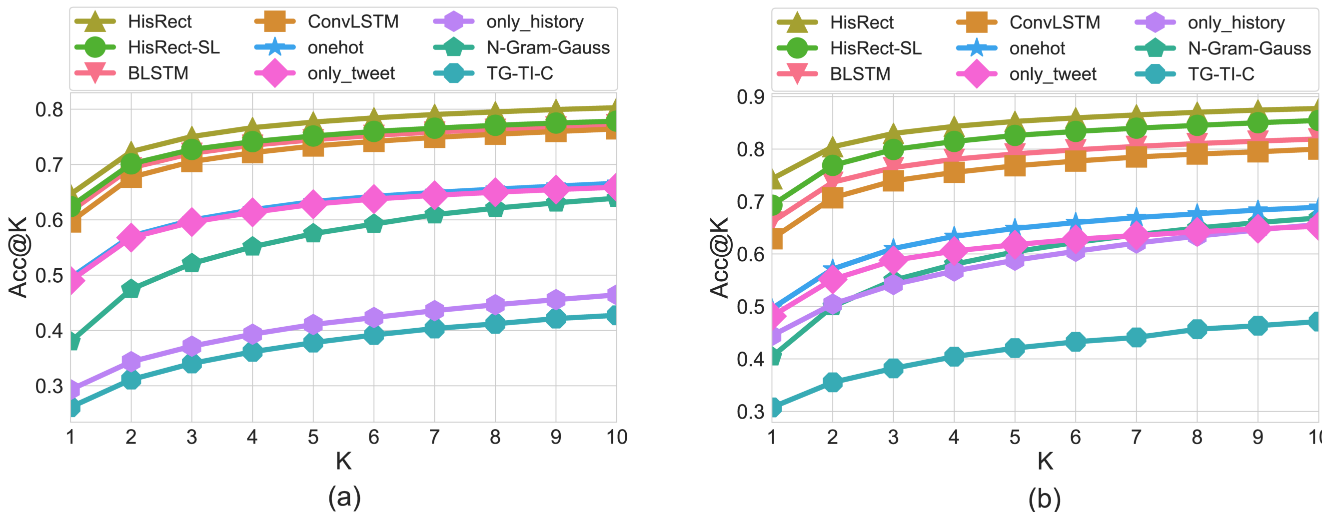


Fig. 4: $Acc@K$ of different approaches with varying K. (a): NYC dataset; (b): LV dataset

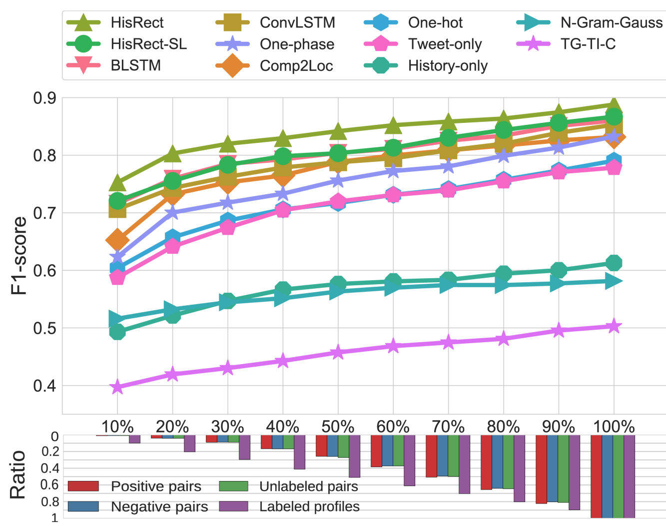


Fig. 5: F1-scores and ratio between data types

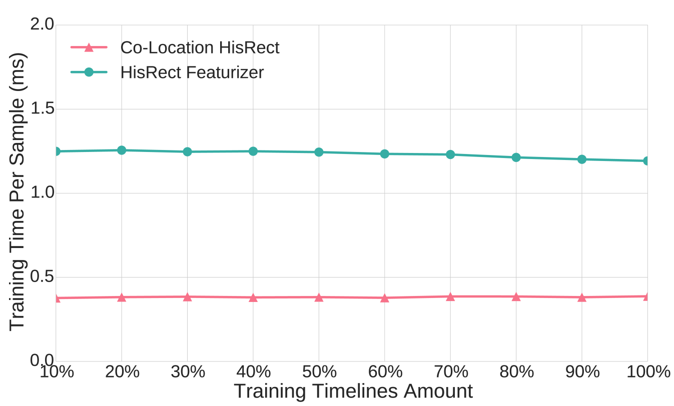


Fig. 6: Average training time per sample

is either a pair or a profile, depending on the kind of data fed into models. Specifically, the input samples of HisRect featurizer contain Γ_L^{train}, Γ_U^{train} and R_L^{train}. However, those of HisRect co-location model contain Γ_L^{train} only. Suppose the training time of HisRect featurizer and HisRect co-location model over one episode is T_F and T_C, respectively. The average training time per sample of these two models is $\frac{T_F}{|\Gamma_L^{train}|+|\Gamma_U^{train}|+|R_L^{train}|}$ and $\frac{T_C}{|\Gamma_L^{train}|}$, respectively. Figure 6 shows that the training time per sample of these two models is almost constant, roughly 0.4 and 1.25 ms, respectively.

6.5 A Case Study: Using a Co-location Judgement Approach to Cluster User Profiles

In many applications such as local user recommendation [6], community detection [60] and group analysis [10], people often care more about identifying who are in the same POI given a group of profiles. To address this issue, we design an experiment as follows. We sample groups of profiles with each group containing 5 profiles and design 5 typical patterns: 5-0, 4-1, 3-2, 3-1-1, 2-2-1. Pattern 3-2 means 3 out of 5 profiles (numbered a, b and c) are located in one POI and the other two (numbered d and e) are located in another POI. The meanings of other patterns are similar. Given a 3-2 pattern, if an approach can identify that a, b and c are co-located in a POI and d, e are located in another POI, we regard that the approach can identify the group pattern correctly. Specifically, we use HisRect to generate a 5×5 probability matrix and find all connected components as clusters. We compare if the predicted clusters are the same with the actual ones. We randomly select 2,000 groups of each different patterns. Table 8 shows the accuracy of HisRect and the other three alternative approaches on identifying group patterns on NYC testing dataset.

TABLE 8: Accuracies of four approaches on identifying group patterns on NYC datasets

Approach	5-0	4-1	3-2	3-1-1	2-2-1
HisRect	**0.8753**	**0.7915**	**0.7658**	**0.6198**	**0.5821**
Comp2Loc	0.0392	0.0579	0.0703	0.1250	0.1767
N-Gram-Gauss	0.0113	0.0339	0.0503	0.0838	0.1237
TG-TI-C	0.0046	0.0189	0.0216	0.0715	0.0984

Referring to Table 8, it can be seen that HisRect yields much higher accuracy than other approaches when identifying group patterns. In particular, the accuracies of HisRect on identifying patterns on these datasets are all larger than 58%. These results demonstrate that HisRect is effective at co-location and clustering tasks, whereas the alternative approaches excel in neither.

7 Conclusion and Future Work

In this paper, we propose a novel approach to judge if two Twitter users stay at the same POI in the same period of time. Our approach profiles each user by taking into account both visit histories and recent tweets. From such profiles, HisRect features are extracted by a HisRect featurizer. A semi-supervised learning framework with profile embedding is employed to train the featurizer. The HisRect features of two user profiles are fed to another embedding layer. Subsequently, a feed-forward network takes their embeddings difference as input and decides whether the pair is co-located in one of those pre-defined POIs. We conduct extensive experiments using large real datasets collected from Twitter. The experimental results demonstrate that our approach achieves high accuracy, recall, precision and F1-score, and clearly outperform alternative approaches.

The proposed approach in this paper uses visit histories and tweet contents. For future work, it is interesting to consider other information such as social relationship among users and frequent patterns shared by users. Such information may be utilized to build better similarities for user profiles and thus help improve the performance of co-location judgement.

Acknowledgments

This research is supported by Natural Science Foundation of China (No.61772460), Ten Thousand Talent Program of Zhejiang Province (No.2018R52039)

References

[1] S. Zhao, J. Ramos, J. Tao, Z. Jiang, S. Li, Z. Wu, G. Pan, and A. K. Dey, "Discovering different kinds of smartphone users through their application usage behaviors," in *Ubicomp*, 2016, pp. 498–509.

[2] S. Zhao, G. Pan, Y. Zhao, J. Tao, J. Chen, S. Li, and Z. Wu, "Mining user attributes using large-scale APP lists of smartphones," *IEEE Systems Journal*, vol. 11, no. 1, pp. 315–323, 2017.

[3] Z. Cheng, J. Caverlee, and K. Lee, "You are where you tweet: A content-based approach to geo-locating twitter users," in *CIKM*, 2010.

[4] F. Morstatter, H. Gao, and H. Liu, "Discovering location information in social media," *IEEE Data Eng. Bull.*, vol. 38, no. 2, pp. 4–13, 2015.

[5] J. Li, J. R. Thomsen, M. L. Yiu, and N. Mamoulis, "Efficient notification of meeting points for moving groups via independent safe regions," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 7, pp. 1767–1781, 2015.

[6] J. Jiang, H. Lu, B. Yang, and B. Cui, "Finding top-k local users in geo-tagged social media data," in *ICDE*, 2015, pp. 267–278.

[7] J. Jiang, H. Lu, P. Li, G. Pan, and X. Xie, "Finding influential local users with similar interest from geo-tagged social media data," in *MDM*, 2017, pp. 82–91.

[8] H. Pham and C. Shahabi, "Spatial influence - measuring followship in the real world," in *ICDE*, 2016, pp. 529–540.

[9] H. Yin, Z. Hu, X. Zhou, H. Wang, K. Zheng, N. Q. V. Hung, and S. W. Sadiq, "Discovering interpretable geo-social communities for user behavior prediction," in *ICDE*, 2016, pp. 942–953.

[10] J. Li, X. Wang, K. Deng, X. Yang, T. Sellis, and J. X. Yu, "Most influential community search over large social networks," in *ICDE*, 2017, pp. 871–882.

[11] G. Pan, W. Zhang, Z. Wu, and S. Li, "Online community detection for large complex networks," *PloS one*, vol. 9, no. 7, p. e102799, 2014.

[12] X. Zheng, J. Han, and A. Sun, "A survey of location prediction on twitter," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 9, pp. 1652–1671, 2018.

[13] A. Zubiaga, A. Voss, R. Procter, M. Liakata, B. Wang, and A. Tsakalidis, "Towards real-time, country-level location classification of worldwide tweets," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 9, pp. 2053–2066, 2017.

[14] M. Belkin, P. Niyogi, and V. Sindhwani, "Manifold regularization: A geometric framework for learning from labeled and unlabeled examples," *JMLR*, vol. 7, pp. 2399–2434, 2006.

[15] S. Kinsella, V. Murdock, and N. O'Hare, ""i'm eating a sandwich in glasgow": modeling locations with tweets," in *SMUC*, 2011, pp. 61–68.

[16] D. Doran, S. S. Gokhale, and A. Dagnino, "Accurate local estimation of geo-coordinates for social media posts," in *SEKE*, 2014, pp. 642–647.

[17] R. Priedhorsky, A. Culotta, and S. Y. D. Valle, "Inferring the origin locations of tweets with quantitative confidence," in *CSCW*, 2014, pp. 1523–1536.

[18] D. Flatow, M. Naaman, K. E. Xie, Y. Volkovich, and Y. Kanza, "On the accuracy of hyper-local geotagging of social media content," in *WSDM*, 2015, pp. 127–136.

[19] Q. Yuan, G. Cong, Z. Ma, A. Sun, and N. Magnenat-Thalmann, "Who, where, when and what: discover spatio-temporal topics for twitter users," in *KDD*, 2013, pp. 605–613.

[20] M. Dredze, M. Osborne, and P. Kambadur, "Geolocation for twitter: Timing matters," in *HLT-NAACL*, 2016, pp. 1064–1069.

[21] P. Paraskevopoulos and T. Palpanas, "Fine-grained geolocalisation of non-geotagged tweets," in *ASONAM*, 2015, pp. 105–112.

[22] ——, "Where has this tweet come from? fast and fine-grained geolocalization of non-geotagged tweets," *Social Netw. Analys. Mining*, vol. 6, no. 1, pp. 89:1–89:16, 2016.

[23] A. Noulas, S. Scellato, N. Lathia, and C. Mascolo, "Mining user mobility features for next place prediction in location-based services," in *ICDM*, 2012, pp. 1038–1043.

[24] K. Ryoo and S. Moon, "Inferring twitter user locations with 10 km accuracy," in *WWW*, 2014, pp. 643–648.

[25] J. McGee, J. Caverlee, and Z. Cheng, "A geographic study of tie strength in social media," in *CIKM*, 2011, pp. 2333–2336.

[26] ——, "Location prediction in social media based on tie strength," in *CIKM*, 2013, pp. 459–468.

[27] Y. Yamaguchi, T. Amagasa, H. Kitagawa, and Y. Ikawa, "Online user location inference exploiting spatiotemporal correlations in social streams," in *CIKM*, 2014, pp. 1139–1148.

[28] L. Kong, Z. Liu, and Y. Huang, "SPOT: locating social media users based on social network context," *PVLDB*, vol. 7, no. 13, pp. 1681–1684, 2014.

[29] G. Ference, M. Ye, and W. Lee, "Location recommendation for out-of-town users in location-based social networks," in *CIKM*, 2013, pp. 721–726.

[30] E. Cho, S. A. Myers, and J. Leskovec, "Friendship and mobility: user movement in location-based social networks," in *KDD*, 2011, pp. 1082–1090.

[31] C. Zhang, L. Shou, K. Chen, G. Chen, and Y. Bei, "Evaluating geo-social influence in location-based social networks," in *CIKI*, 2012, pp. 1442–1451.

[32] W. Zhu, W. Peng, L. Chen, K. Zheng, and X. Zhou, "Modeling user mobility for location promotion in location-based social networks," in *KDD*, 2015, pp. 1573–1582.

[33] C. Yang, L. Bai, C. Zhang, Q. Yuan, and J. Han, "Bridging collaborative filtering and semi-supervised learning: A neural approach for POI recommendation," in *KDD*, 2017, pp. 1245–1254.

[34] C. Cheng, H. Yang, M. R. Lyu, and I. King, "Where you like to go next: Successive point-of-interest recommendation," in *IJCAI*, 2013, pp. 2605–2611.

[35] J. Zhang, C. Chow, and Y. Li, "LORE: exploiting sequential influence for location recommendations," in *SIGSPATIAL*, 2014, pp. 103–112.

[36] W. Zhang and J. Wang, "Location and time aware social collaborative retrieval for new successive point-of-interest recommendation," in *CIKM*, 2015, pp. 1221–1230.

[37] H. Yin, Y. Sun, B. Cui, Z. Hu, and L. Chen, "LCARS: a location-content-aware recommender system," in *KDD*, 2013, pp. 221–229.

[38] W. Wang, H. Yin, L. Chen, Y. Sun, S. W. Sadiq, and X. Zhou, "Geo-sage: A geographical sparse additive generative model for spatial item recommendation," in *KDD*, 2015, pp. 1255–1264.

[39] W. Wang, H. Yin, S. W. Sadiq, L. Chen, M. Xie, and X. Zhou, "SPORE: A sequential personalized spatial item recommender system," in *ICDE*, 2016, pp. 954–965.

[40] W. Wang, H. Yin, X. Du, Q. V. H. Nguyen, and X. Zhou, "TPM: A temporal personalized model for spatial item recommendation," *ACM TIST*, vol. 9, no. 6, pp. 61:1–61:25, 2018.

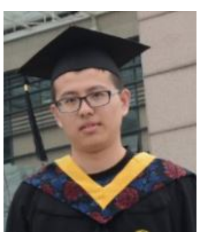
[41] M. Xie, H. Yin, H. Wang, F. Xu, W. Chen, and S. Wang, "Learning graph-based POI embedding for location-based recommendation," in *CIKM*, 2016, pp. 15–24.

[42] H. Yin, X. Zhou, B. Cui, H. Wang, K. Zheng, and N. Q. V. Hung, "Adapting to user interest drift for POI recommendation," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 10, pp. 2566–2581, 2016.

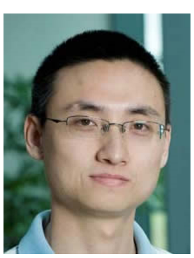
[43] H. Yin, W. Wang, H. Wang, L. Chen, and X. Zhou, "Spatial-aware hierarchical collaborative deep learning for POI recommendation," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 11, pp. 2537–2551, 2017.

[44] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," in *NIPS*, 2003, pp. 321–328.

[45] X. Zhu, Z. Ghahramani, and J. D. Lafferty, "Semi-supervised learning using gaussian fields and harmonic functions," in *ICML*, 2003, pp. 912–919.
[46] J. Weston, F. Ratle, H. Mobahi, and R. Collobert, "Deep learning via semi-supervised embedding," in *Neural Networks: Tricks of the Trade - Second Edition*, 2012, pp. 639–655.
[47] D. T. Wijaya, P. P. Talukdar, and T. M. Mitchell, "PIDGIN: ontology alignment using web text as interlingua," in *CIKM*, 2013, pp. 589–598.
[48] M. Ji, Y. Sun, M. Danilevsky, J. Han, and J. Gao, "Graph regularized transductive classification on heterogeneous information networks," in *ECML/PKDD*, 2010, pp. 570–586.
[49] T. Chen and Y. Sun, "Task-guided and path-augmented heterogeneous network embedding for author identification," in *WSDM*, 2017, pp. 295–304.
[50] Z. Yang, W. W. Cohen, and R. Salakhutdinov, "Revisiting semi-supervised learning with graph embeddings," in *ICML*, 2016, pp. 40–48.
[51] A. R. Barron, "Approximation and estimation bounds for artificial neural networks," *Machine Learning*, vol. 14, no. 1, pp. 115–133, 1994.
[52] J. B. Tenenbaum, V. De Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *science*, vol. 290, no. 5500, pp. 2319–2323, 2000.
[53] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *NIPS*, 2013, pp. 3111–3119.
[54] P. Li, H. Lu, N. Kanhabua, S. Zhao, and G. Pan, "Location inference for non-geotagged tweets in user timelines," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 6, pp. 1150–1165, 2019.
[55] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *ICML*, 2014, pp. 1764–1772.
[56] D. Kinga and J. B. Adam, "A method for stochastic optimization," in *ICLR*, 2015.
[57] I. Sutskever, J. Martens, G. E. Dahl, and G. E. Hinton, "On the importance of initialization and momentum in deep learning," in *ICML*, 2013, pp. 1139–1147.
[58] X. Shi, Z. Chen, H. Wang, D. Yeung, W. Wong, and W. Woo, "Convolutional LSTM network: A machine learning approach for precipitation nowcasting," in *NIPS*, 2015, pp. 802–810.
[59] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *JMLR*, vol. 9, no. Nov, pp. 2579–2605, 2008.
[60] N. Veldt, D. F. Gleich, and A. Wirth, "A correlation clustering framework for community detection," in *WWW*, 2018, pp. 439–448.



Pengfei Li is a PhD student in computer science with Zhejiang University, China. He received the BSc degree in computer science from Zhejiang University, in 2016. His research interests include machine learning and data mining.



Hua Lu is a professor with specific responsibilities in the Department of Computer Science, Aalborg University, Denmark. He received the BSc and MSc degrees from Peking University, China, and the PhD degree in computer science from National University of Singapore. His research interests include database and data management, geographic information systems, and mobile computing. He has served as PC cochair or vice chair for ISA 2011, MUE 2011, MDM 2012 and NDBC 2019, demo chair for SSDBM 2014, and PhD forum cochair for MDM 2016. He has served on the program committees for conferences such as VLDB, ICDE, KDD, CIKM, DASFAA, ACM SIGSPATIAL, SSTD, MDM and PAKDD. He is a senior member of the IEEE.

Qian Zheng received the B.E. and Ph.D. degrees in computer science from Zhejiang University, China, in 2011 and 2017, respectively. He visited Hong Kong Polytechnic University as a Research Associator from 2015 to 2016. He is currently a Research Fellow with the ROSE lab, Nanyang Polytechnic University. His current research interests include computational photography and biometric. He has published several papers in international journals and conference, such as IEEE-TPAMI, IEEE-TIFS, IEEE-TIP, ICCV, AAAI, WWW, and IJCAI. He has served as a program committee for CVPR 2018 Biometrics Workshop and is a reviewer of IEEE-TIP, IEEE-Access, and BMVC.

Shijian Li received the Ph.D. degree from Zhejiang University, Hangzhou, China, in 2006. In 2010, he was a Visiting Scholar with the Institute Telecom SudParis, Évry, France. He is currently with the College of Computer Science and Technology, Zhejiang University. He has published over 40 papers. His research interests include sensor networks, ubiquitous computing, and social computing. He serves as an Editor for the International Journal of Distributed Sensor Networks and as a reviewer or a PC member for more than ten conferences.

Gang Pan is a professor of the College of Computer Science and Technology, and vice-director of State Key Lab of CAD&CG, Zhejiang University. He obtained his B.S. and Ph.D. degrees both from Zhejiang University in 1998 and 2004 respectively. From 2007 to 2008, he was with the University of California, Los Angeles as a visiting scholar. His interests include artificial intelligence, pervasive computing, brain-machine interfaces, and brain-inspired computing. He has co-authored more than 100 refereed papers, and has 35 patents granted. Dr. Pan has received many technical awards, including IEEE TCSC Award for Excellence (Middle Career Researcher, 2018), CCF-IEEE CS Young Computer Scientist Award (2016), TOP-10 Achievements in Science and Technology in Chinese Universities (2016), National Science and Technology Progress Award (2015), Best Paper Award of ACM UbiComp'16, and serves as an associate editor of IEEE Trans. NNLS, IEEE Systems Journal, ACM Proceedings of Interactive, Mobile, Wearable and Ubiquitous Technologies (IMWUT), and Journal of Pervasive and Mobile Computing.