Bridging Collaborative Filtering and Semi-Supervised Learning: A Neural Approach for POI Recommendation

Carl Yang, Lanxiao Bai, Chao Zhang, Quan Yuan and Jiawei Han
University of Illinois, Urbana Champaign
201 N. Goodwin Ave, Urbana, Illinois 61801, USA
{jiyang3, lbai5, czhang82, qyuan, hanj}@illinois.edu

ABSTRACT
Recommender system is one of the most popular data mining topics that keep drawing extensive attention from both academia and industry. Among them, POI (point of interest) recommendation is extremely practical but challenging: it greatly benefits both users and businesses in real-world life, but it is hard due to data scarcity and various context. While a number of algorithms attempt to tackle the problem w.r.t. specific data and problem settings, they often fail when the scenarios change. In this work, we propose to devise a general and principled SSL (semi-supervised learning) framework, to alleviate data scarcity via smoothing among neighboring users and POIs, and treat various context by regularizing user preference based on context graphs. To enable such a framework, we develop PACE (Preference And Context Embedding), a deep neural architecture that jointly learns the embeddings of users and POIs to predict both user preference over POIs and various context associated with users and POIs. We show that PACE successfully bridges CF (collaborative filtering) and SSL by generalizing the de facto methods matrix factorization of CF and graph Laplacian regularization of SSL. Extensive experiments on two real location-based social network datasets demonstrate the effectiveness of PACE.

KEYWORDS
recommender systems; neural networks; collaborative filtering; semi-supervised learning

1 INTRODUCTION
In the era of information explosion, recommender systems play a pivotal role in finding orders within flooded data. They have been not only extensively studied by academia, but also widely explored in data competitions like Netflix Prize\(^3\) and KDD Cup\(^2\), and applied in many on-line services, including E-commerce, on-line news and social media sites. The key concept behind recommender systems is personalized prediction, and the most common approach is known as CF (collaborative filtering), which involves modeling users’ preferences over items based on their past interactions.

\(^1\)http://www.netflixprize.com

Figure 1 gives an illustration of real-world POI recommendation. While it has received considerable research attention in the past several years, the solutions are not satisfying, mainly due to the following two challenges.

**Challenge 1: Data scarcity.** The data scarcity problem suffered by POI recommendations is much worse compared with other recommender systems. E.g., the density of the data used for POI recommendation such as those from Foursquare and Yelp are usually around 0.1% \([24, 45]\), while the density of Netflix data for movie recommendation is around 1.2% \([2]\). Moreover, rather than explicit feedbacks of ratings in ranges (e.g., 1-5) for traditional systems, only binary implicit feedbacks are available in POI recommendations.

The sparsity of training data and the lack of ordinal ratings directly lead to the inefficacy of the de facto CF approach for traditional recommender systems, i.e., MF (matrix factorization) and its various extensions \([24, 25]\).

\(^3\)https://www.yelp.com
\(^4\)https://foursquare.com
\(^5\)https://www.facebook.com/directory/places
\(^6\)https://foursquare.com/about
**Challenge 2: Various context.** While traditional recommender systems basically deal with ratings and reviews, the context for POI recommendations is much richer. First, users’ preference is based on their mobility and the geographical distances among POIs. Most users only visit POIs within small regions. Second, users’ preference is influenced by their social ties, especially on social Apps like Yelp and Foursquare where the check-ins of friends can be seen. Moreover, users’ preference changes over time of the day and may follow certain sequential patterns. E.g., users may visit restaurants during lunch time but bars during the night, and they may visit POIs in specific orders, like home→work→gym→home.

The various contexts of POI recommendations give rise to many hybrid models based on CF [47, 52–54]. However, the modeling towards each type of context is ad-hoc and unstable across different data, as we will discuss in more details in Sec. 2. There lacks a general and principled framework that can easily take various context into account, and automatically benefit from the most important ones to yield satisfactory and stable recommendations.

**Insight: SSL (semi-supervised learning) with context graphs.** In this work, we claim that the SSL framework is a natural and principled alleviation to both challenges of data scarcity and various context for POI recommendation. SSL algorithms aim to leverage unlabeled data to improve the performance of the supervised tasks. They usually jointly optimize two objective functions: the supervised loss over labeled data and the unsupervised loss over both labeled and unlabeled data.

For POI recommendation, as illustrated in Figure 1, the check-in history can be used as direct supervision, while various context can be naturally leveraged as unlabeled data. We instantiate the SSL framework for the POI recommendation through graph-based SSL, because the context can be easily represented by graphs. E.g., social information among users can be naturally built into a graph \( G_U = (V_U, E_U) \), where \( V_U \) is the set of users, and \( E_U \) is the set of edges among friends; geographical distance among POIs can be naturally built into a graph \( G_P = (V_P, E_P) \), where \( V_P \) is the set of POIs, and \( E_P \) is the set of edges between nearby POIs. The graphs can be weighted and heterogeneous to closely represent more complex context as we will discuss later in Sec. 3.3. In this paper, we call such affinity graphs like \( G_U \) and \( G_P \) the context graphs.

By enforcing smoothness among neighboring users and POIs on context graphs, unlabeled rich context on both user and POI sides is leveraged to address the data scarcity on labeled user preference, i.e., the check-in data as implicit feedback, thus improving the overall performance of CF. But the challenge is, what is an effective approach for bridging CF and graph-based SSL?

**Approach: Neural embedding.** This work explores the use of deep neural networks for learning and regularizing user preference over POIs through joint user/POI embedding. Neural networks have been found effective in many domains, ranging from image recognition [12], speech recognition [15], to text processing [8]. Moreover, it has been shown that embeddings trained with distributional contexts can be used to boost the performance of related tasks [31, 39, 46]. However, to the best of our knowledge, there is no previous work on employing neural embedding for POI recommendation, as a bridge between CF and SSL.

In this work, we learn user embeddings and POI embeddings simultaneously, w.r.t. two types of objective functions. Unlike the most recent semi-supervised network embedding work [46] that only learns node embeddings on a graph, we jointly model users and POIs in separate latent factor spaces, to fully leverage user-POI interactions. Rather than most other existing embedding works that are learned to only predict the distributional context [11, 31, 39], our model is jointly trained w.r.t. two types of objective functions to both predict user preference over POIs and enforce smoothness among users and POIs based on various context.

The main contributions of this work are summarized as follows:

1. We develop PACE (Preference And Context Embedding), a general and principled combination of CF and SSL based on neural networks to model user preference over POIs.
2. We closely study the connections among PACE, MF and GLR, so as to understand why PACE successfully addresses the challenges of data scarcity and various context.
3. We perform extensive experiments on two real-world datasets to demonstrate the effectiveness of PACE and comprehensively analyze the three key components of PACE.

The rest of this paper is organized as following. Related works are discussed in Section 2. Section 3 will cover our PACE deep architecture and the semi-supervised POI recommendation pipeline in detail. We will present extensive experimental results on multiple real-world datasets and check-in datasets in Section 4. A quick summery is provided in Section 5.

## 2 RELATED WORK

### 2.1 Modeling Implicit Feedback

To model user preference, while traditional recommender systems have largely focused on explicit ratings [34, 35], POI recommendations usually deal with implicit feedback, which is more practical but challenging [14, 26, 30, 33]. On the one hand, from a (1-5) score in explicit ratings to a single check-in, the ordinal information is missing and the strength of MF-based algorithms in continuously modeling the ratings is useless [24, 25]. On the other hand, a single score 0 corresponding to an unobserved entry can mean either negative or missing. To handle the ambiguity, recent algorithms either treat all unobserved entries as negative feedback [14, 17] or sample negative instances from all unobserved entries [26, 30].

Our approach based on neural networks can be trained on logistic (0-1) predictions, which is naturally proper for the binary schema of implicit feedback. Moreover, following [29, 46], we design a negative sampling process that can be done dynamically along training to effectively mitigate missing negative feedback.

### 2.2 Exploring Context Information

While traditional recommender systems mostly work with ratings and reviews [20], POI recommendations naturally come with various context, such as geographical information [24, 25, 28, 47, 49], categorical information [53, 55], social information [22, 38, 50, 52], temporal information [48, 56], etc. As we discussed in Sec. 1, user preference is largely shaped by many complex factors. While most works present promising results on some specific problem settings and evaluation data, if the settings and data are changed, there is no
guarantee that the models can still perform well or even work. E.g., when the geographical information is missing, many algorithms based on location modeling cease to work; on the other hand, many location modeling algorithms cannot leverage social information, and they are shown to perform poorly in cold start situations [38].

PACE is a general and principled framework that can easily take various context information into account, while continues to work well when any of them is unavailable. As we will discuss in Sec. 3, graphs are powerful in representing the interactions among users and POIs, and it is easy to transform various contextual relations into graphs. As we also show in Sec. 4.3, our model works as long as the basic feedback information of check-ins is available, and its performance generally improves when more context information becomes available.

2.3 Integrating Unlabeled Data

SSL aims at leveraging unlabeled data to boost the overall learning performance. The major approaches are based on affinity graphs, which can be either computed as distances among instances [1, 42, 58, 59], or derived from external data, such as knowledge graphs [43], social networks or citation networks [5, 18]. In this work, we focus on graphs constructed from various context associated in POI recommendation systems.

Graph-based SSL is based on the smoothness assumption that nearby nodes tend to have similar labels [1, 27, 58, 59]. Their general loss function can be written as

\[
\sum_{i \in \mathcal{L}} l(y_i, f(x_i)) + \lambda \sum_{i,j \in \mathcal{L}} a_{ij} ||f(x_i) - f(x_j)||^2_	ext{2}
= \sum_{i \in \mathcal{L}} l(y_i, f(x_i)) + \lambda \mathbf{A}^T \mathbf{L} \mathbf{f},
\]

where matrix \(\mathbf{A}\) encodes the affinity among nodes, and \(f(\cdot)\) is the prediction function to be learned that maps node features \(x\) to labels \(y\). The first part of Eq. 1 is the supervised loss on labeled data, where \(\mathcal{L}\) is the set of labeled nodes, and \(l(\cdot, \cdot)\) can be any proper loss function. It requires the prediction to be close to the true labels. The second term is the unsupervised regularization loss on all data. \(\mathcal{L}\) is the graph Laplacian matrix defined as \(\mathcal{L} = \mathbf{A} - \mathbf{D}\), where \(\mathbf{A}\) is the affinity graph in matrix form, and \(\mathbf{D}\) is the diagonal matrix with \(d_{ii} = \sum_j a_{ij}\). It requires the predictions to be close on nearby nodes. \(\lambda\) is a constant weighting factor.

To leverage SSL for modeling user preference over POIs, we follow the semi-supervised embedding framework proposed by [42], which extends the regularization term in Eq. 1 into \(\sum_{i,j} a_{ij} ||g(x_i) - g(x_j)||^2\), where \(g\) represents the embedding function of instances, which can be the output labels, hidden layers or auxiliary embeddings in neural networks. In this way, the constraints derived from rich unlabeled context can be properly built into the training of the embedding/prediction model.

2.4 Leveraging Neural Networks

Neural networks have been widely leveraged for traditional recommender systems with explicit ratings. The earliest success might be the two-layer Restricted Boltzmann Machines [34], which discretizes user ratings and models the hidden features of users and items underlying the ratings. [32] later extends it to accommodate the ordinal nature of ratings and [57] further improves it with the autoregressive method. Recently, autoencoders have also been successfully adopted for learning user representations based on rated items [23, 36, 44]. To further improve user personalization, various user and item features are also incorporated and learned through deep neural networks with embeddings [6]. However, these algorithms mostly focus on explicit feedback and models the observed data only, which do not work well on positive-only implicit data in POI recommendations.

For implicit feedback, neural networks have been explored to model context information, such as textual description of items [40], acoustic features of musics [41], cross-domain behaviors of users [9] and the information in knowledge bases [51]. However, the features of users and items are learned separately from feedback, rather than jointly in an end-to-end deep neural network to optimize the prediction of user preference as we consider in this work.

In order to model the user-POI interactions, we are inspired by the neural architectures devised for relation learning that combine MF with multiple layers of perceptrons [4, 13, 37]. However, while they model the interactions among entities via embedding, the entities we consider are heterogeneous and the embeddings for users and POIs are naturally in different spaces. Moreover, to accommodate various context information, we build an SSL framework to incorporate user-user and POI-POI dependencies based on path sampling [31] and Skipgram [29] on context graphs.

3 PACE

3.1 Overall Framework

Input. In POI recommendations, the basic input is usually a very sparse 0-1 matrix of user-POI check-ins. To keep track of the total \(N\) users and \(M\) POIs, we use a vector \(\mathbf{u}_i\) to model each user \(u_i\) and a vector \(\mathbf{p}_j\) to model each POI \(p_j\). For the simplest case, \(\mathbf{u}_i\) and \(\mathbf{p}_j\) can both be one-hot vectors representing their identities, while more user/POI features can also be extracted from data and included through vector concatenation [16, 20]. In this work, as we focus on the general architecture of CF and SSL instead of feature construction, we take the simple one-hot vectors as input.

Output. Our PACE model makes predictions on both user preference over POIs and the context associated with users and POIs. To treat the implicit feedback of check-ins in POI recommendations, we output a single 0-1 prediction for each pair of input \(u_i\) and \(p_j\) through a softmax layer on the top of our neural network, indicating if \(u_i\) is interested in \(p_j\). Besides, each user \(u_i\) is predicted with a set of context users through another softmax layer with \(N\) sets of parameters corresponding to the total \(N\) users, indicating whether \(u_i\) is close to each of the other users on the user context graph. The same is done to each POI \(p_j\).

Objective. As PACE outputs three groups of predictions, i.e., user-POI preference, user-user context and POI-POI context, we jointly train the model by optimizing the sum of three loss functions as follows.

\[
\mathcal{J} = \mathcal{J}_p + \lambda_1 \mathcal{J}_{c_u} + \lambda_2 \mathcal{J}_{c_p},
\]

where \(\mathcal{J}_p\) is the loss on preference and \(\mathcal{J}_{c_u} + \mathcal{J}_{c_p}\) is the loss on context. \(\lambda\)’s control the trade-off among the three objectives. Eq. 2
can be further simplified to
\[ \mathcal{J} = \mathcal{J}_P + \mathcal{J}_C, \]
where \( \mathcal{J}_P \) can be understood as a supervised loss on labeled data (implicit feedback), and \( \mathcal{J}_C \) corresponds to the unsupervised loss or regularization penalty applied to enforce smoothness among users/POIs and their context.

Neural Architecture. We present the neural architecture of PACE in Figure 2. Taking the input of users and POIs, we feed each labeled pair of them to a fully connected embedding layer \( E \), which can be seen as performing the latent factor modeling on users and POIs. Then we connect the user/POI embeddings to a context layer \( S \). The training of context predictions yielded by the two separate softmax layers in \( S \) is leveraged to preserve context information among users and POIs. We also merge the user and POI embeddings through vector concatenation and feed them to a preference layer \( H \) of multiple hidden layers of feed-forward neural networks, to deeply model the interactions among users and POIs. The training of preference prediction yielded by the softmax layer on the top of \( H \) is leveraged to learn user preference over POIs.

3.2 Learning Preference

As we discussed before, the first embedding layer \( E \) can be seen as performing the latent factor modeling for users and POIs. It learns two matrices \( E_u \) and \( E_p \), each row of which represents a user and a POI, respectively. As we use one-hot encodings \( u_i \) and \( p_j \) as input vectors, the final prediction of user \( u_i \)'s preference over POI \( p_j \) can be expressed as
\[ \hat{y}_{ij} = h(E_u^T u_i, E_p^T p_j | \Theta_e, \Theta_h), \]
where \( E_u \in \mathbb{R}^{N \times K_u}, E_p \in \mathbb{R}^{M \times K_p} \), denoting the latent factor matrices for users and POIs, \( \Theta_e \) denotes the parameters in the embedding layer, and \( \Theta_h \) denotes the parameters in the preference layer.

To deeply model the interactions among users and POIs, we merge \( E_u^T u_i \) and \( E_p^T p_j \) through vector concatenation, instead of element-wise product, because the latter cannot model non-linear interactions and requires the embeddings to be in the same space \((K_u = K_p)\). Then we feed \([E_u^T u_i, E_p^T p_j]\) to multiple hidden layers of feed-forward neural networks of \( H \). Given the input feature vector \( x \), the \( q \)-th hidden layer of \( H \) is denoted as \( h^q \), which is a non-linear function of the previous hidden layer \( h^{q-1} \) defined as
\[ h^q(x) = ReLU(W^q h^{q-1}(x) + b^q), \]
where \( W^q \) and \( b^q \) are parameters of the \( q \)-th layer, and \( h^0(x) = x = [E_u^T u_i, E_p^T p_j] \). We adopt the currently popular rectified linear unit \( ReLU(x) = \max(0, x) \) as the non-linear function.

Combining Eq. 4 and 5, we get
\[ \hat{y}_{ij} = h_{pred}(h^Q([E_u^T u_i, E_p^T p_j] \ldots)) = h_{pred}(H^Q([E_u^T u_i, E_p^T p_j])), \]
where \( Q \) is the total number of hidden layers. To specifically treat the one-class nature of implicit feedback in POI recommendations, we connect a binary softmax layer on the top of \( H \) as \( h_{pred} \), which is basically a logistic regression with sigmoid function, so we have
\[ \hat{y}_{ij} = \sigma(H^Q([E_u^T u_i, E_p^T p_j] \ldots) w_y), \]
where the sigmoid function is defined as \( \sigma(x) = 1/(1 + e^{-x}) \) and \( w_y \) are the parameters in the softmax layer.

To leverage supervision from implicit feedback and learn the parameters \( \Theta_e \) and \( \Theta_h \) for predicting user preference over POIs, we construct the following loss function, which is a special case of the commonly used cross entropy for softmax outputs. Therefore, we have our supervised loss in Eq. 3 as follows.
\[ \mathcal{J}_P = \log p(L | \Theta_e, \Theta_h) \]
\[ = - \sum_{(u_i, p_j) \in L^+} \log \hat{y}_{ij} - \sum_{(u_i, p_j) \in L^-} \log(1 - \hat{y}_{ij}) \]
\[ = - \sum_{(u_i, p_j) \in L} y_{ij} \log \hat{y}_{ij} + (1 - y_{ij}) \log(1 - \hat{y}_{ij}), \]
where \( L \) is the set of labeled pairs of users and POIs. Since we only have positive labels of observed interactions \( L^+ \) available in the data, we uniformly sample the negative labels from unobserved interactions \( L^- \) during training time and control the number of negative samples \( w.r.t. \) the number of observed instances. Other negative sampling scheme such as hard negative mining [10] can be applied to further improve convergence rate and performance, which we leave as a future work.

3.3 Preserving Context

The learning of \( E \) and \( H \) is essentially performing CF to model user preference over POIs based on past interactions. As we discussed in Sec. 1, in order to deal with data scarcity and various context in POI recommendations, we aim to further enable SSL together with CF, to enforce smoothness among similar users and POIs by regularizing the embeddings based on context graphs.

Context graphs construction. Graphs are powerful in representing various types of interactions and relations. In the literature on SSL, affinity graphs are widely used to encode distance among data, which can be computed from either internal data like node features [1, 58, 59] or external data like knowledge graphs [43] and citation networks [18]. The recent work [5] also shows promising results by leveraging unlabeled data from heterogeneous graphs.
In this work, we define context graphs as \textit{graphs that encode context information as affinity among instances}. We assume that most of the context associated with users and POIs can be built into such context graphs. E.g., the most important context associated with POIs is geographical information, which is often available as pairs of longitudes and latitudes, specifying the exact locations of POIs [24, 25, 28, 47]. Such information can be easily built into a POI graph \(G_p = (V_p, E_p)\), where \(V_p\) is the set of POIs, and \(E_p\) is the set of edges between nearby POIs. \(E_p\) can be further weighted by the inverse of geographical distance among POIs. On the other hand, social information such as friendships in the POI oriented APPs like Yelp and Foursquare is the most important context associated with users [22, 38, 52], which can be easily built into a user graph \(G_u = (V_u, E_u)\), where \(V_u\) is the set of users, and \(E_u\) is the set of edges among friends.

To accommodate other context like temporal, categorical and sequential information, we can construct heterogeneous context graphs augmented by time-of-day nodes, category nodes and sequence edges. To be more specific, each time-of-day node can connect to POI nodes belonging to or tagged with the corresponding category; each sequence edge can connect POIs that are frequently visited in a sequence.

In this work, we focus on developing the general neural architecture that extends CF into the SSL framework. To show the effectiveness of such a framework, we construct a simple POI graph with uniform edges among close POIs filtered by a specific radius \(r\) and a simple user graph with uniform edges among users that are friends. We leave the exploration of more complex weighted and heterogeneous context graphs to further boost recommendation performance as a future work.

\textbf{Context prediction as graph-based regularization.} We formulate our unsupervised loss in Eq. 3 as a regularization on the embeddings based on the context graphs. Recently, a number of embedding learning algorithms based on the Skipgram model [29] have been developed to predict context on graphs [11, 31, 46]. Given an instance and its context, the objective of Skipgram is to minimize the log loss of predicting the context using the embedding of the instance. Following the derivations in [31] and considering our situation of user context, we have

\[
J_{C_u} = - \sum_{(u, c_i)} \log p(u | c_i) \\
= - \sum_{(u, c_i)} \log (\phi_c^T E_u^T u_i - \log \sum_{u_c \in C_u} \exp(\phi_c^T E_u^T u_c)),
\]

where \(C_u\) is the set of all \(N\) possible user contexts, \(\Phi\) is the \(N\) sets of parameters in the Skipgram model, i.e., the softmax layer on the user context side, and \(E_u^T u_i\) as we discussed before is the embedding of user \(u_i\). Similarly, we have

\[
J_{C_p} = - \sum_{(p_i, c_j)} \log p(c_j | p_i) \\
= - \sum_{(p_i, c_j)} \log (\phi_c^T E_p^T p_j - \log \sum_{p_c \in C_p} \exp(\phi_c^T E_p^T p_c)),
\]

where \(C_p\) is the set of all \(M\) possible POI contexts, \(\Psi\) is the \(M\) sets of parameters in the softmax layer on the POI context side. The key insight of Skipgram is the prediction of context. E.g., by looking at the losses \(\log p(u | c_i)\) and \(\log p(c_j | p_i)\) on two users \(u_i\) and \(u_j\) that share the same context \(c_i\), we can see that the user embeddings \(E_u^T u_i\) and \(E_u^T u_j\) must be close in a certain way, because the two losses share the exact same form (the first part is parameterized by \(\phi_c\) and the second part by all other rows of \(\Phi\), and they are jointly minimized. Therefore, it is intuitive that minimizing the loss on all user-context pairs guarantees that users sharing more similar context will have closer embeddings. The situation is exactly the same for POIs.

As pointed out in [46], in terms of smoothing, compared with traditional GLR, graph embeddings are advantageous in producing useful features and fully leveraging the distributional information encoded in the graph structure. In the later part of this section (Sec. 3.4), we show that PACE actually generalize GLR in preserving context/neighborhood information on graphs.

\textbf{Context sampling on graphs.} To train the Skipgram models, we follow the popular negative sampling approach [29] to approximate the intractable normalization over the whole context spaces \(C_u\) and \(C_p\). In our case, we sample \((u_i, u_j, y)\) and \((p_j, p_c, \pi)\) similarly from two distributions, where \(y / \pi = +1\) means a positive pair, i.e., the user/POI is related to the context, and \(y / \pi = -1\) means negative. Given \((u_i, u_c, y)\), we minimize the cross entropy loss of classifying the pair \((u_i, u_c)\) to the binary label \(y\):

\[
\mathcal{J}_{C_u} = - \mathbb{E}_{(u_i, u_c, y)} \log \sigma(y \phi_c^T E_u^T u_i) = \mathbb{E}_{(u_i, u_c, y)} \log \sigma(-\phi_c^T E_u^T u_i),
\]

where \(\sigma\) is the sigmoid function as defined after Eq. 7, and \(\mathbb{E}\) is an indicator function that outputs 1 when the argument is true and otherwise 0.

Therefore, the unsupervised loss on the user context side with negative sampling can be written as

\[
\mathcal{J}_{C_u} = - \mathbb{E}_{(u_i, u_c, y)} \log \sigma(y \phi_c^T E_u^T u_i),
\]

where the expectation is taken w.r.t. the distribution \(p(u_i, u_c, y)\), which is conditioned on the user context graph and encodes the distributional information in the graph structure. We omit the derivation of \(\mathcal{J}_{C_p}\), i.e., the unsupervised loss on the POI context side, because it is similar to \(\mathcal{J}_{C_u}\).

In the following, we describe a sampling process that defines the distribution \(p(u_i, u_c, y)\), which follows [46] to handle real-valued edge weights in graphs and incorporate negative sampling. Due to the similarity, we also omit the sampling process of \((p_j, p_c, \pi)\).

On a user context graph \(A\), we first uniformly sample a random walk sequence \(S\). Specifically, we uniformly sample the first instance \(S_1\) from the set of all \(N\) users. Given the previous user \(S_{k-1} = u_i\), we then sample the next user \(S_k = u_j\) based on the probability \(a_{ij} / \sum_{j'=1}^N a_{ij'}\). With probability \(t\), we sample a positive pair \((u_i, u_c)\) from the set \(\{S_j, S_k : |j - k| < d\}\), where \(d\) is a window size parameter. With probability \((1 - t)\), we uniformly corrupt the context \(c\) to sample a negative pair.
3.4 Generalizing CF and SSL

We show that PACE is a natural bridge between CF and SSL, because it generalizes the \textit{de facto} methods of the two frameworks, i.e., MF for CF and GLR for graph-based SSL.

We first present the connections between PACE and MF. When taking the input as one-hot encodings \( u_i \) of user \( i \) and \( p_j \) of POI \( j \), PACE firstly computes two embeddings, \( E^u_{u} u_i \) for \( u_i \) and \( E^p_{p} p_j \) for \( p_j \). In the merging layer, rather than concatenating the two embedding vectors, we can compute a simple element-wise product of the two vectors (assuming we require the user and POI embeddings to be of the same size), i.e.,

\[
h^0(E^u_{u} u_i, E^p_{p} p_j) = E^u_{u} u_i \odot E^p_{p} p_j.
\]

(13)

Afterwards, instead of multiple hidden non-linear layers, we can directly project the vector to the output layer by summing up all elements and get the prediction:

\[
y_{ij} = h_{pred}(E^u_{u} u_i; E^p_{p} p_j) = \sum_{k=1}^{K} [E^u_{u} u_i k] [E^p_{p} p_j k].
\]

(14)

where \( K \) denotes the dimension of the latent factor space. In this way, we exactly recover the basic MF model.

While MF is an extremely simple version of PACE without considering any context, we use concatenation to merge the two embedding vectors instead of element-wise product, which incurs a large loss of information. We also insert multiple hidden non-linear layers before producing the final predictions, and apply a softmax prediction instead of element summation. Therefore, PACE is much more expressive than MF and can also generalize a large family of factorization models that can be seen as variations of MF.

We now study the connections between PACE and GLR. The core of context modeling in PACE is based on Skipgram with negative sampling [29], which is essentially factorizing a PMI (point-wise mutual information) matrix [21].

The PMI matrix is defined as

\[
M_{ij}^{\text{PMI}} = \phi_j^T E^u_{u} u_i = M_{ij}^{\text{PMI}} (u_i, c_j) - \log k = \log \frac{P(u_i, c_j)}{P(u_i)P(c_j)} - \log k,
\]

(15)

where \( k \) is the number of negative samples for each positive label, \( P(\cdot) \) is the probability of user-context pairs or users and contexts to be seen (sampled) on the graph. Similarly as before, we only show the analysis on the user context side, and the results generalize trivially to the POI context side.

In Eq. 15, as \( \phi_j \) and \( E^u_{u} u_i \) are of the same size, and \( M_{ij}^{\text{PMI}} \) is symmetric for \( u_i \) and \( c_j \), we can intuitively view \( \Phi \) as the embeddings for contexts and \( E_{u} \) as the embeddings for users, which are computed on the same set of \( N \) users.

For text embedding, the schemes of using words to predict contexts and using contexts to predict words lead to quite different embedding models [29]. This is basically due to the sampling and prediction schema, in which word-context pairs are sampled within a small window of size \( k \) sliding along the text, and in each window either the one target word is used to predict the \( (k - 1) \) context words or the other way round, which essentially leads to asymmetric predictions among words and contexts. However, in PACE, users and contexts are sampled interchangeably on the same random walk sequences, and each \( u_i \) is only used to predict one \( c_j \) in each sampled pair. Therefore, we can expect that \( \Phi \) and \( E_{u} \) should be similar, after sufficient amounts of sampling and training.

Based on this observation, the log loss \( \log \sigma(y \phi^T E^u_{u} u_i) \) on each positive user-context pair \((u_i, c_j, y = 1)\) can be viewed as incurring the following square loss under the Lagrange multiplier for the normalization constraints over all user-context pairs [3]:

\[
l(u_i, c_j, y = 1) = ||E^u_{u} u_i - \phi_j||^2 = ||E^T_{u} u_i - E^T_{u} u_i||^2.
\]

(16)

By setting the window size parameter \( d = 2 \) in the path sampling process on context graph \( A \), the square loss is only incurred on directly connected pairs of users on \( A \). Since the sampling is done w.r.t. edge weights in \( A \), Eq. 16 exactly recovers GLR in Eq. 1.

Therefore, PACE is a true SSL framework that generalizes GLR and preserves user/POI context on graphs, while it can be learned in neural networks jointly with a CF objective to reliably predict user preference over POIs.

3.5 Joint Training

We implement PACE with Tensorflow\textsuperscript{\textsuperscript{2}} to jointly train our model w.r.t. the multiple objectives in Eq. 2 by performing SGD (stochastic gradient descent) with mini-batch Adam [19]. We first sample a batch of labeled pairs of users and POIs from \( L \) and take a gradient step to optimize the supervised loss \( J_p \) for learning user preference over POIs. Next we sample a batch of user context \((u_i, c_e, y)\) and take take a gradient step to optimize the unsupervised loss \( J_{E_u} \) to preserve user context. Then we sample a batch of POI context \((p_j, p_c, \pi)\) and take another gradient step to optimize the unsupervised loss \( J_{E_p} \) to preserve POI context. We repeat the above procedures for \( T_0 \), \( T_1 \) and \( T_2 \) iterations respectively to approximate the weighting factors \( \lambda_1 \) and \( \lambda_2 \).

Algorithm 1 illustrates our SGD-based joint training algorithm for learning the PACE model. We update all model parameters \( \Theta = (\Theta_u, \Theta_h, \Theta_c) (\Theta_u = (\Phi, \Psi)) \) in iterations and stop when the overall log loss \( J \) converges or is sufficiently small. Our Tensorflow-based implementation runs efficiently on both CPU and GPU with minimum setups, while we experience 10+ times speedup on GPUs.

We also observe that mildly increasing the batch size \( B \) to several hundred generally permits better parallelization and faster computation as well as convergence, especially on GPU. An implementation of our framework can be found on Github\textsuperscript{\textsuperscript{8}}.

4 EXPERIMENTS

In this section, we evaluate PACE for POI recommendation with extensive experiments on real-world check-in datasets.

4.1 Experimental Settings

Datasets. We use two public datasets with user-POI interactions, friendships of users and locations of POIs. The basic statistics of the two datasets are shown in Table 1. Their details are introduced as follows.

The Gowalla check-in dataset [7] was generated world-wide from February 2009 to October 2010. We filter out the users with fewer than 15 check-in POIs and the POIs with fewer than 10 visitors, which is a common practice for POI recommendation [25, 26, 48].

\textsuperscript{2}https://www.tensorflow.org/

\textsuperscript{8}https://github.com/yangji9181/PACE2017
Algorithm 1 Joint Training of PACE

1: procedure PACETRAIN
2: Input: $A_u, A_p, U, P, T_0, T_1, T_2, B$
3: repeat
4:   for $t \leftarrow 1$ to $T_0$ do
5:     Sample a batch $B_0$ of $(u_i, p_j, y)$ of size $B$
6:     $J_p = \frac{1}{B} \sum_{(u_i, p_j, y) \in B_0} y \log \hat{y} + (1 - y) \log (1 - \hat{y})$
7:     Take a gradient step to optimize $J_p$
8:   end for
9:   for $t \leftarrow 1$ to $T_1$ do
10:     Sample a batch $B_1$ of $(u_i, u_c, y)$ of size $B$
11:     $J_u = \frac{1}{B} \sum_{(u_i, u_c, y) \in B_1} \log \sigma(\gamma \psi B^T \psi_i u_c)$
12:     Take a gradient step to optimize $J_u$
13:   end for
14:   for $t \leftarrow 1$ to $T_2$ do
15:     Sample a batch $B_2$ of $(p_j, p_c, y)$ of size $B$
16:     $J_p = \frac{1}{B} \sum_{(p_j, p_c, y) \in B_2} \log \sigma(\gamma \psi B^T \psi_j p_j)$
17:     Take a gradient step to optimize $J_p$
18:   end for
19: until $J$ converges or is sufficiently small
20: end procedure

<table>
<thead>
<tr>
<th>Dataset</th>
<th>#user</th>
<th>#POI</th>
<th>#check-in</th>
<th>Sparsity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gowalla</td>
<td>18,737</td>
<td>32,510</td>
<td>1,278,274</td>
<td>99.865%</td>
</tr>
<tr>
<td>Yelp</td>
<td>30,887</td>
<td>18,995</td>
<td>860,888</td>
<td>99.860%</td>
</tr>
</tbody>
</table>

Table 1: The statistics of the used data sets.

After filtered, each user visited an average of 43.87 POIs and each POI has an average of 25.21 visitors.

The Yelp check-in dataset is from the Yelp Dataset Challengeootnote{https://www.yelp.com/dataset_challenge} round 7 in 2016. We eliminate the users with fewer than 10 check-in POIs and the POIs with fewer than 10 visitors. After filtered, there are 26.58 check-ins per user and 43.21 visitors per POI on average.

We partition each dataset into training set and test set. For each user, we use the earliest 80% check-ins as the training data, and the most recent 20% check-ins as the test data.

Evaluation metrics. For performance comparison with baselines (Sec. 4.2), we apply four widely-used metrics, i.e., Pre@K (precision), Rec@K (recall), nDCG@K (normalized discounted cumulative gain), and MAP@K (mean average precision). The four metrics reflect different aspects of the results: precision and recall measure the number of correct recommendations in the result, while nDCG and MAP consider the rank of the recommendations.

When studying the performance of PACE under different parameter settings (Sec. 4.3), we need to evaluate the model many times even in the training process. Since it is too time-consuming to rank all POIs for every user at each time, we adopt the commonly used leave-one-out scheme [14]. Given the groundtruth POI for a user, we follow the strategy [9, 20] of mixing it with 100 random POIs that are not visited by the user. Then we rank the groundtruth along with the 100 items and measure HR@10 (hit ratio) for the result ranking list, which checks whether the test item is present on the top-10 list.

Compared algorithms. We compare PACE with the following POI recommendation algorithms:

- iReMF [28]: it models geographical influence by incorporating neighboring characteristics into weighted MF.
- LOCABAL [38]: it models two social relations: social friends and the users with high global reputations, in the MF framework.
- USG [47]: it combines geographical influence, social network and user interest with user-based CF and FCF (friend-based CF).
- iGSLR [52]: it exploits personalized geographical preference and social influence with FCF and KDE (kernel density estimation).
- IORE [54]: it considers sequential influence in addition to social and geographical influence by FCF, KDE and additive Markov chain.
- ASMF and ARMF [22]: an augmented square error based MF model and an augmented ranking error based MF model proposed in the same work to leverage individual’s different types of friends and potential location set.

Parameter settings. When comparing PACE with the baseline methods, we set its parameters to the following default values: for the loss function, we set the weighting factors $\lambda_1 = \lambda_2 = 0.1$; for the hidden layers in $H$, we set the number of layers to 3 and the sizes of the layers to 32 $\rightarrow$ 16 $\rightarrow$ 8; for the context sampling process related to $S$, we set the path length $|S|$ to 10, window size $d$ to 3; for the embedding layer in $E$, we set the embedding size to 16; for the joint training process, we sample 5 negative samples for each positive label, and we set the batch size to 512 and the learning rate to 0.0001. In addition to these default values, we also evaluate the effects of different parameters on the performance of PACE in Sec. 4.3. For all baselines, since we experiment on the standard Gowalla and Yelp datasets and use the common evaluation metrics and preprocessing techniques, we set their parameters to the values suggested in their original works or implementations.

4.2 Performance Comparison with Baselines

We quantitatively evaluate PACE against all baselines on standard POI recommendation. Figure 3 shows the performance under the four metrics on the two datasets. The improvements of PACE over the compared algorithms all passed the paired t-tests with significance value $p < 0.01$.

The relative performance among the baselines varies across different data sets and metrics. However, compared with the strongest baseline, PACE constantly yields around 8.5% relative improvements on the Gowalla dataset and more than 6% improvements on the Yelp dataset in all of the four metrics. Such consistent improvements clearly demonstrate the advantage of PACE over the baseline methods. The neural embedding part can accurately model the interactions between the users and places through the implicit feedback data; while the SSL framework effectively incorporates the context knowledge from the vast unlabeled data.

Taking a closer look at the scores, we observe that the scores on the Gowalla dataset are generally better than those on the Yelp datasets. We attribute this to the fact that users have richer check-in history in the former dataset. The improvements made by PACE are also more significant on the Gowalla dataset, indicating its power in learning user preferences from historical interactions.
Among all compared methods, IRenMF only considers geographical influence while LOCABAL only considers social influence. On the Gowalla dataset where user history data are rich, exploiting POI context helps a lot in improving the overall performance — IRenMF outperforms LOCABAL by about 72% on Gowalla. For the Yelp dataset, on the other hand, leveraging user context significantly alleviates the scarcity of user history — IRenMF only performs about 28% relatively better than LOCABAL on Yelp. Such a contrast indicates the varying significance of different context on different datasets, which further proves the robustness of PACE in jointly leveraging the context on both the user and POI sides.

### 4.3 Parameter Study

Now we study the effects of different parameters on performance of PACE, through an in-depth analysis of its key components under various parameter settings.

<table>
<thead>
<tr>
<th>Capacity</th>
<th>Q = 0</th>
<th>Q = 1</th>
<th>Q = 2</th>
<th>Q = 3</th>
<th>Q = 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gowalla dataset</td>
<td>4</td>
<td>0.565</td>
<td>0.813</td>
<td>0.818</td>
<td>0.824</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>0.566</td>
<td>0.825</td>
<td>0.843</td>
<td>0.852</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>0.566</td>
<td>0.834</td>
<td>0.850</td>
<td>0.859</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>0.564</td>
<td>0.838</td>
<td>0.857</td>
<td>0.864</td>
</tr>
<tr>
<td>Yelp dataset</td>
<td>4</td>
<td>0.425</td>
<td>0.587</td>
<td>0.601</td>
<td>0.612</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>0.427</td>
<td>0.603</td>
<td>0.614</td>
<td>0.626</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>0.425</td>
<td>0.618</td>
<td>0.629</td>
<td>0.634</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>0.420</td>
<td>0.652</td>
<td>0.656</td>
<td>0.641</td>
</tr>
</tbody>
</table>

Table 2: HR@10 with different architectures of $\mathcal{H}$.

**Benefit of neural architectures.** Since there is no previous work on POI recommendations that uses neural networks to model user-POI interactions, we are interested in how the deep architectures can benefit the modeling of user-POI interactions. To this end, we study the effectiveness of our preference learning module $\mathcal{H}$ by varying the number of hidden layers as well as the capacity of each hidden layer. We vary the size of the last hidden layer and shrink the size of the hidden layer by half upwards. E.g., if the model capacity is 8 and the number of hidden layers is 3, the sizes of layers are $32 \rightarrow 16 \rightarrow 8$, and the size of user/POI embeddings is 16. The other model parameters are set to their defaults as described in Sec. 4.1.

Table 2 shows the model performance with different neural architectures. While using one non-linear layer boosts the performance significantly, stacking more layers and increasing the model capacity generally leads to slightly better performance (but requires significantly more training time). The results are nonetheless encouraging, which indicate the effectiveness of using the expressive neural networks for modeling user preference over POIs.

**Effectiveness of context smoothing.** To simultaneously learn user preference over POIs and preserve user/POI context, we jointly train our PACE model w.r.t. the three objectives in Eq. 2. It is interesting to see the impact of context smoothing and the effectiveness of joint training. Therefore, we compare the training loss (averaged over all instances on preference prediction) and prediction performance curves of four models: PACE-H (preference learning only), PACE-HU (preference with user context smoothing), PACE-HP (preference with POI context) and PACE-HUP (preference with user and POI context). The model parameters are set to default as mentioned in Sec. 4.1.
Figure 4 shows the results on Gowalla data. The results on Yelp data show similar trends and thus we omit them due to space limit. First, when the number of epochs increases, the training loss generally decreases and the prediction performance improves. The effects the first several epochs are especially significant. After a few epochs, the loss keeps decreasing slowly, while the performance slightly fluctuates. Comparing the variants of PACE, we see that PACE-H achieves the lowest loss and converges fastest, followed by PACE-HP, PACE-HU and finally PACE-HUP. However, the performance shows a contrary order as PACE-HUP > PACE-HP > PACE-HU > PACE-H. Finally, slight overfitting is observed only for PACE-H, but not the other variants. These results clearly demonstrate that applying context smoothing (on both user and POI sides) and joint training leads to better model performance and robustness.

Note that the prediction performance of PACE usually converges fast after the first few epochs. In our experiments, we also find that when we use two hidden layers, the training time for PACE in one epoch on CPU is comparable to the runtime of most baselines. Therefore, it is possible to stop the iterations early and deliver satisfactory predictions, while it is also promising to further speed up PACE by trivially deploying it on GPU.

5 CONCLUSION

In this paper, we develop a neural architecture called PACE to bridge CF and SSL for POI recommendation. On the one hand, we leverage the expressive neural networks to model non-linear complex interactions between users and POIs. On the other hand, we exploit context graphs and apply user/POI smoothing to address data scarcity and various context. Our extensive experiments show the effectiveness of the proposed approach.

As one limitation of the current framework, it only considers static homogeneous context graphs with uniform edges. For future work, it is promising to explore weighted and heterogeneous context graphs for including even richer context information, which may bring extra performance gain for PACE. As new users/POIs and their connections are added to the network, it is useful to incrementally update the model by training on the relevant set of preference feedbacks and sampled context paths. In addition, the framework of PACE is general, and it is feasible to leverage other neural networks such as LSTM and tackle tasks like online POI recommendation.

ACKNOWLEDGEMENT

This research was sponsored in part by the U.S. Army Research Laboratory under Cooperative Agreement No. W911NF-09-2-0053 (NSCTA), National Science Foundation IIS-1320617 and IIS 16-18481, and grant 1U54GM114838 awarded by NIGMS through funds provided by the trans-NIH Big Data to Knowledge (BD2K) initiative (www.bd2k.nih.gov). The views and conclusions contained in this document are those of the author(s) and should not be interpreted as representing the official policies of the U.S. Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

REFERENCES


[18] Ming Ji, Yizhou Sun, Marina Danilevsky, Jiawei Han, and Jing Gao. 2010. Graph regularized transductive classification on heterogeneous information networks. In Joint European Conference on Machine Learning and Knowledge Discovery in