Differentially-Private Next-Location Prediction with Neural Networks

Ritesh Ahuja
University of Southern California
riteshah@usc.edu

Gabriel Ghinita
University of Massachusetts Boston
gabriel.ghinita@umb.edu

Cyrus Shahabi
University of Southern California
shahabi@usc.edu

ABSTRACT

The emergence of mobile apps (e.g., location-based services, geo-social networks, ride-sharing) led to the collection of vast amounts of trajectory data that greatly benefit the understanding of individual mobility. One problem of particular interest is next-location prediction, which facilitates location-based advertising, point-of-interest recommendation, traffic optimization, etc. However, using individual trajectories to build prediction models introduces serious privacy concerns, since exact whereabouts of users can disclose sensitive information such as their health status or lifestyle choices. Several research efforts focused on privacy-preserving next-location prediction, but they have serious limitations: some use outdated privacy models (e.g., k-anonymity), while others employ learning models with limited expressivity (e.g., matrix factorization). More recent approaches (e.g., DP-SGD) integrate the powerful differential privacy model with neural networks, but they provide only generic and difficult-to-tune methods that do not perform well on location data, which is inherently skewed and sparse.

We propose a technique that builds upon DP-SGD, but adapts it for the requirements of next-location prediction. We focus on user-level privacy, a strong privacy guarantee that protects users regardless of how much data they contribute. Central to our approach is the use of the skip-gram model, and its negative sampling technique. Our work is the first to propose differentially-private learning with skip-grams. In addition, we devise data grouping techniques within the skip-gram framework that pool together trajectories from multiple users in order to accelerate learning and improve model accuracy. Experiments conducted on real datasets demonstrate that our approach significantly boosts prediction accuracy compared to existing DP-SGD techniques.

1 INTRODUCTION

The last decade witnessed a rapid development in mobile devices capabilities, accompanied by the emergence of numerous locations-centric applications, such as point-of-interest (POI) search, geo-social networks, ride-sharing services, etc. As a result, vast amounts of rich trajectory data have become available. Coupled with recent advances in machine learning, these data can benefit numerous application domains, such as traffic analysis, location-based recommendations, homeland security, etc.

Mobile users share their coordinates with service providers (e.g., Google Maps) in exchange for receiving services customized to their location. The service providers analyze the data and create powerful machine learning models. Subsequently, these models can be *(i)* placed on user devices to improve the quality of location-centric services; *(ii)* shared with business affiliates interested in expanding their customer base; or *(iii)* offered in a Machine-Learning-as-a-Service (MLaaS) infrastructure to produce business-critical outcomes and actionable insights (e.g., traffic optimization). Figure 1 illustrates these cases. Given historical trajectories, several approaches exploit recent results in neural networks to produce state-of-the-art POI recommender systems [10, 35, 58]. Even though individual trajectory data are not disclosed directly, the model itself retains significant amounts of specific movement details, which in turn may leak sensitive information about an individual's health status, political orientation, entertainment preferences, etc. The problem is exacerbated by the use of neural networks, which have the tendency to overfit the data, leading to unintended memorization of rare sequences which act as quasi-identifiers of their owners [9, 13]. Hence, significant privacy risks arise if individual location data are used in the learning process without any protection.

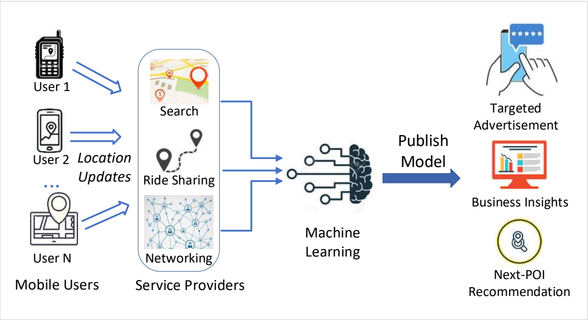


Figure 1: System Model

The research literature identified several fundamental privacy threats that arise when performing machine learning on large collections of individuals' data. One such attack is *membership inference* [25, 52] where an adversary who has access to the model and some information about a targeted individual, can learn whether the target's data was used to train the model. Another attack called *model inversion* [56] makes it possible to infer sensitive points in a trajectory (e.g., a user's favorite bar) from non-sensitive ones (e.g., a user's office). Within the MLaaS setting—where a third party is allowed to only query the model—this implies extracting the training data using only the model's predictions [20].

Iterative procedures such as stochastic gradient descent (SGD) [7] are often used in training deep learning models. Due to the repeated accesses to the data, they raise additional challenges when employing existing privacy techniques. In order to prevent the inference of private information from the training data, recent approaches rely on the powerful *differential privacy (DP)* model [14]. Sequential querying using differentially private mechanisms degrades the overall privacy level. The recent work in [2] provides a tight-bound analysis of the composition of the Gaussian Mechanism for differential privacy under iterative training procedures, enabling the utility of a deep learning model to remain high [39],

© 2020 Copyright held by the owner/author(s). Published in Proceedings of the 23rd International Conference on Extending Database Technology (EDBT), March 30-April 2, 2020, ISBN 978-3-89318-083-7 on OpenProceedings.org.

while preventing the exposure of the training data [6, 27]. While integrating differential privacy techniques into training procedures like stochastic gradient descent is relatively straightforward, computing a tight bound of the privacy loss over multiple iterations is extremely challenging (see Section 6 for a summary of results).

The seminal work in [2] provided *record-level* privacy for a simple feed-forward neural network trained in a centralized manner. The approach provides protection only when each individual contributes a single data item (e.g., a single trajectory). When an individual may contribute multiple data items, a more strict protection level is required, called *user-level* privacy. McMahan et. al. [39] showed that one can achieve *user-level* privacy protection in a federated setting for simple learning tasks. However, ensuring good utility of the trained model for datasets with various characteristics remains a challenge. McMahan et. al. [39] remove skewness in their inputs by pruning each user's data to a threshold, thus discounting the problems of training neural models on inherently sparse location datasets, usually having density around 0.1% [60]. Existing work on privacy-preserving deep learning either assume large and dense datasets, or are evaluated only on dummy datasets [21] that are replicated to a desired size using techniques such as [38]. Such techniques overlook the difficulty of training models on smaller or sparse datasets, which often prevent models from converging [40]. Moreover, they require extensive hyperparameter tuning to achieve good accuracy, and the rough guidelines offered to tune these parameters [37] do not extend to more complex neural architectures, or to datasets different from those used in their work.

We propose a technique that can accurately perform learning on trajectory data. Specifically, we focus on next-location prediction, which is a fundamental and valuable task in location-centric applications. The central idea behind our approach is the use of the skip-gram model [41, 43]. One important property of skip-grams is that they handle well sparse data. At the same time, the use of skip-grams for trajectory data increases the dimensionality of intermediate layers in the neural network. This creates a difficult challenge in the context of privacy-preserving learning, because it increases data sensitivity, and requires a large amount of noise to be introduced, therefore decreasing accuracy.

To address this challenge, we capitalize on the *negative sampling (NS)* technique that can be used in conjunction with skip-grams. NS turns out to be extremely valuable in private gradient descent computation, because it helps reduce the gradient update norms, and thus boosts the ratio of the useful signal compared to the noise introduced by differential privacy. In addition, we introduce a data grouping mechanism that makes learning more effective by combining multiple users into a single bucket, and then training the model per bucket. Grouping has a dual effect: on the positive side, it increases the information diversity in each bucket, improving learning outcomes; on the negative side, it heightens the adverse effect of the introduced Gaussian noise. We study closely this trade-off, and investigate the effect of grouping factors in practice.

Our specific contributions are:

(1) We propose a private learning technique for sparse location data using skip-grams in conjunction with DP-SGD. To our knowledge, this is the first approach to combine skip-grams with DP to build a private ML model. Although our analysis and evaluation focus on location data, we believe that DP-compliant skip-grams can also benefit other scenarios that involve sparse data.
(2) We address the high-dimensionality challenge introduced by skip-grams through the careful use of *negative sampling*, which helps reduce the norm of gradient descent updates, and as a result preserves a good signal-to-noise ratio when perturbing gradients according to the Gaussian mechanism of DP. In addition, we group together data from multiple users into buckets, and run the ML process with each bucket as input. By increasing the diversity of the ML input, we are able to significantly boost learning accuracy.
(3) We perform an extensive experimental evaluation on real-world location check-in data. Our results demonstrate that training a differentially private skip-gram for next-location recommendation clearly outperforms existing approaches for DP-compliant learning. We also perform a thorough empirical exploration of the system parameters to understand in-depth the behavior of the proposed learning model. Our findings show that DP-compliant skip-grams are a powerful and robust approach for location data, and some of the trends that we uncovered can also extend to other types of sparse data, beyond locations.

The rest of the paper is organized as follows: we provide background information in Section 2. Section 3 introduces the system architecture, followed by the details of our private location recommendation technique in Section 4. We perform an extensive experimental evaluation in Section 5. We survey related work in Section 6, followed by conclusions in Section 7.

2 BACKGROUND

2.1 Differential Privacy

Differential Privacy (DP) [17] represents the de-facto standard in protecting individual data. It provides a rigorous mathematical framework with formal protection guarantees, and is the model of choice when releasing aggregate results derived from sensitive data. The type of analyses supported by DP range from simple count or sum queries, to the training of machine learning models. A popular DP flavor that is frequently used in gradient descent due to its refined composition theorems is (ε, δ)-differential privacy. Given non-negative numbers (ε, δ), a randomized algorithm $\mathcal{M}$ satisfies (ε, δ)-differential privacy iff for all datasets D and D' differing in at most one element, and for all $E \subseteq \text{Range}(\mathcal{M})$, the following holds:

$$Pr[\mathcal{M}(D) \in E] \leq e^{\varepsilon} Pr[\mathcal{M}(D') \in E] + \delta \tag{1}$$

The amount of protection provided by DP increases as ε and δ approach 0. Dwork et al. [17] recommend setting δ to be smaller than $1/n$ for a dataset of cardinality n. The parameter ε is called *privacy budget*.

Datasets D and D' that differ in a single element are said to be *neighboring*, or *sibling*. When the adjacency between the datasets is defined with respect to a single data record, then the DP formulation provides *record-level* privacy guarantees. The amount of protection can be extended to account for cases when a single individual contributes multiple data records. In this case, the sibling relationship is defined by allowing D and D' to differ only in the records provided by a single individual. This is a stronger privacy guarantee, called *user-level* privacy.

To achieve (ε, δ)-DP, the result obtained by evaluating a function (e.g., a query) f on the input data must be perturbed by adding noise sampled from a random variable Z. The amount of noise required to ensure the mechanism $\mathcal{M}(D) = f(D) + Z$ satisfies a given privacy guarantee depends on how sensitive the function f is to changes in the input, and the specific distribution chosen for Z. The Gaussian mechanism (GM) [16] is tuned to the sensitivity S_f computed according to the global ℓ_2-norm as $S_f = \sup_{D \simeq D'} ||f(D) - f(D')||_2$ for every pair of sibling datasets D, D' . GM adds zero-mean Gaussian noise calibrated to the function's sensitivity as follows:

Theorem 2.1. *For a query $f : D \to \mathbb{R}$, a mechanism $\mathcal{M}$ that returns $f(D) + Z$, where $Z \sim \mathcal{N}(0, \sigma^2 S_f^2)$ guarantees (ε, δ)-DP if $\sigma^2\varepsilon^2 \geq 2ln(1.25/\delta)$ and $\varepsilon \in [0, 1]$ (see [17] for the proof).*

The *composability* property of DP helps evaluate the effect on privacy when multiple functions are applied to the data (e.g., multiple computation steps). Each step is said to consume a certain amount of privacy budget, and the way the budget is *allocated* across multiple steps can significantly influence data utility.

2.2 Neural Networks

Modern machine learning (ML) models leverage the vast expressive power of artificial neural networks to dramatically improve learning capabilities. Convolutional networks have shown exceptional performance in processing images and video [30]. Recurrent networks can effectively model sequential data such as text, speech and DNA sequences [12, 28]. A neural network is composed of one or more interconnected multilayer stacks, most of which compute non-linear input-output mappings. These layers transform the representation at one level (starting with the raw input) into a representation at a higher, more abstract level. The key to improving inference accuracy with a neural net is to continually modify its internal adjustable parameters.

Stochastic gradient descent (SGD) is the canonical optimization algorithm for training a wide range of ML models, including neural networks. It is an iterative procedure which performs parameter updates for each training example x_i and label y_i. Learning the parameters of a neural network is a nonlinear optimization problem. At each iteration, a batch of data is randomly sampled from the training set. The error between the model's prediction and the training labels, also called *loss*, is computed after each iteration. The loss is then differentiated with respect to the model's parameters, where the derivatives (or gradients) capture their contribution to the error. A *back-propagation* step distributes this error back through the network to change its internal parameters that are used to compute the representation in each layer from the representation in the previous layer. Each internal parameter of the model θ is brought closer to predicting the correct label as follows:

$$\theta = \theta - \eta \cdot \nabla_\theta \mathcal{J}(\theta; x^{(i)}; y^{(i)})$$

where η is the *learning rate* hyper-parameter and $\mathcal{J}$ is the loss function. Iteratively recomputing gradients and applying them to update the model's parameters is referred to as *descent*, and this operation is performed until the model's performance is satisfactory.

2.3 Differentially Private-SGD (DP-SGD)

Introduced in [1], DP-SGD integrates (ε, δ)-DP with neural networks. It modifies traditional SGD in that after calculating the changes in its internal parameters, it obfuscates the gradient values with noise sampled from the Gaussian distribution.

Table 1: Summary of Notations

Notation	Definition
U, P	Sets of users and check-in locations, respectively
N, L	Cardinalities of sets U and P, respectively
U_u	Historical record of user u's check-ins
dim	Dimension of location embedding space
b, η	Batch size and learning rate, respectively
q	User sampling probability per step
m	Expected user sample size per step
ε, δ	Privacy parameters of Gaussian mechanism
σ	Noise scale
λ	Data grouping factor
$\mathcal{H}$	Set of training buckets
C	Per-layer clipping norm

DP-SGD averages together multiple gradient updates induced by training-data examples, *clips* (i.e., truncates) each gradient update to a specified maximum ℓ_2-norm, and adds Gaussian random noise to their averaged value. Clipping each gradient bounds the influence of each training-data example on the model. Accordingly, the sensitivity of the average query can be adjusted as desired, and due to the added noise tuned to the sensitivity of the query, differential privacy is ensured in each iteration. Typically, repeatedly executing a query results in sharp degradation of the privacy protection, as more information is leaked by multiple usages of private iterations. The *moments accountant* technique [1] computes the privacy loss resulting from the composition of Gaussian mechanisms under random sampling. It tracks the moments of the privacy loss variable in each step of the descent, and provides a much tighter upper bound on privacy budget consumption than the standard composition theorem [17].

3 SYSTEM ARCHITECTURE

In Section 3.1 we define the problem statement. We outline the learning model architecture in Section 3.2 and we show how it is utilized in Section 3.3. Table 1 summarizes notations used throughout the paper.

3.1 Problem Statement

Data Representation. The input to our learning model consists of check-in data from a set of N users $U = \{u_1, u_2, ..., u_N\}$. The set of L check-in locations (e.g., points of interest) is denoted as $P = \{l_1, l_2, ..., l_L\}$. Each user $u \in U$ has a historical record of check-ins denoted as $U_u = \{c_1, c_2, ...\}$, where each element c_i is a triplet $\langle u, l, t \rangle$ comprised of user identifier, location and time.

Learning Objective. The objective of our model is to predict the location that a given user u will check into next, given a time-ordered sequence of previous check-ins of the user. The past check-ins can represent the user's current trajectory or his entire check-in history. For each scenario, we describe the usage of the model in Section 3.3. In an initial step, we employ an unsupervised learning method, specifically the skip-gram model [43], to learn the latent distributional context [50] of user movements over the set P of possible check-in locations. A latent representation of every location in a reduced-dimension vector space is the intermediate output. Next, we determine for each user

u its inclination to visit a particular location l by measuring how similar l is in the latent vector space to the locations previously visited by u.

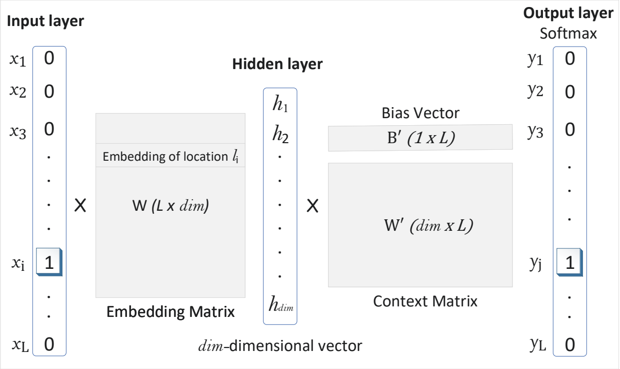


Figure 2: Architecture of the location-recommendation model

3.2 Learning Model

The skip-gram negative sampling (SGNS) model [41, 43] was initially proposed to learn word embeddings from sentences. However, several recent efforts [10, 35, 58] show that the model is also appropriate for location recommendation tasks. Specifically, the model is used to learn location embeddings from user movement sequences, where each location corresponds to a word, and a user's check-in history to a sentence.

Given the set of check-ins of a user, we treat the consecutively visited locations as a trajectory that reflects her visit patterns. A data pre-processing step is required to make the data format compatible with the input of a neural network: every location in P is tokenized to a word in a vocabulary of size $L = |P|$. Given a *target* location check-in c, a symmetric window of *win* context locations to the left and *win* to the right is created to output multiple pairs of target and context locations as training samples. The assumption is that if a model can distinguish between actual pairs of target and context words from random noise, then good location vectors will be learned.

Figure 2 illustrates the neural network used in our solution. The model parameters consist of three tensors $\theta = \{W, W', B'\}$ and two hyper-parameters representing the embedding dimensions *dim* and the negative samples drawn *neg*. Consider a target-context location pair (l_x, l_y). First, both locations are one-hot encoded into binary vectors $\vec{x}$ and $\vec{y}$ of size L. The multiplication of $\vec{x}$ with embedding matrix W produces the embedding vector for the input location l_x (i.e., the i^{th} row of matrix W). $W \times x$ represents the mapping of input location x to a vector $\vec{h}$ in an *dim*-dimensional space. Next, for each positive sample (i.e., true target/context pair), a *neg* number of negative samples are drawn. The context location vector $\vec{y}$ along with the negative samples are passed through a different weight matrix W' and bias vector B'. Finally, a sampled softmax loss function is applied to calculate the prediction error. At a high level (we refer the reader to [49] for a detailed look), the parameters are modified such that the input word (and the corresponding embedding) is tugged closer to its neighbors (i.e., paired context locations), and tugged away from the negative samples. As a result, during back-propagation, only $neg + 1$ vectors in W or W' are updated instead of entire matrices. In the original work [41, 43], negative sampling was devised to improve computational efficiency, as updating the entire model in each iteration can be quite costly. In private learning, it also plays an important role in controlling the adverse affects of noisy training.

We remark here that techniques such as Noise Contrastive Estimation [23] and Negative Sampling use a non-uniform distribution for drawing the samples—for example, by decreasing the sampling weight for the frequent classes—whereas, we use a sampled softmax function with a uniform sampling distribution. This is a necessity for preserving privacy, since estimating the frequency distribution of locations from user-submitted data will cause privacy leakage. Lastly, the embedded vectors are normalized to unit length for efficient use in downstream applications. On top of improving performance [32, 55], normalizing the vectors assists similarity calculation by making cosine similarity and dot-product equivalent.

We detail the privacy-preserving learning model in Section 4. In the remainder of this section, we show how the model, once computed in a privacy-preserving fashion, can be utilized.

3.3 Model Utilization

We provide an overview of how our proposed privacy-preserving next-location prediction model is utilized. Once our privacy-preserving learning technique is executed, the resulting model can be shared with consumers, since the users who contributed the data used in the training are protected according to the semantic model of DP. While the utilization of our model is orthogonal to our proposal, we include it in this section in order to provide a complete, end-to-end description of our solution's functionality.

A typical use of our model is for a mobile user to download[1] it to her device, provide her location history as input, and receive a next-location recommendation. Alternatively, a service provider who already has the locations of its subscribers, will perform the same process to provide a next-location suggestion to a customer. We emphasize that, the model utilization itself does not pose any privacy issues. In both cases above, neither the input, nor the output to the model are shared, so there is no privacy concern. The only time we need to be concerned about privacy is when *training* the model, since a large amount of trajectories from numerous users is required for that task.

Consider a user who has recent check-ins ζ in a relatively short time period (e.g., last few hours). This set of locations forms the basis for recommending to the user the next location to visit. The normalized embedded matrix W in the fully-trained model encodes the latent feature vector of all locations. For each location check-in $l_i \in \zeta$, the embedding vectors $w(l_i)$ are extracted and stacked on top of each other. More precisely, to obtain the embedding vector $w(l_i)$, the binary vector of l_i is multiplied with W (similar to the first step of the training process). This process is equivalent to extracting the *dim*-dimensional row corresponding to location l_i. Then, the average of elements across dimensions of the stacked vectors is computed to produce a representation $\mathcal{F}(\zeta)$ of the recent check-ins of the user. Finally, cosine similarity scores are computed as the dot-product of the vector $\mathcal{F}(\zeta)$ to the embedding vector of each location in the universe L. We rank all locations by their scores and select the top-K locations as the potential recommendations for the user.

In the case when the user has no recent check-ins, the representation $\mathcal{F}()$ can be computed over her movement profile

[1] To reduce communication costs, only the embedding matrix is deployed.

comprising of historical check-ins. Other methods include training an additional model to learn latent feature vectors of each user from her preferences and locations visited. As in [19, 58], a user's feature representation can be used to determine her inclination to visit a particular location. However, modeling each user with such personalized representations, while at the same time preserving *user*-level privacy, is a fundamentally harder problem (in terms of both system design and privacy framework), and is left as future work.

When the model is deployed at an untrusted location-based service provider (LBS), additional privacy concerns must be addressed. In this case, the mobile user must protect the set ζ (or $\mathcal{F}(\zeta)$) *locally*. Techniques such as geo-indistinguishability [3] can be applied to protect the check-in history (discussed in Section 6). For example, the check-in coordinates can be obfuscated to prevent adversaries from pinpointing the user to a certain location with high probability. Addressing these vulnerabilities in the MLaaS setting is orthogonal to the scope of this paper.

4 PRIVATE LOCATION PREDICTION (PLP)

Section 4.1 presents in detail our proposed approach for private next-location prediction. Section 4.2 provides a privacy analysis of our solution.

4.1 Private Location Prediction (PLP)

PLP is a customized solution to location recommendation. It learns latent factor representations of locations while controlling the influence of each user's trajectory data to the training process. Bounding the contribution of a *single* data record in the SGD computation has been proposed in previous work [2, 53]. We make several extensions and contribute data grouping techniques to boost model performance. Even while combining data of multiple users, we guarantee user-level privacy (such as in [21, 39]). By grouping data records of multiple users, we benefit from cross-user learning to improve model performance.

Algorithm 1 depicts the procedure of this learning process. Model hyperparameters labeled batch size β, learning rate η and loss function $\mathcal{J}$ are related to gradient descent optimization, whereas hyperparameters labeled grouping factor λ, sampling probability q, gradient clipping norm bound C, noise scale σ and privacy parameters ε, δ are introduced to create an efficient and privacy-preserving system. We briefly describe each component in isolation before coupling them together to illustrate the big picture.

User Sampling. Given a sampling probability $q = m/N$, each element of the user set is subjected to an independent Bernoulli trial which determines whether the element becomes part of the sample. As a consequence, the size of sampled set of users U_{sample} is equal to m only in expectation. This is a necessary step in correctly accounting for the privacy loss via the moments accountant [2].

Data Grouping. Data grouping is essentially a pre-processing technique that significantly boosts model performance. It has a dual purpose. The first is to reduce the effects of skewness and sparsity inherent to location data, where the frequency of check-ins of users at locations follows the Zipf's law [11]. The second is to provide cross-user learning to smooth updates in the model parameters produced by the function in lines 15-22. The underlying intuition is simple: to ensure good performance of the context model, each update of a training step must contribute to the final result. By combining the profiles of multiples users we also reduce minor observation errors that may be produced from specific data points in a user's profile.

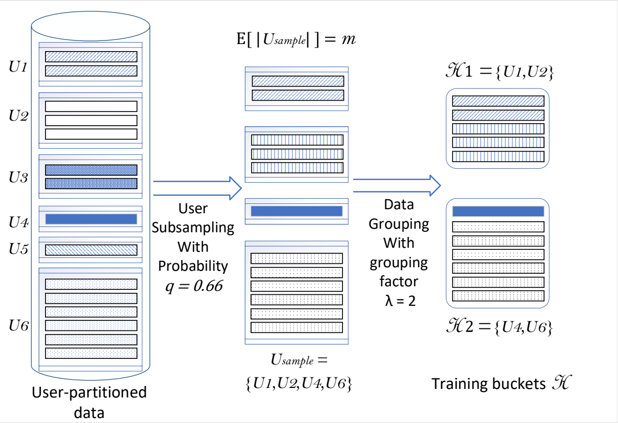

$U1$
$U2$
$U3$
$U4$
$U5$
$U6$
User-partitioned data
User Subsampling With Probability $q = 0.66$
$E[\,|U_{sample}|\,] = m$
$U_{sample} = \{U1,U2,U4,U6\}$
Data Grouping With grouping factor λ = 2
$\mathcal{H}1 = \{U1,U2\}$
$\mathcal{H}2 = \{U4,U6\}$
Training buckets $\mathcal{H}$

Figure 3: Data sampling and grouping

Our data grouping technique agglomerates the data of multiple users into buckets $\mathcal{H}$. Given a grouping factor λ, users (and their entire data) are randomly assigned to buckets such that each bucket contains λ users. This operation is encapsulated in the *groupData*$(\cdot)$ function in line 6. As a separate method, we also tried equal frequency grouping, where a global pass over the record count of each user is used to produce buckets such that each contains approximately the same number of records (while ensuring that the data records of each user are not split into multiple buckets). However, we noticed no statistically significant benefit in model accuracy from equal frequency grouping than with a random grouping. Accordingly, we use the latter in the rest of the work.

Figure 3 illustrates the data sampling and grouping process (corresponding to lines 5-6) for a sampling probability of 0.66 and $\lambda = 2$. Grouped data in each bucket is organized as a single array for processing by gradient descent optimization. Recall from Section 3.2 that a symmetric moving window is applied to create training examples, after the array is read by the *generateBatches*() function (in line 17). A number β of target-context location pairs are placed in each batch.

In brief, at each step of PLP, we sample a random subset of users (line 5), combine the data of multiple users into buckets (line 6), compute a gradient update with bounded ℓ_2 norm from each bucket (lines 7-8), add noise to the sum of the clipped gradients (line 9), take their approximate average, and finally update the model by adding this approximation (line 10). Alongside, a privacy ledger is maintained to keep track of the privacy budget spent in each iteration by recording the values of σ and C (lines 3 and 11). This tracker has the added benefit of allowing privacy accounting at any step of the training process. Given a value of δ and the recorded ledger, the moments accountant can compute the overall privacy cost in terms of ε. This functionality is provided by the cumulative_budget_spent() function in line 12, which implements the moments accountant from [2].

Privacy Mechanism. The gradient values computed in line 20 do not have an a-priori bound. This complicates the application of the Gaussian Mechanism (GM), which is generally tuned to the sensitivity of the performed query. In this particular use case, we employ a Gaussian sum query in line 9, the results of which are then averaged using a fixed-denominator estimator. To bound the sensitivity of this query, a maximum sensitivity of C is enforced

Algorithm 1 Algorithm for Private Location Prediction with user-level privacy.

```
Input: loss function $\mathcal{J}(\theta)$, grouping factor $\lambda$, learning rate $\eta$,
sampling probability $q = m/N$, gradient norm bound $C$, batch
size $\beta$, privacy parameters $\varepsilon, \delta$
1:  procedure TrainPrivateLocationEmbedding
2:      Initialize: Model $\theta_0 = \{W, W', B'\}$,
3:                  Privacy Accounting ledger $\mathcal{A}(\delta, q)$
4:      for each step $t = 1, \ldots$ do
5:          $U_{sample} \leftarrow$ a random sample of $m_t$ users
6:          Initialize buckets $\mathcal{H} \leftarrow groupData(U_{sample}, \lambda)$
7:          for each data bucket $d_h \in \mathcal{H}$ do
8:              $\bar{g}_h \leftarrow$ ModelUpdateFromBucket($\theta_t, d_h$)
9:          $\widehat{g}_t = \frac{1}{|\mathcal{H}|}(\sum_{h \in \mathcal{H}} \bar{g}_h + \mathcal{N}(0, \sigma^2 C^2 I))$      ▷ Noise.
10:         $\theta_{t+1} = \theta_t + \widehat{g}_t$      ▷ Model Update.
11:         $\mathcal{A}$.track_budget($C, \sigma$)
12:         if $\mathcal{A}$.cumulative_budget_spent() $\geq \varepsilon$ then:
13:             return $\theta_{t-1}$
14:
15: function ModelUpdateFromBucket($\theta_t, d_h$)
16:     $\Phi \leftarrow \theta_t$
17:     $\mathcal{B} \leftarrow generateBatches(d_h, \beta)$
18:     for each $b \in \mathcal{B}$ do
19:         $\Phi \leftarrow \Phi - \eta \frac{1}{|b|} \sum_{(x_i, y_i) \in b} \nabla_\Phi \mathcal{J}(\Phi, x_i, y_i)$
20:     $g_h = \Phi - \theta_t$
21:     $\bar{g}_h = g_h / \max(1, \frac{\|g_h\|_2}{C})$      ▷ Gradient Norm Clipping.
22:     return $\bar{g}_h$
```

on every gradient computed on bucket h as follows (equivalent to line 21):

$$\|\bar{g}_h\|_2 = \begin{cases} \|g_h\|_2 & \text{for } \|g_h\|_2 \leq C \\ C & \text{for } \|g_h\|_2 > C. \end{cases}$$

Gradient clipping places a strict limit on the maximum contribution—in terms of its ℓ_2 norm—of the gradient computed on a bucket. Formally, $||\bar{g}_h||_2 \leq C$. The sensitivity of the scaled gradient updates with respect to the summing operation is thus upper bounded by C. Finally, dividing the GM's output by the number of buckets $|\mathcal{H}|$ yields an approximation of the true average of the buckets' updates.

We note that increasing the number of users in each bucket increases the valuable information in each gradient update. At the same time, the noise introduced by the Gaussian mechanism is scaled to the sensitivity of each bucket's update (i.e., C). If too few buckets are utilized, this distortion may exceed a limit, meaning that too much information output by the summing operation is destroyed by the added noise. This will impede any learning progress. We treat the grouping factor λ as a hyper-parameter and tune it.

In a multi-layer neural network such as the one described in our work, each tensor can be set to a different clipping threshold. However, we employ the *per-layer clipping* approach of [37], where given an overall clipping magnitude C, each tensor is clipped to $C/\sqrt{|\theta|}$. In the skip-gram model, $\theta_0 = \{W, W', B'\}$, hence $|\theta| = 3$, so we clip the ℓ_2-norm of each tensor to $C/\sqrt{3}$. However, the effect of clipping on the three tensors is rather different due to the difference in their dimensionality. Context matrix W' is clipped to the same degree as bias vector B', despite the fact that they have dimensions ($L \times dim$) and ($1 \times L$), respectively. While the dimensionality of the embedding matrix W is ($L \times dim$), only a fraction of the weights—proportional to *neg*, instead of L—are considered for clipping due to the sampling of *neg* number of negative examples in the sampled softmax function. Simply put, $||W||_2$ is proportional to *neg* and when carefully tuned, the clipping parameter is large enough that nearly all updates are smaller than the clip value of $C/\sqrt{|\theta|}$, improving the signal-to-noise ratio over iterative computations. We discuss the effect of this parameter in controlling the distortion of Gaussian noise in Section 5.

4.2 Privacy Analysis

Recall that, our proposed system provides *user-level* differential privacy to individuals who contribute their check-in history to the training data. This ensures that all individuals are protected, regardless of how much data they contribute (i.e., even if the length of the check-in history varies significantly across users). Let U_k denote the data of a single user. The sensitivity of the Gaussian Sum Query (GSQ) function w.r.t. to neighboring datasets that differ in the records of a single user is defined as

$$S_{GSQ} = \max_{\{U_{sample}, U_k\}} ||GSQ(U_{sample} \cup U_k) - GSQ(U_{sample})||_2$$

In Algorithm 1, GSQ is executed over the bucket gradients, which complicates the analysis of the privacy properties of the algorithm. We consider two distinct scenarios where a user's data may be assigned to: (i) exactly one bucket; or (ii) more than one bucket. We define ω as the *data split factor*, meaning that a user's data may be placed in at most ω buckets.

Case 1 [$\omega = 1$]. This represents the scenario where multiple (up to λ) users' data may be present in a single bucket, but a single user's data may be allocated to at most one bucket. Figure 4(a) depicts this case, which is assumed by default in Algorithm 1. This is a sufficient condition to ensure that the per-user contribution to a bucket's gradient update is tightly bounded. Formally, there exists a unique $d_h \in \mathcal{H}$ s.t. $U_k \subseteq d_h$. In addition, when the ℓ_2 norm of the gradient $||\bar{g}_h||_2$ computed on a data-bucket d_h is upper-bounded by the clipping factor C, we get

$$S_{GSQ} \leq \max_{\{\mathcal{H}, d_h\}} ||GSQ(\mathcal{H} \cup d_h) - GSQ(\mathcal{H})||_2 \leq C$$

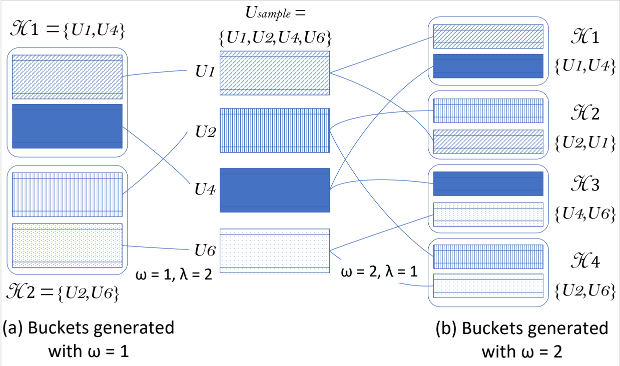
An informal proof that this approach satisfies (ε, δ)-DP is as follows: The sensitivity of the gaussian sum query $GSQ = \sum_{h \in \mathcal{H}} \bar{g}_h$ is bounded as $S_{GSQ} \leq C$, if for all buckets we have $||\bar{g}_h||_2 \leq C$. By extension, if a sampled user (and his location visits) can be assigned to exactly one bucket, *sibling* datasets that differ in the data of a single user can change the output of GSQ by at most C. Therefore, Gaussian Noise drawn from $\mathcal{N}(0, \sigma^2 C^2 I)$ guarantees user-level (ε, δ)-DP.

Case 2 [$\omega > 1$]. If the data of a single user is split over multiple buckets, then it is possible that even after scaling the bucket gradients to C, the sensitivity of the Gaussian sum query is no longer C w.r.t. to user-neighboring datasets. Figure 4(b) illustrates an example with $\omega = 2$. A similar split strategy (proposed in [38]) is used in the empirical evaluation of [21], wherein a small dataset is scaled up to amplify privacy accounting. However, the authors fail to regulate their noise scale to reflect the altered data sensitivity or alternatively recompute the achieved privacy guarantee. We show that when the data of a user U_k is split across multiple buckets, the sensitivity of the query increases to

ω. Assuming that $|\mathcal{H}| \leq |U_{sample}|$, we can write,

$$\omega = \max_{\{U_k \in U_{sample}\}} |\{d_h : d_h \in \mathcal{H} \text{ and } d_h \cap U_k \neq \emptyset\}|$$

meaning that the data of a user can influence the gradients of at most ω buckets. Accordingly, if for all buckets $||\bar{g}_h||_2 \leq C$, a single user can change the output of GSQ by at most ωC. Therefore, to guarantee user-level (ε, δ)-DP, Gaussian Noise must be drawn from $\mathcal{N}(0, \sigma^2\omega^2C^2I)$.



(a) Buckets generated with ω = 1

(b) Buckets generated with ω = 2

Figure 4: Sensitivity of Gaussian Sum Query over U_{sample} users: (a) $\omega = 1$, a single user's data is placed in exactly one bucket; (b) $\omega = 2$, a single user's data is split across two buckets. Since gradients computed over the generated buckets $\mathcal{H}_1, ..., \mathcal{H}_4$, are bounded by C, a user can contribute at most $2C$ to the computed sum.

We remark here that values of $\omega > 1$ produced no positive effect in our evaluation. We experimented with $\omega = 2$ by splitting a user's data to exactly two random buckets. We found that the signal-to-noise ratio is adversely affected, since the marginally improved signal from the split data is offset by the now quadrupled (proportional to ω^2) noise variance. In the rest of the work, we set ω to 1.

5 EXPERIMENTS

Section 5.1 provides the details of the experimental testbed. Section 5.2 focuses on the evaluation of the proposed technique in comparison with the state-of-the-art DP-SGD approach. In Section 5.3 we evaluate in detail our approach when varying system parameters, and provide insights into hyper-parameter tuning.

5.1 Experimental Settings

Dataset. We use a real dataset collected from the operation of a prominent geo-social network, namely *Foursquare* [59]. The data consist of a set of user check-ins. Every check-in is described by a record comprising of user identifier, the latitude and longitude of the check-in, and the identifier of the POI location. In order to simulate a realistic environment of a city and its suburbs, we focus on check-ins within a single urban area, namely Tokyo, Japan. In particular, we consider a large geographical region covering a $35 \times 25\text{km}^2$ area bounded to the South and North by latitudes 35.554, 35.759, and to the West and East by longitudes 139.496, 139.905. We filter out the users with fewer than ten check-ins, as well as the locations visited by fewer than two users (such filtering is commonly performed in the location recommendation literature [33, 61]). The remainder of the data contains a total of 739, 828 check-ins from 4, 602 unique users over 5, 069 locations during a time period of 22 months from April 2012 to January 2014.

Implementation. All algorithms were implemented in Python on a Ubuntu Linux 18.04 LTS operating system. The experiments were executed on an Intel Xeon Platinum 8164 CPU, with 64GB RAM. All data and intermediate structures (e.g., neural network parameters, gradients) are stored in main memory. The proposed neural model is built using Google's Tensorflow library [1]. To account for the privacy budget consumption of the complex iterative mechanism used in learning, we use the privacy accounting method from [54], which allows for a tight composition of privacy-preserving steps. At each step of the computation, we calculate the (ε, δ) tuple from moment bounds, according to the moments accountant procedure introduced in [37].

Evaluation Metric. To evaluate the performance of location recommendation, we adopt the "leave-one-out" approach, which has been widely used in the recommender systems literature [10, 19, 26, 35, 57, 58]. This metric simulates the behavior of a user looking for the next location to visit. Given a time-ordered user check-in sequence, recommendation models utilize the first $(t - 1)$ location visits as an input and predict the t^{th} location as the recommended location. The recommendation quality is measured by *Hit-Rate (HR)*. *HR@k* is a recall-based metric, measuring whether the test location is in the top-k locations of the recommendation list. The outcome of the evaluation is binary: 1 if the test location is included in the output set of the recommender, and 0 otherwise. In the rest of the section, we use the terms *prediction accuracy* and *HR@k* interchangeably.

Model Training. Our testing and validation sets consist of location visits of users who are not part of the training set. Since we do not train models to learn user specific representations (such as in [10, 35, 58]), this is an accurate representation of real-life model utilization at a user's device. Validation and testing sets are created in a similar fashion. First, a randomly selected set of 100 users and their corresponding check-ins are removed from the dataset. From these, time ordered sequences of trajectories are generated. Each individual trajectory does not exceed a total duration of six hours (following the work in [10, 34]). The remaining 4402 users and their check-ins represent the training dataset for learning the parameters of the proposed model.

To train the model, we utilize Adam [29], a widely adopted optimization algorithm for deep neural network models that has specific properties to mitigate disadvantages of traditional SGD, such as its difficulty in escaping from saddle points, or extensive tuning of its learning rate parameter. We implement the optimizer in a differentially private manner by tracking an exponential moving average of the noisy gradient and the squared noisy gradient, as illustrated in [24]. We found that tuning the initial learning rate and decay scheme for Adam only affects the learning in the very first few steps. Typically, Adam does not require extensive tuning [29] and a learning rate between 0.001 to 0.1 is most often appropriate. In our experiments, we found that a learning rate value $\eta \in [0.02, 0.07]$ produces similar results, so we set it to 0.06 for all our runs.

Parameter Settings. We select the training hyper-parameters of the skip-gram model via a cross validation grid search. Figure 5 depicts the validation accuracy over 200 data epochs using the non-private learning approach. We plot the validation Hit-Rate for $k = 5$, 10 and 20 candidates, respectively. We look for those models that reach the highest accuracy. The embedding dimension *dim* is set to 50. While a larger number of hidden units allows more predictive power, the accuracy improvement reaches a plateau when the embedding dimension is in the range [50, 150]. In non-private training, it is preferable to use more units, whereas for private learning a larger model increases the sensitivity of the gradient. We keep our model at the lower end of the *dim* range to keep the number of internal parameters of the models low. The batch size is set to $b = 32$, and the context window parameter $win = 2$ (for a total window size of 5). These parameters are also consistent with those utilized in previous work [10, 58]. Varying the number of negative examples sampled (denoted by *neg*) marginally affects the non-private model, whereas with private learning we find that it directly controls the sensitivity of the private sum query (in Section 5.3 we show experiments on how to tune it). The default value for negative samples is $neg = 16$.

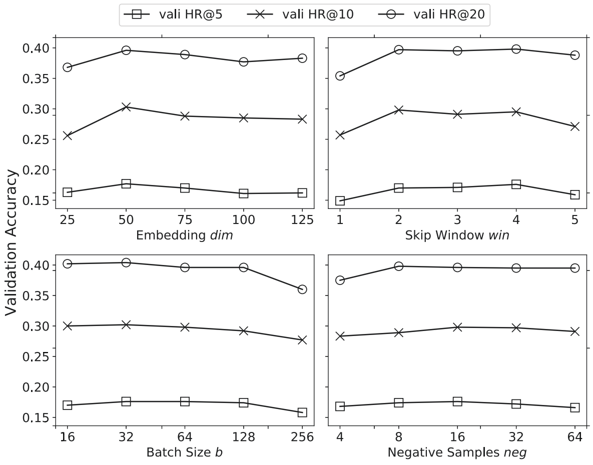


Figure 5: Non-private model hyperparameter tuning

For the privacy parameters, we fix the value of $\delta = 2 \times 10^{-4} < 1/N$ as recommended in previous work on differentially-private machine learning [2, 39]. For a given value of δ, the privacy budget ε affects the amount of steps we can train until we exceed that budget threshold. We set the default value of the hyper-parameters to $q = 0.06$, $\sigma = 2.5$, $C = 0.5$, $\lambda = 4$ (please see Table 1 for a summary of notations). Recall that, the sampling ratio of each lot is $q = m/N$, so each epoch consists of $1/q$ steps.

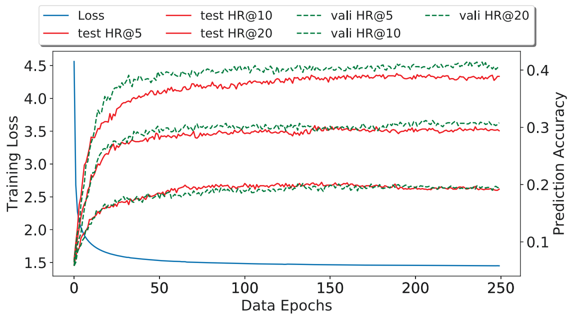


Figure 6: Non-private model performance

5.2 Comparison with Baseline

We evaluate the performance of our proposed approach in comparison with two baselines: *(i)* a non-private learning approach using SGD, and *(ii)* the state-of-the-art user-level DP-SGD approach from [2, 39].

First, we evaluate the non-private location prediction model described in Section 3.2. Figure 6 illustrates the validation and testing Hit-Rate at $k = 5$, 10 and 20. The model generalizes well to the test set, and there appears to be no evidence of overfitting up to 250 data epochs. The presented results are competitive with existing approaches in [35, 58], suggesting that the model hyper-parameters are suitable to capture the underlying semantics of mobility patterns. The best testing accuracy of the non-private model for the *HR*@10 setting is 29.5%.

Throughout our evaluation we found that, when the model is trained in a differentially private manner, there is only a small difference between the model's accuracy on the training and the test sets. This is consistent with both the theoretical argument that differentially private training generalizes well [5, 15], and the empirical evidence in previous studies [2, 39]. For brevity of presentation, in the rest of this section we only show *HR*@10 evaluation results (similar trends were recorded for *HR*@5 and *HR*@20).

Next, we evaluate our proposed Private Location Prediction (PLP) approach in comparison with DP-SGD [2], which is summarized in Section 2. We adapt the model to work on user-partitioned data, so that it guarantees user-level privacy. The improvements of PLP over DP-SGD passed the paired t-test with significance value $p < 0.01$.

Figure 7 plots the prediction accuracy of the privately trained models for varying levels of privacy ε. For each ε value, we consider two settings each for sampling probability $q = 0.06$ (upper left) and $q = 0.10$ (bottom right). We set $\sigma = 1.5$. We compare PLP against the baseline DP-SGD for two values of the grouping factor λ. As expected, a general trend we observed is that providing more privacy budget allows the models to train to a higher accuracy. However, for the baseline approach, the convergence of the model is thwarted because the model update computed on the data of a single user contributes a limited *signal*, which is often offset by the introduced Gaussian noise. On the other hand, the results show that by incorporating data grouping in its design, PLP is able to ameliorate the data sparsity problem inherent to location datasets. The gain is more pronounced when the grouping factor increases (i.e., higher λ).

Next, we measure the effect of sampling probability q on accuracy. From the theoretical model [2], we know that q directly affects the amount of privacy budget utilized in each iteration (q is also called "privacy amplification factor"). A lower sampling rate includes less data in each iteration, hence the amount of budget consumed in each step is decreased. Our results in Figure 8 confirm this trend. We vary the rate of user sampling q from 4% to 12%. For all runs, we fixed the budget allowance at $\varepsilon = 2$. For a higher sampling probability, the privacy budget is consumed faster, hence the count of total training steps is smaller, leading to lower accuracy. Our proposed PLP method clearly outperforms DP-SGD, whose accuracy drops sharply with an increase in q. We note that, due to the proposed grouping strategy, PLP is more robust to changes in sampling rate, as its accuracy degrades gracefully. In general, a larger bucket cardinality leads to better accuracy, except for the lowest considered sampling rate, where the small fraction of records included in the computation at each

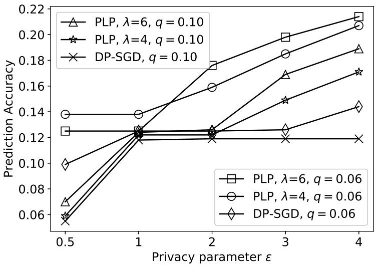


Figure 7: PLP vs DP-SGD: varying privacy budget ε

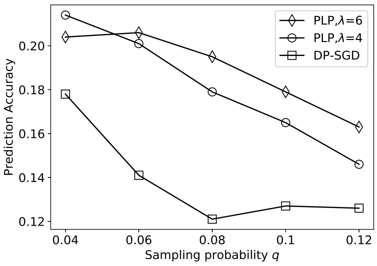


Figure 8: PLP vs DP-SGD: varying sampling ratio q

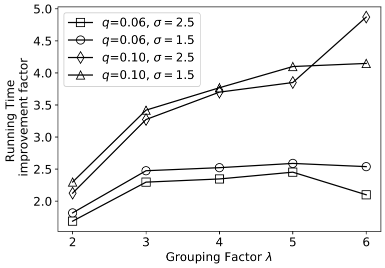


Figure 9: Running time, varying grouping factor λ

step prevents buckets from reaching a significant diversity in their composition.

Finally, we provide a result on the runtime improvements offered by PLP. The y-axis in Figure 9 depicts the multiplicative factor by which PLP is faster that DP-SGD. We show results for two values of q, and for each we present the runtime with two values of noise scale. Linearly scaling the grouping factor has two opposing effects: on the one hand, fewer buckets implies that equally few bucket gradients need to be computed and averaged. On the other hand, as each bucket gets assigned more users, it takes longer to compute each bucket gradient. When fewer users are sampled (i.e., $q = 0.06$) the latter effect begins to dominate for $\lambda > 5$, whereas for $\lambda \in [2, 5]$, the computational efficiency scales from 1.6× to 2.5×. In the setting where sampling rate is higher, at $q = 0.10$, the runtime improvements scale monotonically, to over 4.8× for $\lambda = 6$. These results are consistently observed even with a different number of total iterations (as a larger σ allows more iterations).

In summary, our results so far show that PLP clearly outperforms the existing state-of-the-art DP-SGD. Furthermore, its accuracy was observed to reach values as high as 24%, which is quite reasonable compared to the maximum of 29.5% reached by the non-private learning approach. In the rest of the evaluation, we no longer consider DP-SGD, and we focus on tuning the parameters of the proposed PLP technique.

5.3 Hyper-parameter Tuning

The objective of tuning model hyper-parameters is to obtain a good balance of accuracy and computational overhead of learning. We focus on the following parameters, which we observed throughout the experiments to have a significant influence: grouping factor λ, noise scale σ, the magnitude of ℓ_2 clipping norm, and the number of negative samples *neg*.

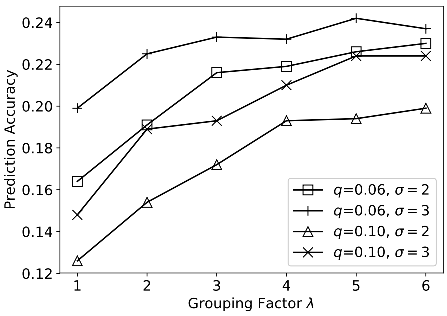


Figure 10: Effect of varying λ

Effect of Grouping factor λ. Figure 10 shows the influence on accuracy of grouping factor λ. We consider two distinct settings each of sampling parameter q and noise scale σ (for a total of four lines in the graph). To limit sensitivity, we clip the gradient norm of each tensor to a maximum l_2 norm of 0.5. Choosing the grouping factor must balance two conflicting objectives: on the one hand, assigning the data of multiple users to the same bucket improves the signal in each bucket gradient, by improving the data diversity within the bucket. On the other hand, the Gaussian noise must be scaled to the sensitivity of a bucket gradient, and a larger bucket size results to fewer buckets, which in turn increases the effect of added noise. Our results confirm this trade-off: initially, when λ increases there is a pronounced increase in accuracy. After a certain point, the accuracy levels off, and reaches a plateau around the value of $\lambda = 5$. When the grouping factor is increased further (not shown in the graph), the accuracy starts decreasing, because there is no significant gain in per-bucket diversity, whereas the relative noise-to-signal ratio keeps increasing.

Effect of Noise Scale σ. The noise scale parameter σ directly controls the noise added in each step. A larger σ leads to more noise, but at the same time it decreases the budget consumption per step, which in turn allows the execution of more learning steps. Figure 11 depicts the model accuracy for varying settings of noise scale. The results presented correspond to two settings each of sampling rate and privacy budget (for a total of four lines). We observe that for the lower-range of σ values, the accuracy is rather poor, especially for smaller settings of privacy budget ε. This is explained by the fact that too little noise is added per step, and the privacy consumption per step is high. As a result, only a small number of steps can be executed before the privacy budget is exhausted, leading to insufficient learning. For larger ε settings, the effect is less pronounced, because there is sufficient budget to allow more steps, even when the noise scale is low.

Conversely, a larger σ allows more steps to be executed, so the best accuracy is obtained for the largest $\sigma = 3.0$ setting. However, we also note that the accuracy levels off towards that setting. For larger σ values (not showed in the graph), we observed that the noise magnitude is too high, and even if budget is slowly exhausted, the training loss in each learning step is excessively high, preventing the model from converging, and leading to very low accuracy. We conclude that the choice of noise scale must be carefully considered relative to the total privacy budget, such that a sufficient number of steps are allowed to execute, while at the same time the loss function value per step is not excessive.

The total number of executed steps also influences the computational overhead of learning. If execution time is a concern, one may want to reduce the number of steps by reducing σ, in an attempt to accelerate the learning (intuitively, since less noise is added at each step, the model will converge faster). This approach is still subject to ensuring that a sufficient number of steps are executed, as neural networks need to perform several complete passes over the dataset.

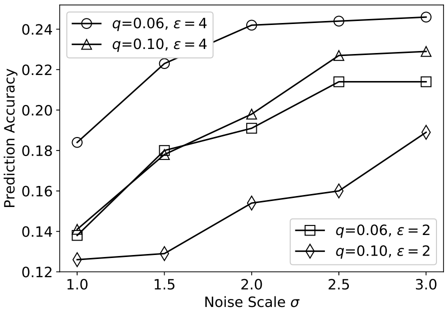


Figure 11: Effect of varying σ

Effect of Clipping norm C. We vary the clipping bound of every tensor in the model $\theta_0 = \{W, W', B'\}$. The value C represents the magnitude of per-tensor clipping, which is set to be equal for every tensor in the model. Clipping more aggressively decreases sensitivity, which in turn leads to a lower privacy budget consumption per step, and allows additional learning iterations to be executed. Conversely, setting the threshold too low also limits the amount of learning that the model can achieve per step. Figure 12 plots the obtained results for several combinations of sampling probability and grouping factor.

We observe that the for the range of values considered, the decrease in sensitivity has a more pronounced impact, and as a result the smaller clipping bounds lead to better accuracy. Of course, one cannot set the clipping bound arbitrarily low, as that will significantly curtail learning. Another factor to consider is the nature of the data, and the effect on gradient values. If the norm of the resulting tensors following gradient computation is high, then a low clipping threshold will destroy the information and prevent learning. In our case, we were able to keep the gradient norm low by using negative sampling, which in turn allowed us to obtain good accuracy for that setting. In cases where this is not possible, it is recommended to increase the clipping threshold value.

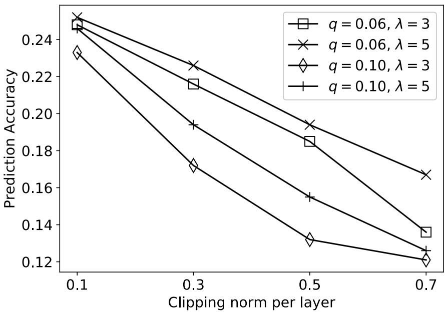


Figure 12: Effect of varying ℓ_2 clipping norm

Effect of Negative samples *neg*. In our final experiment, we investigate the effect on accuracy of negative sampling, which is an important factor in the training success of a skip-gram model. We plot the model accuracy for various values of negative sampling in Figure 13. The number of negative samples *neg* controls the total fraction of weights that are updated for each training sample, and as a side effect it helps keeping the gradient norm low. We can observe a clear 'U'-shaped dependency, reaching a maximum at $neg = 16$. The observed trend is the result of two conflicting factors: if the number of negative samples is too low, training is slowed down, due to the fact that only a small part of the layers are updated per step. Conversely, if too many samples are drawn, then the correspondingly many parameters that need to be updated lead to a large norm. Gradient clipping has an aggressive effect, and as a result, the amount of information that can be learned in each update is obliterated by the noise.

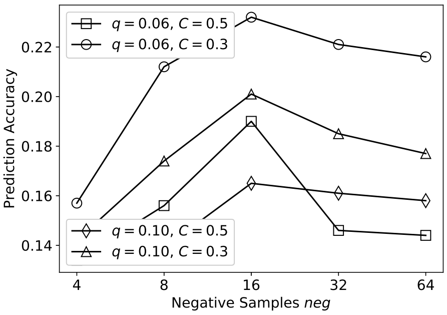


Figure 13: Effect of varying *neg*

6 RELATED WORK

Location recommendation. The problems of location recommendation and prediction have received significant attention in the last decade. Recommending a location to visit to a user necessitates modeling human mobility for the sequential prediction task. Markov Chain (MC) based methods, Matrix Factorization (MF) techniques, and Neural Network models (NN) are the schemes of choice for this objective. MC-based methods utilize a per-user transition matrix comprised of location-location transition probabilities computed from the historical record of check-ins [62]. The m^{th}-order Markov chains emit the probability of the user visiting the next location based on the latest m visited locations. Private location recommendation over Markov Chains is studied in [63]. Aggregate counts of check-ins in discretized regions are published as differentially private statistics. However, due to the sparsity in check-in behavior and the general-purpose privacy mechanisms, their method can only extend to coarse spatial decompositions (e.g., grids having larger than $5km^2$ cells). Factorizing Personalized Markov Chains (FPMC) [47] extend MC by factorizing this transition matrix for the collaborative filtering task. Matrices containing implicit user feedback on locations can also be exploited for location recommendation via weighted matrix factorization [33]. Private Matrix Factorization has been explored in [36, 51], but we are not aware of any proposal for their application to the problem we are considering. Neural Networks have become a powerful tool in recommender applications due to their flexibility, expressive power and non-linearity. Recurrent Neural Networks (RNN) can model sequential data effectively, especially language sentences [42]. Recurrent nets have also been adapted for location sequences [34, 64]. However, RNNs assume that temporal dependency changes monotonically with the position in a sequence. This is often a poor assumption in sparse location data. As a result, the state-of-art [10, 19, 35, 58] employs the skip-gram model [43] to learn the distributional context of users check-in behavior. Extensions incorporate temporal [19, 35, 61], textual [10] and other contextual features [58]. However, none of these studies provide any privacy features, which is the crux of our work.

Differential Privacy (DP) and Neural Networks. A recent focus in the differential privacy literature is to reason about cumulative privacy loss over multiple private computations given the values of ε used in each individual computation [8, 18, 44, 54]. A fundamental tool used in this task is privacy amplification via sampling [4], wherein the privacy guarantees of a private mechanism are amplified when the mechanism is applied to a small random subsample of records from a given dataset. Abadi et. al. [2] provide an amplification result for the Gaussian output perturbation mechanisms under Poisson subsampling. Their technique, called moments accountant, is based on the moment generating function of the privacy loss random variable. Other privacy definitions that lend themselves to tighter composition include Rényi Differential Privacy [44] and zero-Concentrated Differential Privacy [8], and their application to private learning with data subsampling ([54],[31] respectively). However, these privacy models are relatively new and the distinctions in privacy guarantees at the user-end remain to be investigated. In practice, (ε, δ)-differential privacy is the de-facto privacy standard [21, 39].

Location Privacy We overview literature that focuses on preventing the location based service provider (the adversary) from inferring a mobile user's location in the online setting. Spatial k-anonymity (SKA) [22] generalizes the specific position of the querying user to a region that encloses at least k users. The resulting anonymity set bounds the adversary's probability of identifying the query user to at most $1/k$. However, this syntactic notion of privacy can be easily circumvented when the data are *sparse*, i.e., the distribution of the number of location visits of an average user over the universe of POIs is long-tailed. Moreover, check-ins in sparse regions are especially vulnerable to an adversary with background knowledge, significantly increasing the probability that de-anonymization succeeds [45]. Another source of leakage is when the querying user moves, disconnecting himself from the anonymity set. DP can be used in the context of publishing statistics over datasets of locations or trajectories collected from mobile users. The Local Differential Privacy paradigm is well suited for this purpose, and its application to location data is explored in [46]. The Randomized Response mechanism is used to report, in addition to users actual locations, a large number of erroneous locations. Recommendation models that utilize these statistics can at best leverage spatial proximity queries [48] or apply to coarse spatial decompositions [46], and are incapable of cross-user learning such as in the case of the skip-gram model. Lastly, adapting the powerful guarantees of DP to protecting exact location coordinates, Geo-indistinguishability (GeoInd) [3] relaxes the DP definition to the euclidean space. It is the privacy framework of choice for obfuscating user check-ins in the absence of a trusted data curator.

Note that, SKA and GeoInd rely on obfuscating individual location records that make up the larger dataset, making them suitable only for applications that utilize spatial proximity queries (e.g., a user that sends noisy coordinates to obtain points of interest in her vicinity). Utilizing these methods to publish data for training ML models is not viable, since adding noise to the coordinates wipes out any contextual information on the POI visited (beginning with the POI identifier). Moreover, since the same user may have numerous check-in records in a longitudinal location dataset, data publishing with the common techniques suffers from serious privacy leakages. *User-level correlations* (e.g., multiple checkins of a user that are closely related) severely increase the possibility of de-anonymization in the case of SKA. Likewise, in the case of GeoInd, the cumulative privacy loss variable calculated via a standard composition theorem exceeds reasonable privacy levels.

7 CONCLUSIONS

We proposed a new approach for differentially-private next-location prediction using the skip-gram model. To the best of our knowledge, ours is the first technique that deploys DP-SGD for skip-grams. We made use of negative sampling to reduce the norms of gradient updates when dealing with high-dimensional internal neural network layers, and provided a data grouping technique that can improve the signal-to-noise ratio and allows for effective private learning. Our extensive experiments show that the proposed technique outperforms the state-of-the-art, and they also provide insights into how to tune system parameters.

Although our results focus on location data, we believe that our findings can be extended to other types of sparse data. In future work, we plan to test the viability of our approach for other learning tasks. Furthermore, we plan to investigate flexible privacy budget allocation strategies across different stages of the learning process, such that accuracy is further improved. Finally, we will study more sophisticated data grouping approaches that make informed decisions on which users to place together in the same bucket. Since such decisions are data dependent, a careful trade-off must be considered between the budget consumed performing the grouping and the remaining budget used for learning, such that prediction accuracy is maximized.

Acknowledgements. This research has been funded in part by NSF grants IIS-1910950 and IIS-1909806, the USC Integrated Media Systems Center (IMSC), and unrestricted cash gifts from Microsoft and Google. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of any of the sponsors such as the National Science Foundation.

REFERENCES

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pages 265–283, 2016.

[2] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 308–318. ACM, 2016.

[3] M. E. Andrés, N. E. Bordenabe, K. Chatzikokolakis, and C. Palamidessi. Geo-indistinguishability: Differential privacy for location-based systems. In *ACM CCS*, 2013.

[4] B. Balle, G. Barthe, and M. Gaboardi. Privacy amplification by subsampling: Tight analyses via couplings and divergences. In *Advances in Neural Information Processing Systems*, pages 6277–6287, 2018.

[5] R. Bassily, K. Nissim, A. Smith, T. Steinke, U. Stemmer, and J. Ullman. Algorithmic stability for adaptive data analysis. In *Proceedings of ACM Symposium on Theory of Computing*, pages 1046–1059, 2016.

[6] R. Bassily, A. Smith, and A. Thakurta. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *IEEE Symposium on Foundations of Computer Science*, pages 464–473, 2014.

[7] L. Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. 2010.

[8] M. Bun and T. Steinke. Concentrated differential privacy: Simplifications, extensions, and lower bounds. In *Theory of Cryptography Conference*, pages 635–658. Springer, 2016.

[9] N. Carlini, C. Liu, J. Kos, Ú. Erlingsson, and D. Song. The secret sharer: Measuring unintended neural network memorization & extracting secrets.

arXiv preprint arXiv:1802.08232, 2018.

[10] B. Chang, Y. Park, D. Park, S. Kim, and J. Kang. Content-aware hierarchical point-of-interest embedding model for successive poi recommendation. In *IJCAI*, pages 3301–3307, 2018.

[11] E. Cho, S. A. Myers, and J. Leskovec. Friendship and mobility: user movement in location-based social networks. In *Proc. of ACM SIGKDD Conf. on Knowledge discovery and data mining*, pages 1082–1090, 2011.

[12] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(Aug):2493–2537, 2011.

[13] Y.-A. De Montjoye, C. A. Hidalgo, M. Verleysen, and V. D. Blondel. Unique in the crowd: The privacy bounds of human mobility. *Scientific reports*, 3:1376, 2013.

[14] C. Dwork. Differential privacy: A survey of results. In *Theory and Applications of Models of Computation*, pages 1–19, 2008.

[15] C. Dwork, V. Feldman, M. Hardt, T. Pitassi, O. Reingold, and A. Roth. Generalization in adaptive data analysis and holdout reuse. In *Advances in Neural Information Processing Systems*, pages 2350–2358, 2015.

[16] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006.

[17] C. Dwork, A. Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[18] C. Dwork and G. N. Rothblum. Concentrated differential privacy. *arXiv preprint arXiv:1603.01887*, 2016.

[19] S. Feng, G. Cong, B. An, and Y. M. Chee. Poi2vec: Geographical latent representation for predicting future visitors. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[20] M. Fredrikson, S. Jha, and T. Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1322–1333. ACM, 2015.

[21] R. C. Geyer, T. Klein, and M. Nabi. Differentially private federated learning: A client level perspective. *arXiv preprint arXiv:1712.07557*, 2017.

[22] M. Gruteser and D. Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *Proceedings of the 1st international conference on Mobile systems, applications and services*, pages 31–42. ACM, 2003.

[23] M. U. Gutmann and A. Hyvärinen. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of Machine Learning Research*, 13(Feb):307–361, 2012.

[24] R. Gylberth, R. Adnan, S. Yazid, and T. Basaruddin. Differentially private optimization algorithms for deep neural networks. In *2017 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, pages 387–394. IEEE, 2017.

[25] J. Hayes, L. Melis, G. Danezis, and E. De Cristofaro. Logan: Membership inference attacks against generative models. *Proceedings on Privacy Enhancing Technologies*, 2019(1):133–152, 2019.

[26] X. He, Z. He, J. Song, Z. Liu, Y.-G. Jiang, and T.-S. Chua. Nais: Neural attentive item similarity model for recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 30(12):2354–2366, 2018.

[27] B. Hitaj, G. Ateniese, and F. Pérez-Cruz. Deep models under the gan: information leakage from collaborative deep learning. In *Proc. of ACM Conf. on Computer and Communications Security*, pages 603–618, 2017.

[28] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[29] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[30] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[31] J. Lee and D. Kifer. Concentrated differentially private gradient descent with adaptive per-iteration privacy budget. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1656–1665. ACM, 2018.

[32] O. Levy, Y. Goldberg, and I. Dagan. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225, 2015.

[33] D. Lian, C. Zhao, X. Xie, G. Sun, E. Chen, and Y. Rui. Geomf: joint geographical modeling and matrix factorization for point-of-interest recommendation. In *Proc. of ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 831–840, 2014.

[34] Q. Liu, S. Wu, L. Wang, and T. Tan. Predicting the next location: A recurrent model with spatial and temporal contexts. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[35] X. Liu, Y. Liu, and X. Li. Exploring the context of locations for personalized location recommendations. In *IJCAI*, pages 1188–1194, 2016.

[36] Z. Liu, Y.-X. Wang, and A. Smola. Fast differentially private matrix factorization. In *Proceedings of the 9th ACM Conference on Recommender Systems*, pages 171–178. ACM, 2015.

[37] H. B. McMahan and G. Andrew. A general approach to adding differential privacy to iterative training procedures. *arXiv preprint arXiv:1812.06210*, 2018.

[38] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, et al. Communication-efficient learning of deep networks from decentralized data. *arXiv preprint arXiv:1602.05629*, 2016.

[39] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang. Learning differentially private language models without losing accuracy. *ICLR*, 2018.

[40] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *IEEE Symposium on Security and Privacy*, 2019.

[41] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[42] T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, and S. Khudanpur. Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*, 2010.

[43] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[44] I. Mironov. Rényi differential privacy. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, pages 263–275. IEEE, 2017.

[45] A. Narayanan and V. Shmatikov. Robust de-anonymization of large sparse datasets. In *2008 IEEE Symposium on Security and Privacy (sp 2008)*, May 2008.

[46] D. Quercia, I. Leontiadis, L. McNamara, C. Mascolo, and J. Crowcroft. Spotme if you can: Randomized responses for location obfuscation on mobile phones. In *2011 31st International Conference on Distributed Computing Systems*, pages 363–372. IEEE, 2011.

[47] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme. Factorizing personalized markov chains for next-basket recommendation. In *Proc. of Intl. Conf. on World Wide Web*, pages 811–820, 2010.

[48] D. Riboni and C. Bettini. Differentially-private release of check-in data for venue recommendation. In *2014 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 190–198. IEEE, 2014.

[49] X. Rong. word2vec parameter learning explained. *arXiv preprint arXiv:1411.2738*, 2014.

[50] M. Sahlgren. The distributional hypothesis. *Italian Journal of Disability Studies*, 20:33–53, 2008.

[51] H. Shin, S. Kim, J. Shin, and X. Xiao. Privacy enhanced matrix factorization for recommendation with local differential privacy. *IEEE Transactions on Knowledge and Data Engineering*, 30(9):1770–1782, 2018.

[52] R. Shokri, M. Stronati, C. Song, and V. Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18. IEEE, 2017.

[53] S. Song, K. Chaudhuri, and A. D. Sarwate. Stochastic gradient descent with differentially private updates. In *2013 IEEE Global Conference on Signal and Information Processing*, pages 245–248. IEEE, 2013.

[54] Y.-X. Wang, B. Balle, and S. Kasiviswanathan. Subsampled renyi differential privacy and analytical moments accountant. *arXiv preprint arXiv:1808.00087*, 2018.

[55] B. J. Wilson and A. M. Schakel. Controlled experiments for word embeddings. *arXiv preprint arXiv:1510.02675*, 2015.

[56] X. Wu, M. Fredrikson, S. Jha, and J. F. Naughton. A methodology for formalizing model-inversion attacks. In *2016 IEEE 29th Computer Security Foundations Symposium (CSF)*, pages 355–370. IEEE, 2016.

[57] X. Xin, X. He, Y. Zhang, Y. Zhang, and J. Jose. Relational collaborative filtering: Modeling multiple item relations for recommendation. *arXiv preprint arXiv:1904.12796*, 2019.

[58] 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 *Proc. of ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*, pages 1245–1254, 2017.

[59] D. Yang, B. Qu, J. Yang, and P. Cudre-Mauroux. Revisiting user mobility and social relationships in lbsns: A hypergraph embedding approach. In *Proc. of Intl. Conf. on World Wide Web*, 2019.

[60] D. Yang, D. Zhang, L. Chen, and B. Qu. Nationtelescope: Monitoring and visualizing large-scale collective behavior in lbsns. *Journal of Network and Computer Applications*, 55:170–180, 2015.

[61] Q. Yuan, G. Cong, Z. Ma, A. Sun, and N. M. Thalmann. Time-aware point-of-interest recommendation. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pages 363–372. ACM, 2013.

[62] C. Zhang, K. Zhang, Q. Yuan, L. Zhang, T. Hanratty, and J. Han. Gmove: Group-level mobility modeling using geo-tagged social media. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1305–1314. ACM, 2016.

[63] J. D. Zhang, G. Ghinita, and C. Y. Chow. Differentially private location recommendations in geosocial networks. In *2014 IEEE 15th International Conference on Mobile Data Management*, volume 1, pages 59–68. IEEE, 2014.

[64] S. Zhao, T. Zhao, I. King, and M. R. Lyu. Geo-teaser: Geo-temporal sequential embedding rank for point-of-interest recommendation. In *Proc. of Intl, Conf. on World Wide Web*, pages 153–162, 2017.