Revisiting User Mobility and Social Relationships in LBSNs: A Hypergraph Embedding Approach

Dingqi Yang, Bingqing Qu, Jie Yang and Philippe Cudre-Mauroux
University of Fribourg, Switzerland
{firstname.lastname}@unifr.ch

ABSTRACT

Location Based Social Networks (LBSNs) have been widely used as a primary data source to study the impact of mobility and social relationships on each other. Traditional approaches manually define features to characterize users' mobility homophily and social proximity, and show that mobility and social features can help friendship and location prediction tasks, respectively. However, these hand-crafted features not only require tedious human efforts, but also are difficult to generalize. In this paper, by revisiting user mobility and social relationships based on a large-scale LBSN dataset collected over a long-term period, we propose LBSN2Vec, a hypergraph embedding approach designed specifically for LBSN data for automatic feature learning. Specifically, LBSN data intrinsically forms a hypergraph including both user-user edges (friendships) and user-time-POI-semantic hyperedges (check-ins). Based on this hypergraph, we first propose a random-walk-with-stay scheme to jointly sample user check-ins and social relationships, and then learn node embeddings from the sampled (hyper)edges by preserving n-wise node proximity ($n = 2$ or 4). Our evaluation results show that LBSN2Vec both consistently and significantly outperforms the state-of-the-art graph embedding methods on both friendship and location prediction tasks, with an average improvement of 32.95% and 25.32%, respectively. Moreover, using LBSN2Vec, we discover the asymmetric impact of mobility and social relationships on predicting each other, which can serve as guidelines for future research on friendship and location prediction in LBSNs.

CCS CONCEPTS

• **Information systems** → *Data mining*; *Web applications*;

KEYWORDS

Mobility, Social relationship, Location based social network, Link prediction, Hypergraph, Embeddings

1 INTRODUCTION

Understanding the correlation between human mobility and social relationships is crucial for studying human dynamics, which is also a key ingredient for friendship and location prediction tasks. Location Based Social Networks (LBSNs), such as Foursquare, have attracted millions of users and generated a considerable amount of digital footprints from their daily life. As such, they have become a primary data source to study the impact of mobility and social relationships on each other. Specifically, in LBSNs, users can share their real-time presences with their friends by checking-in at a Point of Interest (POI), such as a restaurant or a gym. In addition to such fine-grained and semantic user mobility information, the social network of the corresponding users is also available.

Based on LBSN data, empirical analyses have shown various socio-spatial properties of user activities, in particular the correlation between user mobility characteristics (e.g., distance, co-location rate, etc.) and social networks [8, 26–28, 42]. Based on these findings, two main applications have been investigated, i.e., friendship prediction (a.k.a. link prediction) and location prediction. *Friendship prediction* aims at recommending social relationships that are likely to be established in the future [28], while *location prediction* tries to predict which POI a user will go to in a given context (e.g., at a given time) [8]. Existing work has shown that considering the correlation between user mobility and social relationships can improve the performance of both friendship prediction [7, 25, 28, 32, 42] and location prediction [8, 12, 18, 21, 25]. More precisely, these approaches usually select a set of hand-crafted features either from user mobility data or from the corresponding social network, and then show the impact of one on the other. For example, social features usually involve different network proximity metrics including common neighbor, Adamic-Adar [1], Katz index [17], etc. For mobility features, co-location rate is a widely used metric to measure the homophily between two users in terms of mobility traces; it has different variations such as normalized/unnormalized, weighted/unweighted, spatial only/spatiotemporal [28, 33]. However, such a manual feature engineering process not only requires tedious efforts from domain experts, but also shows less generalizability [13] (see Section 5.2.2 and 5.3.2 for more detail).

To overcome the limitation of hand-crafted features, automatic feature learning (a.k.a. representation learning) has been proposed [3]. When applied to networks or graphs, this paradigm is typically known as *graph (or network) embedding* [5], which represents nodes of a graph in a low-dimensional vector space while preserving key structural properties of the graph (e.g., topological proximity of the nodes). Based on such embeddings, graph analysis tasks (e.g., link prediction) can then be efficiently performed. As shown in Figure 1, LBSN data in this context intrinsically form a hypergraph consisting of four key data domains, i.e., spatial, temporal, semantic and
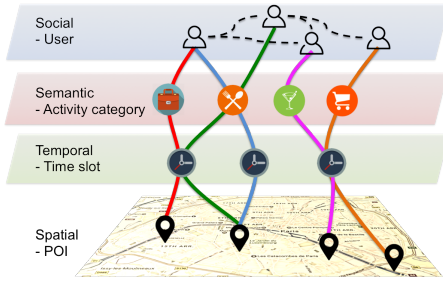
Figure 1: LBSN hypergraph containing both user mobility data (check-ins) and the corresponding social network. A friendship is represented by an edge (a black dotted line) linking two user nodes. A check-in is represented by a hyperedge (a colored thick line) linking four nodes, i.e., a user, an activity type, a time stamp and a POI.

social. This graph contains not only classical edges (i.e., friendships between two users in the social network), but also hyperedges (i.e., check-ins linking four nodes, one from each domain, representing a *user*'s presence at a *POI* at a specific *time* along with the *semantic* information about the user's activity there).

However, existing graph embedding techniques cannot fully grasp the complex data structure of LBSNs. First, most of the existing techniques were developed for classical graphs [6, 13, 22–24, 29, 30], where an edge links two nodes only; the node embeddings are then learnt such that the pairwise node proximity is preserved. However, preserving the pairwise node proximity cannot fully capture the information from the check-in hyperedges. Even though a hypergraph can be transformed into a classical graph by breaking each hyperedge into multiple classical edges, such an irreversible process causes a certain information loss, leading to degraded performance of the learnt node embeddings on different tasks (see Section 5.2 and 5.3 for more detail). Second, there are also a few techniques studying the hypergraph embedding problem, but they either focus on a n-uniform hypergraph (where all hyperedges contain a fixed number n of nodes) and thus capture only fixed-n-wise node proximity [2, 31, 44], or learn from hyperedges for heterogeneous events only (hyperedges linking nodes from different data domains) while ignoring edges within a data domain [14]. However, as shown in Figure 1, the LBSN hypergraph typically contains both classical friendship edges (n=2) within the user domain and check-in hyperedges (n=4) across all four data domains. Subsequently, these techniques cannot be directly applied to the full LBSN hypergraph. Even though these techniques could be applied on check-in hyperedges (n=4) only, ignoring social relationship indeed shows suboptimal performance for different tasks (see Section 5.3 for more detail).

Against this background, we revisit human mobility and social relationships in LBSNs in this paper and propose LBSN2Vec, a hypergraph embedding approach designed specifically for LBSN data with the special consideration of learning from *both* friendship edges *and* check-in hyperedges at the same time. Our contributions are three-fold:

- We collect a large-scale LBSN dataset over a long-term period and conduct a rigorous empirical analysis to systematically reveal the correlations between user check-ins and the corresponding social

network. Specifically, existing publicly available LBSN datasets collected from Foursquare [12], Gowalla [8] or Brightkite [8], usually contain check-in data from a set of users over a period of time, and one snapshot of the corresponding user social network. However, it is often unclear when the social network snapshot is collected (e.g., before, during or after the check-in data collection period); such chronological information is important in studying the impact of user mobility and the corresponding social network on each other. For example, to investigate the impact of social networks on user mobility patterns, only friendship formed before a check-in should be considered. Therefore, our collected dataset contains not only check-ins of a set of users over a two-year period, but also two snapshots of the corresponding user social network before and after the check-in data collection period, respectively. Based on this dataset, we conduct a rigorous empirical analysis of the impact of user mobility and social network on each other, showing that each of the four data domains has a clear impact on both friendship and location prediction tasks.

- We propose LBSN2Vec, a hypergraph embedding approach that can efficiently learn node embeddings from both friendship edges and check-in hyperedges in a LBSN hypergraph. To this end, we first propose a random-walk-with-stay scheme to jointly sample friendship and check-ins from a LBSN hypergraph. To balance the impact of social relationships and user mobility on the learnt node embeddings, we incorporate a tunable parameter to control the portion of social relationships and check-ins in the learning process in a probabilistic manner. Moreover, to learn node embeddings from hyperedges containing n nodes, we design LBSN2Vec to preserve the n-wise node proximity by simultaneously optimizing the proximity of the n nodes from a hyperedge. More precisely, we first compute the best-fit-line of the n nodes in cosine space, and then iteratively maximize the cosine similarity between each node of the hyperedge and the best-fit-line.

- We conduct a thorough evaluation using our collected dataset. Our results show that LBSN2Vec outperforms state-of-the-art graph embedding techniques on both friendship and location prediction tasks, with an average improvement of 32.95% and 25.32%, respectively. Moreover, using LBSN2Vec, we discover the asymmetric impact of user mobility and social networks on predicting each other. We find that node embeddings learnt from 80% social and 20% mobility data results in the best performance on the friendship prediction task, while those learnt from 40% mobility and 60% social data give the best performance on the location prediction task. Such an observation can serve as guidelines for future research on friendship and location prediction.

2 RELATED WORK

2.1 Human Mobility and Social Relationships

The interaction between human mobility and social relationships has been widely studied using different data sources over the past decade. In the earlier stage, call detail records have been often used to study human mobility, social ties and link prediction problems [8, 33]. Although these studies have revealed several interesting properties of human mobility and social ties, such data shows two obvious limitations. First, it contains only coarse-grained user location information on a cell tower level. Second, it does not contain

actual information about social relationships between users; the corresponding social network are usually built from the users' communication activities based on certain heuristics. For example, a friendship is assumed between a pair of users when there is at least one call between them [33] (or 10 calls in [8]).

The emergence of Location Based Social Networks provides a novel opportunity to collect both large-scale user mobility data (i.e., check-ins) and the corresponding social network. Using LBSN data, positive correlations between social proximity and mobility homophily have been universally found [8, 33]. However, existing publicly available LBSN datasets (Foursquare [12], Gowalla [8] or Brightkite [8]) mainly contain check-in data of a set of user over a period of time, and one snapshot of the corresponding social network without mentioning when the social network is collected[1]; such chronological information is important in studying the impact of user mobility and the corresponding social network on each other. Therefore, in this paper, we collected our own dataset from Foursquare, containing not only check-ins of a set of users over about two years, but also two snapshots of the corresponding user social network before and after the check-in data collection period, respectively. Based on this dataset, we investigate the impact of user check-ins and the corresponding social network on both friendship and location prediction tasks.

First, friendship prediction is a classical problem in social network analysis. It predicts the potential friendship between two users based on the existing social network [19]. Using LBSN data, friendship prediction approaches [7, 25, 28, 32, 42] often combine social proximity and mobility pattern similarity to achieve better prediction performance. For example, Scellato et al. [28] investigated a set of hand-crafted features and showed that besides social proximity, mobility pattern similarity is also a strong indicator to predict future friendship. Sadilek et al. [25] manually combined features from social proximity, mobility similarity, and textual similarity extracted from users' Tweets for friendship prediction.

Second, location prediction is a typical problem in human mobility modeling, which predicts the location of a user under a certain context based on the user's historical mobility traces. Using LBSN data, location prediction approaches incorporate social factors in mobility models, showing an improved prediction performance [8, 12, 18, 21, 25]. For example, Gao et al. [12] combined check-in patterns and social ties by considering mobility similarity between friends (co-location rate measured by cosine distance). Noulas et al. [21] investigated the next place prediction problem and found that besides a user's own check-in history, a "social filtering" feature serves as an important predictor for the user's next location.

However, the manual feature engineering process adopted by the existing work not only requires a lot of human efforts, but also shows less generalizability [13]. In this paper, we propose LBSN2Vec, a hypergraph embedding approach to automatically learn the node embeddings from a LBSN hypergraph, based on which both friendship and location prediction tasks can be efficiently performed. Moreover, our LBSN2Vec can also model the impact of user mobility and social relationships on each other.

2.2 Graph Embeddings

Most existing graph embedding approaches focus on preserving pairwise node proximity in a classical graph, which can be further classified into two categories according to the embedding learning process. First, *factorization based approaches* [6, 22, 24] measure pairwise node proximity as a matrix using a certain network proximity metric, such as common neighbor or Adamic-Adar, and then factorize this proximity matrix using matrix factorization techniques to learn the node embeddings. However, factorization-based approaches have an intrinsic scalability limitation, due to the quadratic complexity of matrix factorization algorithms [4]. Second, *graph-sampling based approaches* [13, 16, 23, 29, 30] sample node pairs (directly or via random walks) from a graph, and then design specific models to learn node embeddings from the sampled node pairs via stochastic optimization. These graph-sampling based approaches are able to scale up to large datasets, as their complexity mainly depends on the number of the sampled node pairs.

Moreover, there are also a few approaches studying the hypergraph embedding problem, but they either focus on a n-uniform hypergraph (where all hyperedges contain a fixed number n of nodes) and thus capture only fixed-n-wise node proximity [2, 31, 44], or learn from hyperedges for heterogeneous event only (hyperedges linking nodes from different data domains) while ignoring edges within a data domain [14]. However, in this paper, as shown in Figure 1, our LBSN hypergraph contains both classical friendship edges (n=2) within the user domain and check-in hyperedges (n=4) across all four data domains. Subsequently, these techniques cannot be directly applied to our full LBSN hypergraph. Even though they can be applied only on the check-in hyperedges (n=4) for example, ignoring social relationships indeed yields suboptimal performance (see Section 5.3 for more detail). Against this background, we propose LBSN2Vec to efficiently learn node embeddings from the LBSN hypergraph by preserving the n-wise node proximity encoded by the hyperedges (where $\{n \in \mathbb{Z}^+ | n \geq 2\}$).

We also note that there are several works exploited embedding techniques on LBSN data, but they mostly focus on check-in data only (without using social networks) for specific tasks, such as semantic annotation of POIs [34], POI recommendation [10, 35], next visitor prediction [11]. Only one recent work [43] used both social network and check-in data on LBSNs for learning embeddings for users and POIs, but it overlooks the temporal and semantic information of check-ins. Moreover, none of these works have investigated the correlations between human mobility and social relationships.

3 EMPIRICAL DATA ANALYSIS

In this section, we first present our dataset collection and its main characteristics, followed by empirical studies on the impact of mobility on the social network and vice versa.

3.1 Dataset Collection and Characteristics

Our collected Foursquare dataset contains a set of global-scale check-ins over about two years (from Apr. 2012 to Jan. 2014), and two snapshots of the corresponding user social network before (in Mar. 2012) and after (in May 2014) the check-in data collection period. The check-in data is collected from Foursquare, which has

---

[1]Note that one of the existing work [28] used multiple social network snapshots to study the friendship prediction problem, but the dataset is not publicly accessible. Moreover, it used hand-crafted features for friendship prediction.

been widely used as a key data source for studying location based services [39–41]. Specifically, as Foursquare user's check-ins can only be accessed from the user's social circle, they are not publicly available. However, many Foursquare users choose to also share their check-ins via Twitter [37]. Therefore, we collect global-scale check-in data by searching Foursquare-tagged Tweets from Twitter Public Streams for about two years (from Apr. 2012 to Jan. 2014). We keep only active users (who have performed at least one check-in per month). User social relationships are collected from Twitter, as Foursquare friendship information is not publicly available. Specifically, for each user, we first crawl her "followers" (users who follow this user) and "friends" (users who are followed by this user) on Twitter. We then build the corresponding user social network by keeping only reciprocal relationships [38], i.e., a friendship is assumed between two users if the two users follow each other. We collect two snapshots of such social network before (in Mar. 2012) and after (in May 2014) the check-in data collection period. After filtering out isolated users (i.e., users without social links), we keep only users appearing in both social network snapshots. In summary, our dataset contains 22,809,624 check-ins from 114,324 users on 3,820,891 POIs, and two snapshots of the corresponding user social network before (363,704 friendships) and after (607,333 friendships) the check-in data collection period. Our dataset is available here[2].

Figure 2 shows a series of visualizations of our dataset in all four data domains. For the social domain, Figure 2(a) shows the old social network with a color bar indicating user node degrees. We observe only a few nodes with high degrees, while most of the nodes are of low degrees, which suggests a heavy-tailed distribution. We do not show the new social network as its plot shows a very similar visual structure as the old one. For the semantic domain, Figure 2(b) shows the categorical distribution of check-in POIs, where we have a total number of 453 POI categories (i.e., activity categories). We find that daily-routine related activities including "Shopping Mall", "Home", "Train station" and "Office", are among the most frequent categories. For the temporal domain, Figure 2(c) plots the weekly temporal traffic pattern of check-ins over 168 hours in a week. We observe not only a clear daily periodicity, but also the difference between weekday (with three peaks in the morning, afternoon and evening, respectively) and weekend (with only one flat peak during the daytime). Based on this observation, we define the time granularity (time slot) as 168 hours (corresponding to a full week) for this study. For the spatial domain, we visualize the GPS coordinates of POIs on a world map in Figure 2(d). We observe POIs from our dataset spread all over the world, with urban areas concentrating many of the POIs.

In addition, we take a closer look at the dataset from a statistical point of view. Figure 3 shows the Complementary Cumulative Distribution Function (CCDF) of different statistics of the collected dataset. First, we observe a heavy-tailed distribution of all statistics, including user node degrees in Figure 3(a), number of check-ins per user in Figure 3(b), number of check-ins per POI in Figure 3(c), and check-in distance from home location [8] in Figure 3(d), which implies a large variance in these statistics. For example, the heavy-tailed distribution of check-in distance from home shows that 70% of the check-ins happen less than 100km from home. Moreover, we
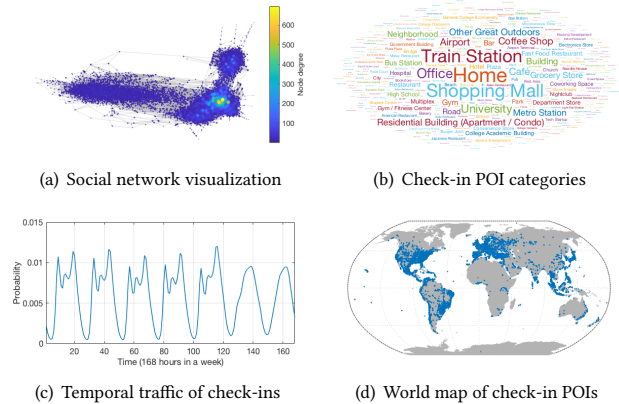
(a) Social network visualization

(b) Check-in POI categories

(c) Temporal traffic of check-ins

(d) World map of check-in POIs

Figure 2: Visualization of the collected LBSN data

(a) Node (user) degree

(b) Number of check-ins per user

(c) Number of check-ins per POI
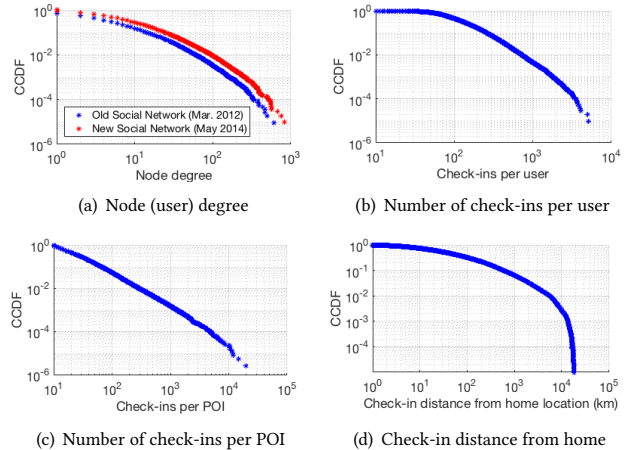
(d) Check-in distance from home

Figure 3: Complementary Cumulative Distribution Function (CCDF) of various statistics in the collected dataset

observe that the user social network significantly expanded over the two years in Figure 3(a); the average degree increases from 3.38 to 5.56. In the following, we study the impact of mobility on the corresponding social network and vice versa.

3.2 Impact of Mobility on New Friendships

We investigate the probability of establishing a *new* friendship (appearing in the new but not in the old social network), w.r.t. various social and mobility factors in the old social network and user check-in data. Figure 4 shows the results. On one hand, the existing social network is unsurprisingly the primary predictor of future friendships [19]. Figure 4(a) shows the impact of the social proximity between users in the old social network. We observe that pairs of users at closer social distance are more likely to establish a new friendship. On the other hand, we use the following three metrics to measure the similarity of user mobility patterns on spatial, temporal and semantic domains, respectively.

- **Co-location** [33]: By modeling each user's check-in data as a vector, where each element in the vector indicates whether the user has checked-in at the corresponding POI or not, we measure the Jaccard distance between two users' check-in vectors.
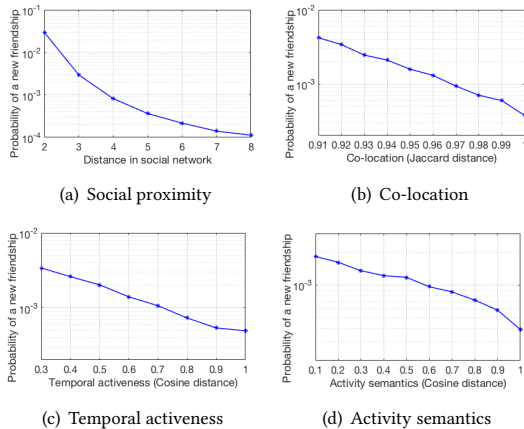
(a) Social proximity

(b) Co-location

(c) Temporal activeness

(d) Activity semantics

Figure 4: Impact of social and mobility factors on establishing new friendships

- **Temporal activeness** [36]: We model a user's check-in data by a vector encoding the user's check-in frequency over 168 hours in a week (as shown in Figure 2(c)), and then measure the Cosine distance between two users' temporal activeness vectors.
- **Activity semantics** [40]: We model a user's check-in data by a vector encoding the categorical distribution of a user's checked POIs (i.e., the distribution on POI categories as shown in Figure 2(b)), and then measure the Cosine distance between two users' activity semantic vectors.

Figure 4(b), 4(c) and 4(d) show the impact of the above three metrics on the probability of establishing a new friendship, respectively. We consistently observe that pairs of users with similar mobility patterns (measured in any of the three domains) are more likely to establish a new friendship in the future. *These observations suggest that all four data domains should be considered in the friendship prediction task. Note that we only use these metrics to investigate which data domains should be considered in the LBSN2Vec, yet none of these metric is required in our node embedding learning process.*

3.3 Impact of Friendship on Mobility

We investigate the similarity of user mobility patterns w.r.t. the social proximity between users in the *old* social network. Figure 5 shows the results. We consistently observe that pairs of users at closer social proximity are more likely to exhibit similar mobility patterns measured by all the three metrics, i.e., co-location, temporal activeness and activity semantics. Such an observation suggests that social relationships can be used to help predict user mobility.

In addition, we also investigate the impact of a user's historical mobility traces on her future mobility. To achieve this goal, we chronologically split our check-in data (of about two years) into two parts of the same duration, i.e., a first-year check-in dataset and a second-year check-in dataset. Subsequently, we compute the similarity between each user's mobility patterns in the first and the second year, measured by the aforementioned three metrics. The average similarity is 0.12, 0.32 and 0.38 for co-location, temporal activeness and activity semantics, respectively. Compared to the mobility similarity between friends (the distance in the old social network is 1 in Figure 5, i.e., 0.04, 0.16 and 0.29 for co-location,
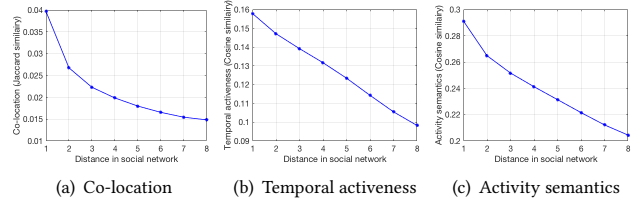

(a) Co-location

(b) Temporal activeness

(c) Activity semantics

Figure 5: Impact of social proximity on mobility factors

temporal activeness and activity semantics, respectively), we find that a user's own mobility history is indeed the primary predictor of her future mobility. *Subsequently, we also consider all four data domains in our LBSN2Vec for node embedding learning in the location prediction task. We note again that as an automatic feature learning approach, LBSN2Vec does not involve any of the metrics defined in this section.*

4 LBSN2VEC

In this paper, we adopt a graph-sampling based embedding paradigm due to its intrinsic scalability advantage as discussed in Section 2.2. Specifically, our method first uses a novel random-walk-with-stay scheme to jointly sample friendships and check-in hyperedges[3] from the LBSN hypergraph, and then learn node embeddings from these hyperedges to preserve the n-wise node proximity by optimizing the proximity of the n nodes from a hyperedge simultaneously.

4.1 Random Walk with Stay

As shown in Figure 1, the LBSN hypergraph consists of four data domains, i.e., spatial, temporal, semantic and social domain, and two types of edges, i.e., classical edges (friendships) linking pairs of user nodes and hyperedges (check-ins) linking four nodes, one from each domain. To sample (hyper)edges from this graph, we propose a random-walk-with-stay scheme to jointly sample friendship and check-in hyperedges. As shown in Figure 6, our random-walk-with-stay scheme performs classical random walk only on user nodes based on their friendships in social domain, while for each encountered user node, it stays on the user node to sample a set of hyperedges (check-ins) from the corresponding user. Subsequently, in the node embedding learning process, when iterating over each user node in a random walk sequence, we not only learn from two user nodes appearing with a context window of length k, but also stay on the corresponding user node to learn from its check-in hyperedges. In other words, the node embedding learning process alternates between these two types of edges (i.e., friendships and check-ins). To perform classical random walk on the user social network, we use the same strategy as used by existing works [13, 23], generating r walks of length l rooted on each user node.

Moreover, in order to balance the impact of friendships and check-ins on the learnt node embeddings, LBSN2Vec incorporates a tunable parameter α to control the portion of learning edges of each type. Specifically, for a given context window of length k, we iterate over $2k$ contexts for each user node v_i (k context nodes on each side), resulting in $2k$ pairs of user nodes. Figure 6 shows an example

---

[3] A classical edge linking two user nodes (i.e., a friendship) can be regarded as a special case of a hyperedge consisting of only two nodes. In the following, besides the check-in hyperedges, we also use the term "hyperedge" to describe a pair of user nodes.
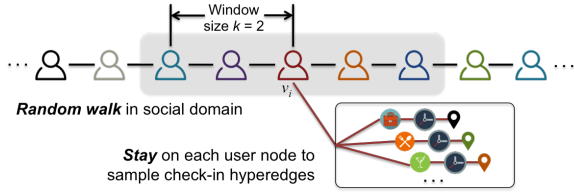
Figure 6: Random walk with stay on the LBSN hypergraph

(a) Euclidean space with perpendic-ular distance

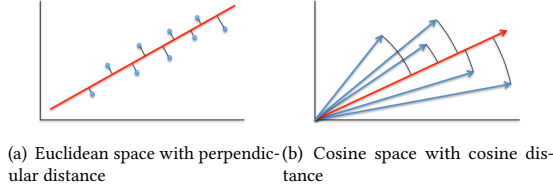(b) Cosine space with cosine distance

Figure 7: Examples of the best-fit-lines in Euclidean and Cosine spaces. The given data points are represented in blue, while the corresponding best-fit-line is shown in red.

where the context window size is 2. Subsequently, we sample for each user node the same number ($2k$) of check-ins. Afterwards, we learn from each check-in with a probability α, while from each user node pair with a probability $(1 - \alpha)$, where α and $(1 - \alpha)$ actually specify the impact of mobility and social network on the learnt node embeddings, respectively. In summary, for each node v_i in a random walk, the expected total number of learnt edges is $2k$, including an expected number $2k\alpha$ of check-ins and an expected number of $2k(1-\alpha)$ of user node pairs. A small value of α gives less importance on the check-in data and more on social network, and vice versa. In the following, we present our node embedding learning process preserving the n-wise node proximity for hyperedges.

4.2 Learning from Hyperedges

We design a novel embedding model in LBSN2Vec to preserve the n-wise node proximity by maximizing the similarity between the nodes of a hypergraph and their best-fit-line under cosine similarity. Specifically, our embedding model needs to learn from hyperedges linking either 2 or 4 nodes. We further generalize this problem to hyperedges containing n nodes, where $\{n \in \mathbb{Z}^+ | n \geq 2\}$, and design our model that can preserve the n-wise node proximity from those hyperedges. As the hyperedges are actually sampled and learnt one by one from a random walk, we focus in the following on the learning process for one hyperedge.

To learn node embeddings from a hyperedge containing n nodes, we borrow the idea of best-fit-line (a.k.a. line of best fit) from linear regression [15]. In general, a best-fit-line is a straight line that is the best approximation of the given data points. Figure 7(a) illustrates an example of the best-fit-line in Euclidean space that minimizes the sum of perpendicular distances, which can be computed via linear least squares. In this paper, following existing graph embedding techniques [6, 13, 22–24, 29, 30], we learn node embeddings in cosine space where the proximity between nodes are measured using cosine similarity (or dot product of the normalized embedding vectors). Moreover, we show below that the computation of the best-fit-line in cosine space can be significantly simplified. Formally, the best-fit-line of a set of node embeddings is the vector that minimizes

the sum of cosine distances between each node embedding and the best-fit-line, as shown in Figure 7(b), which can be efficiently computed using Proposition 1:

PROPOSITION 1. *For a set of nodes (in a hyperedge) $\{v_i | i = 1, 2, \ldots, n\}$, the corresponding best-fit-line can be computed as $\vec{v}_b = \sum_{i=1}^{n} \frac{\vec{v}_i}{\|\vec{v}_i\|}$, where $\vec{v}_i$ refer to the embedding vector of node v_i.*

PROOF. The problem of finding the best-fit-line $\vec{v}_b$ that minimizes the sum of cosine distances can be formulated as follows:

$$\operatorname*{argmin}_{\vec{v}_b} \sum_{i=1}^{n} (1 - \cos(\vec{v}_i, \vec{v}_b)) \tag{1}$$

As minimizing the cosine distance is equivalent to maximizing the cosine similarity, we obtain:

$$\operatorname*{argmax}_{\vec{v}_b} \sum_{i=1}^{n} \cos(\vec{v}_i, \vec{v}_b) = \operatorname*{argmax}_{\vec{v}_b} (\sum_{i=1}^{n} \frac{\vec{v}_i}{\|\vec{v}_i\|}) \cdot \frac{\vec{v}_b}{\|\vec{v}_b\|} \tag{2}$$

Let $\vec{v}_a = \sum_{i=1}^{n} \frac{\vec{v}_i}{\|\vec{v}_i\|}$, we obtain:

$$\operatorname*{argmax}_{\vec{v}_b} \vec{v}_a \cdot \frac{\vec{v}_b}{\|\vec{v}_b\|} = \operatorname*{argmax}_{\vec{v}_b} \|\vec{v}_a\| \frac{\vec{v}_a \cdot \vec{v}_b}{\|\vec{v}_a\|\|\vec{v}_b\|} \tag{3}$$

where $\|\vec{v}_a\|$ is a constant for the given set of node. Subsequently, the above function actually maximizes the cosine similarity between $\vec{v}_a$ and $\vec{v}_b$. As the maximal cosine similarity between any two vectors is achieved if and only if the two vectors have the same orientation, the best-fit-line $\vec{v}_b$ can be simply computed as:

$$\vec{v}_b = \vec{v}_a = \sum_{i=1}^{n} \frac{\vec{v}_i}{\|\vec{v}_i\|} \tag{4}$$

This completes the proof. □

Therefore, we optimize the n-wise node proximity by iteratively maximizing the cosine similarity between each node embedding $\vec{v}_i$ from the hyperedge $\{v_i | i = 1, 2, \ldots, n\}$ and the best-fit-line $\vec{v}_b$.

$$\Theta = \sum_{i=1}^{n} \cos(\vec{v}_i, \vec{v}_b) \tag{5}$$

Note that here we keep using cosine similarity rather than dot product (which is widely used by existing techniques), because cosine similarity is not affected by the norm of the input vectors, in particular when the norm of the best-fit-line $\vec{v}_b$ computed by Eq. 4 has a large variance due to the variant number of the nodes n in different hyperedges ($n = 2$ or 4 in our LBSN hypergraph).

In addition, similar to other graph-sampling based embedding techniques [13, 23, 29, 30], we also adopt a negative sampling technique to maximize the cosine distance between each negative sample node v_N and the best-fit-line vector $\vec{v}_b$. Finally, we want to maximize the following objective function for one hyperedge $\{v_i | i = 1, 2, \ldots, n\}$:

$$\Theta = \sum_{i=1}^{n} \left(\cos(\vec{v}_i, \vec{v}_b) + \gamma \cdot \mathbb{E}_{v_N} [1 - \cos(\vec{v}_N \cdot \vec{v}_b)] \right) \tag{6}$$

where $\gamma \in \mathbb{Z}^+$ is the number of negative samples. Note that as the LBSN hypergraph contains four data domains, the negative samples for a node v_i from one data domain are uniformly sampled from the nodes in the same data domain.

Table 1: Statistics of the selected datasets

Dataset	NYC	TKY	IST	JK	KL	SP
#User	4,024	7,232	10,367	6,395	6,432	3,954
#POI	3,628	10,856	12,693	8,826	10,817	6,286
#Check-ins	105,961	699,324	908,162	378,559	526,405	249,839
#Friendships(Before)	8,723	37,480	21,354	11,207	16,161	9,655
#Friendships(After)	10,545	51,704	36,007	16,950	31,178	14,402

The above objective function can be optimized using stochastic gradient descent techniques. Note that we only need to update $\vec{v}_i$ and $\vec{v}_N$ when learning from one specific hyperedge, as the best-fit-line vector $\vec{v}_b$ is fixed. The gradient of the above objective function with respect to $\vec{v}_i$ and $\vec{v}_N$ is computed as follows:

$$\frac{\partial \Theta}{\partial \vec{v}_i} = \frac{\vec{v}_b}{\|\vec{v}_i\|\|\vec{v}_b\|} - \frac{\vec{v}_i \cos(\vec{v}_i, \vec{v}_b)}{\|\vec{v}_i\|^2} \tag{7}$$

$$\frac{\partial \Theta}{\partial \vec{v}_N} = -\frac{\vec{v}_b}{\|\vec{v}_N\|\|\vec{v}_b\|} + \frac{\vec{v}_N \cos(\vec{v}_N, \vec{v}_b)}{\|\vec{v}_N\|^2} \tag{8}$$

In summary, to learn node embeddings from the LBSN hypergraph, we first sample a set of friendship and check-in hyperedges using our proposed random-walk-with-stay scheme, and then iteratively learn from each hyperedge $\{v_i | i = 1, 2, \ldots, n\}$ ($n = 2$ for friendships or $n = 4$ for check-ins) by optimizing the objective function in Eq. 6.

5 EXPERIMENTS

In this section, we evaluate LBSN2Vec on both friendship and location prediction tasks using our collected dataset. In the following, we first present our experimental setup, and then discuss the results on individual tasks, followed by a parameter sensitivity study.

5.1 Experimental Setup

5.1.1 Dataset. In this study, we investigate the friendship and location prediction at an urban scale. Without loss of generality, we select six cities with a large number of check-ins, while also considering the cultural diversity of the selected cities: New York City (**NYC**), Tokyo (**TKY**), Istanbul (**IST**), Jakarta (**JK**), Kuala Lumpur (**KL**), Sao Paulo (**SP**). We select users in the largest connected components of the old social network. Table 1 summarizes the statistics of the selected datasets.

5.1.2 Baselines. As our objective is automatic feature learning from LBSN data, we compare our method to the following state-of-the-art graph embedding methods.

- **DeepWalk** [23] first generates node sequences using random walks, and then feeds them to a SkipGram model to output node embeddings. We set the walk length to $l = 80$, the number of walks per node to $r = 10$, and the context window size for the SkipGram model to $k = 10$.
- **Node2vec** [13] extends DeepWalk by introducing a parameterized random walk method to balance the breadth-first search (return parameter p) and depth-first search (in-out parameter q) strategies, to capture richer graph structures. Following the suggestions in the original paper, we tune p and q with a grid search over $p, q \in \{0.25, 0.05, 1, 2, 4\}$. We keep the other parameters l, r, k similar as for DeepWalk.

- **LINE** [29] explicitly defines 1st and 2nd-order node proximity to learn graph embeddings. Specifically, to learn node embeddings of dimension d, it first learns two sets of $d/2$-dimension node embeddings preserving 1st and 2nd-order node proximity, respectively, and then concatenates them together.
- **HEBE** [14] learns node embeddings for *heterogeneous event data* using tensor modeling, where a heterogeneous hyperedge links nodes across different domains only and is modeled as an entry in an high-order tensor.
- **DHNE** [31] uses a deep autoencoder to learning node embeddings from a *n-uniform heterogeneous hypergraph*. It preserves both fixed-n-wise node proximity encoded by the hyperedges, and also the high-order node proximity defined as the similarity between nodes' neighborhood structures.

As the baseline methods cannot directly take the LBSN hypergraph as input graph, we propose three settings to adapt our LBSN hypergraph:

- **(S)**: Only the <u>S</u>ocial network is considered by keeping nodes in user domain and their friendship edges. This setting can be applied to classical graph embedding techniques (preserving pairwise node proximity), i.e., **DeepWalk**, **Node2vec** and **LINE**.
- **(M)**: Only the user <u>M</u>obility is considered by keeping check-in hyperedges only. This setting can be directly applied to **HEBE** and **DHNE** which can take heterogeneous check-in hyperedges as input. In addition, we also apply this setting to **DeepWalk**, **Node2vec** and **LINE** by breaking each check-in hyperedge into multiple classical edges linking each pair of nodes in the hyperedge (e.g., user-time, time-POI, etc.).
- **(S&M)**: The full LBSN hypergraph containing both friendship edges and check-in hyperedges. This setting can be directly applied to our **LBSN2Vec**. In addition, we also apply this setting to **DeepWalk**, **Node2vec** and **LINE** by mapping the full LBSN hypergraph onto a classical graph by keeping friendship edges and also breaking each check-in hyperedge into multiple classical edges linking each pair of nodes in the hyperedge (same as for the M setting).

For our method LBSN2Vec, we keep the same random walk parameters as for DeepWalk and Node2vec ($l = 80, r = 10, k = 10$), and tune the parameter α within $[0, 1]$ with a step of 0.1 to balance the impact of social and mobility on the learnt embeddings (more discussion on this point in Section 5.4). The dimension of the node embeddings d is set to 128 and the number of negative samples γ is set to 10 for all methods in all experiments, if not specified otherwise. The implementation of LBSN2Vec is available here[4].

5.2 Friendship (link) Prediction

Friendship prediction recommends friendships that will probably be established in the future. In this study, we advocate an unsupervised friendship prediction approach [19, 20] which generates a ranking list of potential links between pairs of users. Specifically, after learning the node embeddings based on the *old* social network and check-ins, we rank pairs of user nodes (not being friends in the old social network) according to the cosine similarity between their embeddings. We then evaluate the obtained ranking list against

---

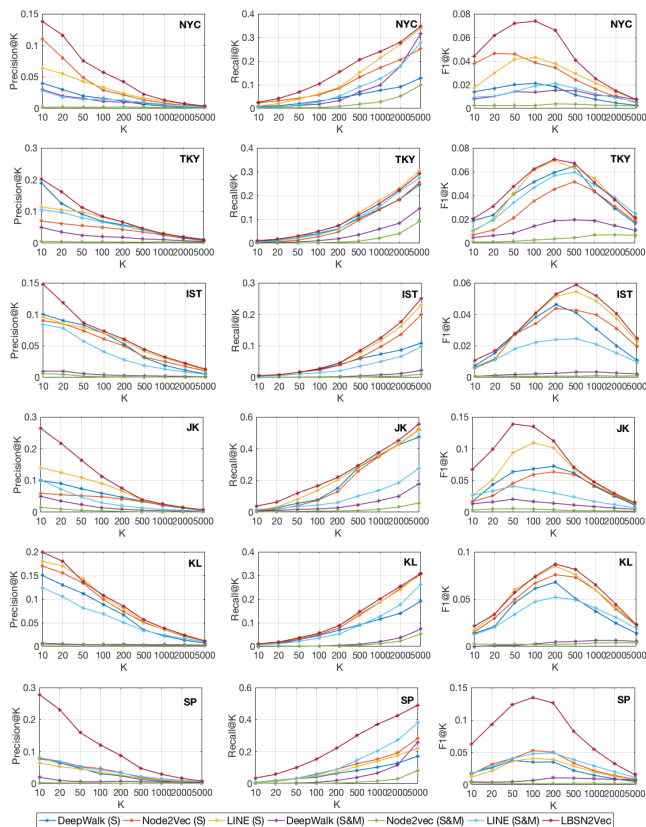[4]https://github.com/eXascaleInfolab/LBSN2Vec

Figure 8: Friendship prediction performance comparison with other graph embedding methods. Each row shows the results on one dataset, while each column shows the results for one evaluation metric.

Figure 9: Friendship prediction performance comparison with hand-crafted features.

the new friendships (appearing in the new but not in the old social network), using three common metrics for evaluating ranked results, i.e., precision, recall, and F1-score on the top K predicted friendships [19]. As the number of candidate pairs of nodes is too large, we randomly sample 1% pairs of nodes for the evaluation, and report the average results from 10 repeated trials.

5.2.1 Comparison with other graph embedding methods. We report only the methods including social networks (setting S and S&M), since the social proximity in the old social network is indeed the primary predictor of new friendships (see Figure 4(a) for more details) and since we found out that methods using setting M perform very poorly on this task. Figure 8 shows the results. We clearly observe that LBSN2Vec achieves the highest precision, recall and F1-score in general. For example, we find that LBSN2Vec outperforms all baselines with an average improvement of 32.95% on precision@10 over the best baselines across different datasets.

One interesting observation is that for all the baseline methods, considering both the social network and mobility (setting S&M) leads to worse performance than considering only the social network (setting S). In other words, for all baseline methods, adding user mobility information decreases the friendship prediction performance. Despite the counter-intuitive nature of this observation,
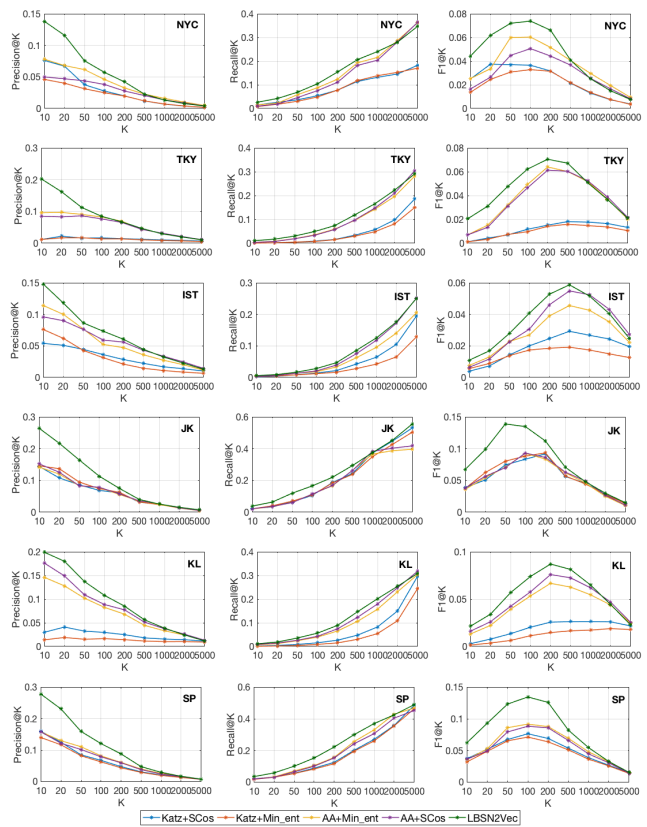
we notice that these methods fail to balance the impact of social network and user mobility on the learnt node embeddings. More precisely, as the number of check-ins is usually much larger than the number of friendships, the S&M graph is dominated by the edges representing check-ins. When the baseline methods uniformly sample edges from the S&M graph to learn the node embeddings, the sampled edges are also dominated by the edges representing check-ins, which naturally imposes a strong impact of mobility on the learnt node embeddings. In fact, as mentioned for Figure 4(a) in Section 3.2, the social proximity in the old social network is indeed the primary predictor of new friendships (our parameter sensitivity study in Section 5.4 also verifies this point below). Therefore, the learnt node embeddings from those baseline methods (with setting M) result in the worst performance in friendship prediction.

5.2.2 Comparison with hand-crafted features. We compare LBSN2Vec with the following hand-crafted features suggested specifically for the friendship prediction task by previous work:

- *Katz Index (**Katz**)* [17] is the best performing social feature suggested by [33]. It is defined as the weighted sum of all paths between two users on the social networks, where the weight of a path decays exponentially with its length. $Katz(u, v) = \sum_{l=1}^{\infty} \beta^l \cdot |path_{u,v}^l|$, where $path_{u,v}^l$ is the set of all paths with length l from u to v. We set $\beta = 0.05$ as suggested by [33].
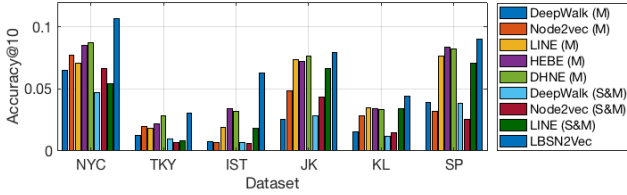
Figure 10: Location prediction performance comparison with other graph embedding methods on all test check-ins

Figure 11: Location prediction performance comparison with other graph embedding methods on new POIs.

- *Adamic-Adar (AA)* [1] is the best performing social feature suggested by [28]. It is the weighted sum of the common neighbors between two users. The weight for each common neighbor is the inverse logarithm of its degree. $AA(u,v) = \sum_{z \in \Gamma(u) \cap \Gamma(v)} \frac{1}{\log |\Gamma(z)|}$, where $\Gamma(u)$ is the set of neighbors (friends) of user u.
- *Spatial Cosine Similarity (SCos)* is the best performing mobility feature suggested by [33]. It is defined as the cosine similarity between two users' check-in vectors C_u and C_v, i.e., $\cos(C_u, C_v)$, where each element $C_u(i)$ is u's check-in count on POI i.
- *Minimum place entropy (Min_ent)* is the best performing mobility feature suggested by [28]. It is defined as the minimum POI entropy (i.e., the entropy of the empirical distribution of check-ins at that POI over users) that two users share.

As combining social and mobility features is suggested by both [33] and [28], we consider all the combinations of the above social and mobility features, including **Katz+SCos**, **Katz+Min_ent**, **AA+Min_ent** and **AA+SCos**. Specifically, as these features have different value ranges, we first generate one ranking list of predicted friendships using each feature, and then merge two ranking lists using reciprocal rank fusion [9], which is a popular rank fusion method widely used in information retrieval research. Figure 9 shows the results. We clearly observe that LBSN2Vec outperforms the hand-crafted feature based methods in general, and we find an average improvement of 66.44% on precision@10 over the best baselines across different datasets. Moreover, We observe that none of the four combinations can consistently outperform others over all datasets, which shows the generalization limitations of the hand crafted features across different datasets.

5.3 Location Prediction

Location prediction tries to predict the POI a user will be located in at a given time slot. To implement this task, we chronologically split our check-in data into two parts, i.e., the first 80% for training and the last 20% for testing. We then learn node embeddings from the old social network and the training check-in data. Based on the learnt node embeddings, for each test check-in (containing a user, a time slot and a POI), we rank all POIs according to the sum of two cosine similarities: one between the node embeddings of the user and a candidate POI, and one between the node embeddings of the time slot and the candidate POI. For DHNE, we use its own similarity function for ranking POIs. Subsequently, we evaluate the obtained POI ranking list against the actual POI in the test check-in. We report the average accuracy@10 over all test check-ins.

5.3.1 Comparison with other graph embedding methods. Figure 10 shows the results. Note that only settings involving check-in data (setting M and S&M) are eligible for this task. First, we observe
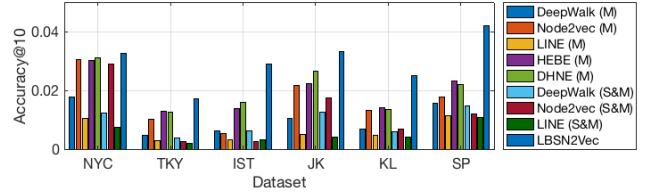
that our LBSN2Vec consistently outperforms all baselines with an average improvement of 25.32% on accuracy@10 over the best baselines across different datasets. Second, we see that using setting M, hypergraph embedding techniques (HEBE and DHNE) achieve better performance than classical graph embedding techniques (preserving only pairwise node proximity) in general, as the irreversible process of transforming a hypergraph into a classical graph causes a certain information loss, leading to degraded performance for the location prediction task. Finally, similar to friendship prediction, we also find that for all baselines considering both the social network and mobility (setting S&M) results in worse performance than considering mobility (M) only, as the baseline methods fail to balance the impact of the social network and user mobility on the learnt node embeddings. However, the performance drop is relatively smaller than that of the friendship prediction task, as the S&M graph is actually dominated by the edges from the check-ins.

In addition, we also investigate the prediction performance on new POIs only, i.e., the POIs appearing in the test check-ins but not in the training check-ins. As historical (training) check-ins of a user are represented as hyperedges in the LBSN hypergraph, check-ins with new POIs actually refer to the non-observed hyperedges that will be established in the future. Therefore, this task is more difficult as it requires to explore the LBSN hypergraph structure for predicting potential hyperedges, rather than preserving the observed hypergraph structure. Figure 11 shows the accuracy@10 for only new POIs. We observe that compared to the case of all testing POIs, the improvement of our method over the baselines are much higher for new POIs. We observe an average improvement of 50.17% on accuracy@10 in the case of only new POIs, compared to an average improvement of 25.32% in the case of all test POIs. Such an observation demonstrates the strong advantage of LBSN2Vec in predicting potential check-in hyperedges.

5.3.2 Comparison with hand-crafted features. We also compare LBSN2Vec with the following hand-crafted features suggested specifically for the location prediction task by previous work:

- *Most Frequent POI (MFP)* is the best performing mobility feature suggested by [21]. For each user, it ranks a POI according to the number of the user's check-ins at that POI in the training dataset.
- *Most Frequent Time (MFT)* is suggested by [40]. For each user and each time slot, it ranks a POI according to the number of user's check-ins at that POI and at that time slot in the training dataset.
- *Social Filtering (SF)* is suggested by [21]. For a user u and his friends $\Gamma(u)$, it ranks a POI according to the total number of check-ins that any friend v ($v \in \Gamma(u)$) of the user has performed at the POI.
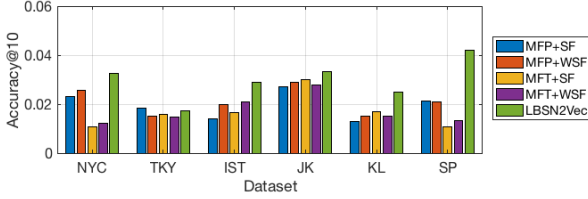
Figure 12: Location prediction performance comparison with hand-craft features on new POIs.

- *Weighted Social Filtering (WSF)* is suggested by [12]. It is a weighted version of SF, where each friend's check-in count is weighed by the mobility similarity between the user and that friend. It ranks a POI according to the weighted sum of the number of check-ins that any her friend v, $v \in \Gamma(u)$, has performed at that POI.

Similar to the friendship prediction task, we combine the results from mobility and social features using reciprocal rank fusion [9], resulting in four combinations (i.e., **MFP+SF**, **MFP+WSF**, **MFT+SF**, **MFT+WSF**). Figure 12 shows the results on new POIs. We observe that our method outperforms the hand-crafted feature in most cases (expect for the TKY dataset where MFP+SF is slightly better). In addition, similar to the case of friendship prediction, none of the four combinations consistently achieve higher accuracy than others, showing the generalization limitations of hand-crafted features.

5.4 Parameter Sensitivity Study

5.4.1 Balance trade-off between social and mobility with α. We investigate the trade-off between the social network and mobility patterns on both the friendship and location prediction tasks by varying α within $[0, 1]$ with a step of 0.1. A small value of α gives more importance on the social network and less on check-in data when learning node embeddings, and vice versa. Figure 13 shows the results. We see a clear trade-off between the social network and mobility on all datasets. On one hand, when increasing α from 0, the friendship prediction performance slightly increases and reaches its peak around $\alpha = 0.2$, as considering mobility factors indeed helps the friendship prediction task. When further increasing α, the performance start to decrease, and we observe a sharp drop around $\alpha = 0.5$. On the other hand, when decreasing α from 1, the location prediction performance slightly increases and reaches its peak around $\alpha = 0.6$ in most cases (expect for KL dataset), meaning that considering social factors can also help the location prediction task. When further decreasing α, the performance start to decrease, but we observe a sharp drop around $\alpha = 0.1$ or 0.2. Interestingly, by comparing the values of α giving the peak performance in the two tasks, we find the asymmetric impact of social and mobility factor on each other, which seems to be universal across different datasets. More precisely, combining 80% social with 20% mobility data results in the best performance for the friendship prediction task, while combining 40% mobility with 60% social data gives the best performance for the location prediction task.

5.4.2 Dimension of the node embeddings d. We study the impact of the node embedding dimension d on both the friendship and location prediction tasks, as well as on the node embedding learning time on a commodity PC (Intel Core i7-4770HQ@2.20GHz, 16GB RAM, Mac OS X). Figure 14 shows the results on the NYC dataset.
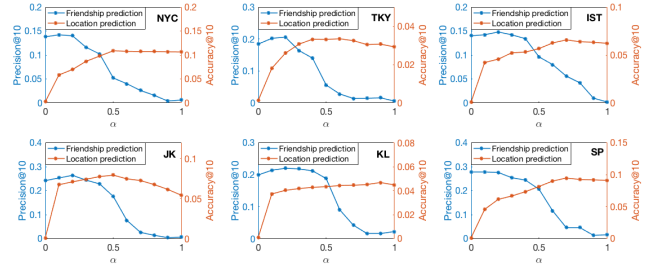

Figure 13: Impact of α on friendship and location prediction tasks
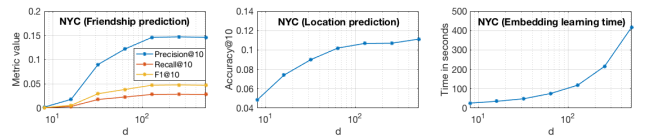

Figure 14: Impact of the node embedding dimension d on the NYC dataset

First, we see that a small d leads to poor performance on both tasks, as the resulting node embeddings cannot capture sufficient information from the LBSN hypergraph. Then, the performance increases with increasing d, and converges around $d = 128$, meaning that further increasing d leads to little performance improvement. In contrast, the embedding learning time always increases with increasing d, as a higher dimension of node embedding requires more computation in the node embedding learning process. Therefore, we select $d = 128$ for all previous experiments.

6 CONCLUSION

In this paper, by revisiting user mobility and social relationships in LBSNs, we propose LBSN2Vec, a hypergraph embedding approach designed specifically for LBSN data. First, by collecting a large-scale LBSN dataset over a long-term period, we conducted a rigorous empirical analysis to reveal the correlations between user check-ins and the corresponding social network. Then, we designed LBSN2Vec to learn node embeddings from the LBSN hypergraph. It performs random-walk-with-stay to jointly sample user mobility patterns and social relationships from the LBSN hypergraph, and then learns node embeddings from the sampled hyperedges by preserving the n-wise node proximity captured by a hyperedge containing n nodes. Using the collected LBSN data, our extensive evaluation shows that LBSN2Vec consistently outperforms the state-of-the-art graph embedding methods on both the friendship and location prediction tasks, with an average improvement of 32.95% and 25.32%, respectively. Moreover, using LBSN2Vec, we discover the asymmetric impact of mobility and social relationships on predicting each other, which can serve as guidelines for future research on friendship and location prediction in LBSNs.

In the future, we plan to extend LBSN2Vec by further considering sequential patterns in the user check-in data.

REFERENCES

[1] Lada A Adamic and Eytan Adar. 2003. Friends and neighbors on the web. *Social networks* 25, 3 (2003), 211–230.

[2] Amin Bahmanian and Mike Newman. 2016. Embedding Factorizations for 3-Uniform Hypergraphs II: r-Factorizations into s-Factorizations. *The Electronic Journal of Combinatorics* 23, 2 (2016), 2–42.

[3] Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence* 35, 8 (2013), 1798–1828.

[4] Matthew Brand. 2003. Fast online svd revisions for lightweight recommender systems. In *SDM*. SIAM, 37–46.

[5] Hongyun Cai, Vincent W Zheng, and Kevin Chang. 2018. A comprehensive survey of graph embedding: problems, techniques and applications. *IEEE Transactions on Knowledge and Data Engineering* (2018).

[6] Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2015. Grarep: Learning graph representations with global structural information. In *CIKM*. ACM, 891–900.

[7] Ran Cheng, Jun Pang, and Yang Zhang. 2015. Inferring friendship from check-in data of location-based social networks. In *ASONAM*. ACM, 1284–1291.

[8] Eunjoon Cho, Seth A Myers, and Jure Leskovec. 2011. Friendship and mobility: user movement in location-based social networks. In *KDD*. ACM, 1082–1090.

[9] Gordon V Cormack, Charles LA Clarke, and Stefan Buettcher. 2009. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *SIGIR*. ACM, 758–759.

[10] Ruifeng Ding and Zhenzhong Chen. 2018. RecNet: a deep neural network for personalized POI recommendation in location-based social networks. *International Journal of Geographical Information Science* (2018), 1–18.

[11] Shanshan Feng, Gao Cong, Bo An, and Yeow Meng Chee. 2017. POI2Vec: Geographical Latent Representation for Predicting Future Visitors.. In *AAAI*. 102–108.

[12] Huiji Gao, Jiliang Tang, and Huan Liu. 2012. Exploring social-historical ties on location-based social networks.. In *ICWSM*.

[13] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *KDD*. ACM, 855–864.

[14] Huan Gui, Jialu Liu, Fangbo Tao, Meng Jiang, Brandon Norick, and Jiawei Han. 2016. Large-scale embedding learning in heterogeneous event data. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on*. IEEE, 907–912.

[15] John Hopcroft and Ravi Kannan. 2014. *Foundations of data science*.

[16] Rana Hussein, Dingqi Yang, and Philippe Cudré-Mauroux. 2018. Are Meta-Paths Necessary?: Revisiting Heterogeneous Graph Embeddings. In *CIKM*. ACM, 437–446.

[17] Leo Katz. 1953. A new status index derived from sociometric analysis. *Psychometrika* 18, 1 (1953), 39–43.

[18] Defu Lian, Xing Xie, Vincent W Zheng, Nicholas Jing Yuan, Fuzheng Zhang, and Enhong Chen. 2015. CEPR: A collaborative exploration and periodically returning model for location prediction. *TIST* 6, 1 (2015), 8.

[19] David Liben-Nowell and Jon Kleinberg. 2007. The link-prediction problem for social networks. *journal of the Association for Information Science and Technology* 58, 7 (2007), 1019–1031.

[20] Linyuan Lü and Tao Zhou. 2011. Link prediction in complex networks: A survey. *Physica A: statistical mechanics and its applications* 390, 6 (2011), 1150–1170.

[21] Anastasios Noulas, Salvatore Scellato, Neal Lathia, and Cecilia Mascolo. 2012. Mining user mobility features for next place prediction in location-based services. In *ICDM*. IEEE, 1038–1043.

[22] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. 2016. Asymmetric transitivity preserving graph embedding. In *KDD*. ACM, 1105–1114.

[23] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *KDD*. ACM, 701–710.

[24] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. 2018. Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec. In *WSDM*. ACM, 459–467.

[25] Adam Sadilek, Henry Kautz, and Jeffrey P Bigham. 2012. Finding your friends and following them to where you are. In *WSDM*. ACM, 723–732.

[26] Salvatore Scellato, Cecilia Mascolo, Mirco Musolesi, and Vito Latora. 2010. Distance Matters: Geo-social Metrics for Online Social Networks.. In *WOSN*.

[27] Salvatore Scellato, Anastasios Noulas, Renaud Lambiotte, and Cecilia Mascolo. 2011. Socio-spatial properties of online location-based social networks. *ICWSM* 11 (2011), 329–336.

[28] Salvatore Scellato, Anastasios Noulas, and Cecilia Mascolo. 2011. Exploiting place features in link prediction on location-based social networks. In *KDD*. ACM, 1046–1054.

[29] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *WWW*. ACM, 1067–1077.

[30] Anton Tsitsulin, Davide Mottin, Panagiotis Karras, and Emmanuel Müller. 2018. VERSE: Versatile Graph Embeddings from Similarity Measures. In *WWW*. International World Wide Web Conferences Steering Committee, 539–548.

[31] Ke Tu, Peng Cui, Xiao Wang, Fei Wang, and Wenwu Zhu. 2017. Structural Deep Embedding for Hyper-Networks. In *AAAI*.

[32] Jorge Valverde-Rebaza, Mathieu Roche, Pascal Poncelet, and Alneu de Andrade Lopes. 2016. Exploiting social and mobility patterns for friendship prediction in location-based social networks. In *ICPR*. IEEE, 2526–2531.

[33] Dashun Wang, Dino Pedreschi, Chaoming Song, Fosca Giannotti, and Albert-Laszlo Barabasi. 2011. Human mobility, social ties, and link prediction. In *KDD*. Acm, 1100–1108.

[34] Yan Wang, Zongxu Qin, Jun Pang, Yang Zhang, and Jin Xin. 2017. Semantic Annotation for Places in LBSN through Graph Embedding. In *CIKM*. ACM, 2343–2346.

[35] Min Xie, Hongzhi Yin, Hao Wang, Fanjiang Xu, Weitong Chen, and Sen Wang. 2016. Learning graph-based poi embedding for location-based recommendation. In *CIKM*. ACM, 15–24.

[36] Dingqi Yang, Daqing Zhang, Longbiao Chen, and Bingqing Qu. 2015. NationTelescope: Monitoring and visualizing large-scale collective behavior in LBSNs. *Journal of Network and Computer Applications* 55 (2015), 170–180.

[37] Dingqi Yang, Daqing Zhang, and Bingqing Qu. 2016. Participatory cultural mapping based on collective behavior data in location-based social networks. *TIST* 7, 3 (2016), 30.

[38] Dingqi Yang, Daqing Zhang, Zhiyong Yu, and Zhu Wang. 2013. A sentiment-enhanced personalized location recommendation system. In *HT*. ACM, 119–128.

[39] Dingqi Yang, Daqing Zhang, Zhiyong Yu, and Zhiwen Yu. 2013. Fine-grained preference-aware location search leveraging crowdsourced digital footprints from LBSNs. In *UbiComp*. ACM, 479–488.

[40] Dingqi Yang, Daqing Zhang, Vincent W Zheng, and Zhiyong Yu. 2015. Modeling user activity preference by leveraging user spatial temporal characteristics in LBSNs. *TSMC* 45, 1 (2015), 129–142.

[41] Zhiwen Yu, Huang Xu, Zhe Yang, and Bin Guo. 2016. Personalized travel package with multi-point-of-interest recommendation based on crowdsourced user footprints. *IEEE Transactions on Human-Machine Systems* 46, 1 (2016), 151–158.

[42] Yang Zhang and Jun Pang. 2015. Distance and friendship: A distance-based model for link prediction in social networks. In *APWeb*. Springer, 55–66.

[43] Wayne Xin Zhao, Feifan Fan, Ji-Rong Wen, and Edward Y Chang. 2018. Joint Representation Learning for Location-Based Social Networks with Multi-Grained Sequential Contexts. *ACM Transactions on Knowledge Discovery from Data* 12, 2 (2018), 22.

[44] Yu Zhu, Ziyu Guan, Shulong Tan, Haifeng Liu, Deng Cai, and Xiaofei He. 2016. Heterogeneous hypergraph embedding for document recommendation. *Neurocomputing* 216 (2016), 150–162.