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

ACM Reference format:
Carl Yang, Lanxiao Bai, Chao Zhang, Quan Yuan and Jiawei Han. 2017. Bridging Collaborative Filtering and Semi-Supervised Learning: A Neural Approach for POI Recommendation. In *Proceedings of ACM SIGKDD conference, Halifax, Nova Scotia - Canada, August 2017 (KDD'17),* 9 pages.
DOI: 10.475/123_4

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[1] 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
[2]http://www.kdd.org/kdd-cup/view/kdd-cup-2007

With the prominence of location-aware social media, such as Yelp[3], Foursquare[4], and Facebook Places[5], people can easily share content associated with locations. *E.g.*, Foursquare alone has collected more than 10 billion check-ins to POIs (point of interests) so far and has about 50 million active users in 2016[6]. Such vast amount of check-in data give rise to a specifically useful type of recommender systems, *i.e.*, POI recommendation. By helping users explore new POIs, it greatly benefits both users and businesses in real life.

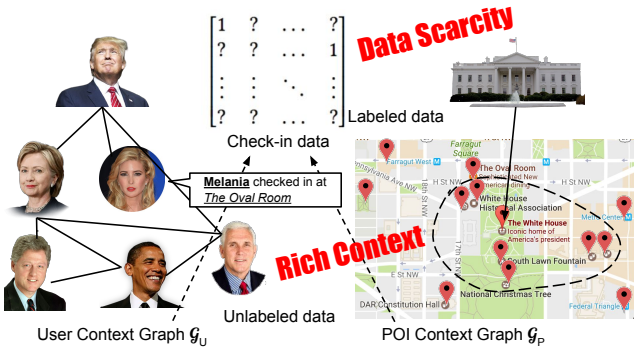


Figure 1: An illustration of the challenges in real-world POI recommendation and the key leverage of our approach.

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% [22, 38], 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 [22, 23].

Challenge 2: Various context. While traditional recommender systems basically deal with ratings and reviews, the context for

[3]https://www.yelp.com
[4]https://foursquare.com
[5]https://www.facebook.com/directory/places
[6]https://foursquare.com/about

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 [40, 43–45]. 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 $\mathcal{G}_U = \{\mathcal{V}_U, \mathcal{E}_U\}$, where $\mathcal{V}_U$ is the set of users, and $\mathcal{E}_U$ is the set of edges among friends; geographical distance among POIs can be naturally built into a graph $\mathcal{G}_P = \{\mathcal{V}_P, \mathcal{E}_P\}$, where $\mathcal{V}_P$ is the set of POIs, and $\mathcal{E}_P$ 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 $\mathcal{G}_U$ and $\mathcal{G}_P$ 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 [14], 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 [27, 33, 39]. 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 [39] 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, 27, 33], 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.

2 RELATED WORK

2.1 Modeling Implicit Feedback

To model user preference, while traditional recommender systems have largely focused on explicit ratings [28, 29], POI recommendations usually deal with implicit feedback, which is more practical but challenging [13, 24]. 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 [22, 23]. 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 [13] or sample negative instances from all unobserved entries [24].

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 [26, 39], 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 [18], POI recommendations naturally come with various context, such as geographical information [22, 23, 25, 40], categorical information [44], social information [20, 32, 43], temporal information [41] and *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 [32].

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, 36, 46, 47], or derived from external data, such as knowledge graphs [37] or citation networks [5, 16]. 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, 46, 47]. Their general loss function can be written as

$$\begin{aligned} &\sum_{i \in \mathcal{L}} l(y_i, f(\mathbf{x}_i)) + \lambda \sum_{i,j} a_{ij} ||f(\mathbf{x}_i) - f(\mathbf{x}_j)||^2 \\ &= \sum_{i \in \mathcal{L}} l(y_i, f(\mathbf{x}_i)) + \lambda \mathbf{f}^T L \mathbf{f}, \end{aligned} \tag{1}$$

where $f(\cdot)$ is the prediction function to be learned that maps node features $\mathbf{x}$ 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. L is the graph Laplacian matrix defined as $L = A - D$, where A is the affinity graph in matrix form, and D is the diagonal matrix with $d_{ii} = \sum_j a_{ij}$. It requires the predictions to be close on nearby nodes. λ is a constant weighting factor.

To leverage SSL for modeling user preference over POIs, we follow the semi-supervised embedding framework proposed by [36], which extends the regularization term in Eq. 1 into $\sum_{i,j} a_{ij} ||\mathbf{g}(\mathbf{x}_i) - \mathbf{g}(\mathbf{x}_j)||^2$, where $\mathbf{g}$ 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 [28], which discretizes user ratings and models the hidden features of users and items underlying the ratings. Recently, autoencoders have also been successfully adopted for learning user representations based on rated items [21, 30]. 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 [34], acoustic features of musics [35], cross-domain behaviors of users [9] and the information in knowledge bases [42]. 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, 31]. 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 [27] and Skipgram [26] 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 [15, 18]. 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 $\mathbf{u}_i$ and $\mathbf{p}_j$ 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.

$$\mathcal{J} = \mathcal{J}_P + \lambda_1 \mathcal{J}_{C_u} + \lambda_2 \mathcal{J}_{C_p}, \tag{2}$$

where $\mathcal{J}_P$ is the loss on preference and $\mathcal{J}_{C_u} + \mathcal{J}_{C_p}$ is the loss on context. λ's control the trade-off among the three objectives. Eq. 2 can be further simplified to

$$\mathcal{J} = \mathcal{J}_P + \mathcal{J}_C, \tag{3}$$

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 $\mathcal{E}$, 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 $\mathcal{S}$. The training of context predictions yielded by the two separate softmax layers in $\mathcal{S}$ 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 $\mathcal{H}$ 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 $\mathcal{H}$ is leveraged to learn user preference over POIs.

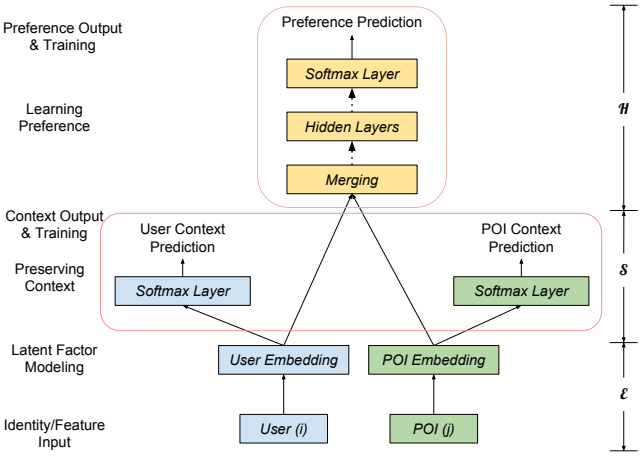


Figure 2: The neural architecture of PACE.

3.2 Learning Preference

As we discussed before, the first embedding layer $\mathcal{E}$ can be seen as performing the latent factor modeling for users and POIs. It learns two matrices $\mathbf{E}_u$ and $\mathbf{E}_p$, each row of which represents a user and a POI, respectively. As we use one-hot encodings $\mathbf{u}_i$ and $\mathbf{p}_j$ as input vectors, the final prediction of user u_i's preference over POI p_j can be expressed as

$$\hat{y}_{ij} = h(\mathbf{E}_u^T \mathbf{u}_i, \mathbf{E}_p^T \mathbf{p}_i | \Theta_e, \Theta_h), \tag{4}$$

where $\mathbf{E}_u \in \mathbb{R}^{N \times K_u}$, $\mathbf{E}_p \in \mathbb{R}^{M \times K_p}$, denoting the latent factor matrices for users and POIs. Θ_e denotes the parameters in the embedding layer, and Θ_h denotes the parameters in the preference layer.

To deeply model the interactions among users and POIs, we merge $\mathbf{E}_u^T \mathbf{u}_i$ and $\mathbf{E}_p^T \mathbf{p}_i$ 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 $[\mathbf{E}_u^T \mathbf{u}_i, \mathbf{E}_p^T \mathbf{p}_i]$ to multiple hidden layers of feed-forward neural networks of $\mathcal{H}$. Given the input feature vector $\mathbf{x}$, the q-th hidden layer of $\mathcal{H}$ is denoted as $\mathbf{h}^q$, which is a non-linear function of the previous hidden layer $\mathbf{h}^{q-1}$ defined as

$$\mathbf{h}^q(\mathbf{x}) = ReLU(\mathbf{W}^q \mathbf{h}^{q-1}(\mathbf{x}) + b^q), \tag{5}$$

where $\mathbf{W}^q$ and b^q are parameters of the q-th layer, and $\mathbf{h}^0(\mathbf{x}) = \mathbf{x} = [\mathbf{E}_u^T \mathbf{u}_i, \mathbf{E}_p^T \mathbf{p}_i]$. 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

$$\begin{aligned} \hat{y}_{ij} &= h_{pred}(\mathbf{h}^Q(\ldots \mathbf{h}^1(\mathbf{h}^0([\mathbf{E}_u^T \mathbf{u}_i, \mathbf{E}_p^T \mathbf{p}_i])) \ldots)) \\ &= h_{pred}(\mathbf{H}^Q([\mathbf{E}_u^T \mathbf{u}_i, \mathbf{E}_p^T \mathbf{p}_i])), \end{aligned} \tag{6}$$

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 $\mathcal{H}$ as h_{pred}, which is basically a logistic regression with sigmoid function, so we have

$$\hat{y}_{ij} = \sigma(\mathbf{H}^Q([\mathbf{E}_u^T \mathbf{u}_i, \mathbf{E}_p^T \mathbf{p}_i])^T \mathbf{w}_y), \tag{7}$$

where the sigmoid function is defined as $\sigma(x) = 1/(1 + e^{-x})$ and $\mathbf{w}_y$ are the parameters in the softmax layer.

To leverage supervision from implicit feedback and learn the parameters Θ_e and Θ_h for predicting user preference over POIs, we construct the following *log 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.

$$\begin{aligned} \mathcal{J}_P &= \log p(\mathcal{L} | \Theta_e, \Theta_h) \\ &= - \sum_{(u_i, p_j) \in \mathcal{L}^+} \log \hat{y}_{ij} - \sum_{(u_i, p_j) \in \mathcal{L}^-} \log(1 - \hat{y}_{ij}) \\ &= - \sum_{(u_i, p_j) \in \mathcal{L}} y_{ij} \log \hat{y}_{ij} + (1 - y_{ij}) \log(1 - \hat{y}_{ij}), \end{aligned} \tag{8}$$

where $\mathcal{L}$ is the set of labeled pairs of users and POIs. Since we only have positive labels of observed interactions $\mathcal{L}^+$ available in the data, we uniformly sample the negative labels from unobserved interactions $\mathcal{L}^-$ 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 $\mathcal{E}$ and $\mathcal{H}$ 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, 46, 47] or external data like knowledge graphs [37] and citation networks [16]. The recent work [5] also shows promising results by leveraging unlabeled data from heterogeneous graphs.

In this work, we define context graphs as *graphs that encode context information as affinity among instances.* We assume that most of the various 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 [22, 23, 25, 40]. Such information can be easily built into a POI graph $\mathcal{G}_P = \{\mathcal{V}_P, \mathcal{E}_P\}$, where $\mathcal{V}_P$ is the set of POIs, and $\mathcal{E}_P$ is the set of edges between nearby POIs. $\mathcal{E}_P$ 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 [20, 32, 43], which can be easily built into a user graph $\mathcal{G}_U = \{\mathcal{V}_U, \mathcal{E}_U\}$, where $\mathcal{V}_U$ is the set of users, and $\mathcal{E}_U$ 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 all POI nodes that receive most check-ins during the corresponding time period of the day; each category 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.

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 [26] have been developed to predict context on graphs [11, 27, 39]. 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 [27] and considering our situation of user context, we have

$$\begin{aligned}\mathcal{J}_{C_u} &= -\sum_{(u_i,u_c)} \log p(u_c|u_i) \\ &= -\sum_{(u_i,u_c)} \log(\phi_c^T \mathbf{E}_u^T \mathbf{u}_i - \log \sum_{u'_c \in C_u} \exp(\phi_{c'}^T \mathbf{E}_u^T \mathbf{u}_i)), \end{aligned} \tag{9}$$

where C_u is the set of all N possible user contexts, Φ is the N sets of parameters in the Skipgram model, *i.e.*, the softmax layer on the user context side, and $\mathbf{E}_u^T \mathbf{u}_i$ as we discussed before is the embedding of user u_i. Similarly, we have

$$\begin{aligned}\mathcal{J}_{C_p} &= -\sum_{(p_j,p_c)} \log p(p_c|p_j) \\ &= -\sum_{(p_j,p_c)} \log(\psi_c^T \mathbf{E}_p^T \mathbf{p}_j - \log \sum_{p'_c \in C_p} \exp(\psi_{c'}^T \mathbf{E}_p^T \mathbf{p}_j)), \end{aligned} \tag{10}$$

where C_p is the set of all M possible POI contexts, Ψ 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 $\log p(u_c|u_i)$ and $\log p(u_c|u_j)$ on two users u_i and u_j that share the same context u_c, we can see that the user embeddings $\mathbf{E}_u^T \mathbf{u}_i$ and $\mathbf{E}_u^T \mathbf{u}_j$ must be close in a certain way, because the two losses share the exact same form (the first part is parameterized by ϕ_c and the second part by all other rows of Φ), 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 [39], 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.

Context sampling on graphs. To train the Skipgram models, we follow the popular negative sampling approach [26] to approximate the intractable normalization over the whole context spaces C_u and C_p. In our case, we sample (u_i, u_c, γ) and (p_j, p_c, π) similarly from two distributions, where $\gamma/\pi = +1$ means a positive pair, *i.e.*, the user/POI is related to the context, and $\gamma/\pi = -1$ means negative. Given (u_i, u_c, γ), we minimize the cross entropy loss of classifying the pair (u_i, u_c) to the binary label γ:

$$\mathcal{J}_{C_u} = -\mathbb{I}(\gamma = 1) \log \sigma(\phi_c^T \mathbf{E}_u^T \mathbf{u}_i) - \mathbb{I}(\gamma = -1) \log \sigma(-\phi_c^T \mathbf{E}_u^T \mathbf{u}_i), \tag{11}$$

where σ is the sigmoid function as defined after Eq. 7, and $\mathbb{I}(\cdot)$ 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

$$\mathcal{J}_{C_u} = -\mathbb{E}_{(u_i,u_c,\gamma)} \log \sigma(\gamma \phi_c^T \mathbf{E}_u^T \mathbf{u}_i), \tag{12}$$

where the expectation is taken *w.r.t.* the distribution $p(u_i, u_c, \gamma)$, 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 $p(u_i, u_c, \gamma)$, which follows [39] 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, π).

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 *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 $\mathbf{u}_i$ of user u_i and $\mathbf{p}_j$ of POI p_j, PACE firstly computes two embeddings, $\mathbf{E}_u^T \mathbf{u}_i$ for u_i and $\mathbf{E}_p^T \mathbf{p}_j$ 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.*,

$$\mathbf{h}^0(\mathbf{E}_u^T \mathbf{u}_i, \mathbf{E}_p^T \mathbf{p}_j) = \mathbf{E}_u^T \mathbf{u}_i \odot \mathbf{E}_p^T \mathbf{p}_j. \tag{13}$$

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:

$$\hat{y}_{ij} = h_{pred}(\mathbf{E}_u^T \mathbf{u}_i \odot \mathbf{E}_p^T \mathbf{p}_j) = \sum_{k=1}^{K} [\mathbf{E}_u^T \mathbf{u}_i]_k [\mathbf{E}_p^T \mathbf{p}_j]_k. \tag{14}$$

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 [26], which is essentially factorizing a PMI (point-wise mutual information) matrix [19], which is defined as

$$M_{ij}^{SGNS} = \phi_j^T \mathbf{E}_u^T \mathbf{u}_i = M_{ij}^{PMI}(u_i, c_j) - \log k = \log \frac{P(u_i, c_j)}{P(u_i)P(c_j)} - \log k, \tag{15}$$

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 ϕ_j and $\mathbf{E}_u^T \mathbf{u}_i$ are of the same size, and M_{ij}^{SGNS} is symmetric for u_i and c_j, we can intuitively view Φ as the embeddings for contexts and $\mathbf{E}_u$ 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 [26]. 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 Φ and $\mathbf{E}_u$ should be similar, after sufficient amounts of sampling and training,

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

$$l(u_i, c_j, \gamma = 1) = ||\mathbf{E}_u^T \mathbf{u}_i - \phi_j||^2 = ||\mathbf{E}_u^T \mathbf{u}_i - \mathbf{E}_u^T \mathbf{u}_j||^2. \tag{16}$$

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[7] to jointly train our model *w.r.t.* the multiple objectives in Eq. 2 by performing SGD (stochastic gradient descent) with mini-batch Adam [17]. We first sample a batch of labeled pairs of users and POIs from $\mathcal{L}$ and take a gradient step to optimize the supervised loss $\mathcal{J}_P$ for learning user preference over POIs. Next we sample a batch of user context (u_i, u_c, γ) and take take a gradient step to optimize the unsupervised loss $\mathcal{J}_{C_u}$ to preserve user context. Then we sample a batch of POI context (p_j, p_c, π) and take take another gradient step to optimize the unsupervised loss $\mathcal{J}_{C_p}$ to preserve POI context. We repeat the above procedures for T_0, T_1 and T_2 iterations respectively to approximate the weighting factors λ_1 and λ_2.

Algorithm 1 illustrates our SGD-based joint training algorithm for learning the PACE model. We update all model parameters $\Theta = (\Theta_e, \Theta_h, \Theta_s)$ $(\Theta_s = (\Phi, \Psi))$ in iterations and stop when the overall log loss $\mathcal{J}$ converges or is sufficiently small. Our Tensorflow-based implementation runs efficiently on both CPU and GPU with minimum setups, while we experience about 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. We will make the code available upon acceptance of the work.

Algorithm 1 Joint Training of PACE

procedure PACETRAIN
  Input: $A_u, A_p, \mathbf{U}, \mathbf{P}, T_0, T_1, T_2, B$
  repeat
    for $t \leftarrow 1$ to T_0 **do**
      Sample a batch $\mathcal{B}_0$ of (u_i, p_j, y) of size B
      $\mathcal{J}_P = \frac{1}{B} \sum_{(u_i, p_j, y) \in \mathcal{B}_0} y \log \hat{y} + (1-y) \log(1-\hat{y})$
      Take a gradient step to optimize $\mathcal{J}_P$
    end for
    for $t \leftarrow 1$ to T_1 **do**
      Sample a batch $\mathcal{B}_1$ of (u_i, u_c, γ) of size B
      $\mathcal{J}_{C_u} = \frac{1}{B} \sum_{(u_i, u_c, \gamma) \in \mathcal{B}_1} \log \sigma(\gamma \phi_c^T \mathbf{E}_u^T \mathbf{u}_i)$
      Take a gradient step to optimize $\mathcal{J}_{C_u}$
    end for
    for $t \leftarrow 1$ to T_2 **do**
      Sample a batch $\mathcal{B}_2$ of (p_j, p_c, γ) of size B
      $\mathcal{J}_{C_p} = \frac{1}{B} \sum_{(p_j, p_c, \gamma) \in \mathcal{B}_2} \log \sigma(\gamma \psi_c^T \mathbf{E}_p^T \mathbf{p}_j)$
      Take a gradient step to optimize $\mathcal{J}_{C_p}$
    end for
  until $\mathcal{J}$ converges or is sufficiently small
end procedure

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.

Dataset	#user	#POI	#check-in	Sparsity
Gowalla	18,737	32,510	1,278,274	99.865%
Yelp	30,887	18,995	860,888	99.860%

Table 1: The statistics of the used data sets.

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,

[7] https://www.tensorflow.org/

which is a common practice for POI recommendation [23, 24, 41]. After filtered, each user visited an average of 43.87 POIs and each POI has an average of 25.25 visitors.

The Yelp check-in dataset is from the Yelp Dataset Challenge[8] 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 *leave-one-out* scheme [13]. Given the groundtruth POI for a user, we follow the strategy [9, 18] 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:

- **IRenMF** [25]: it models geographical influence by incorporating neighboring characteristics into weighted MF.
- **LOCABAL** [32]: it models two social relations: social friends and the users with high global reputations, in the MF framework.
- **USG** [40]: it combines geographical influence, social network and user interest with user-based CF and FCF (friend-based CF).
- **iGSLR** [43]: it exploits personalized geographical preference and social influence with FCF and KDE (kernel density estimation).
- **LORE** [45]: it considers sequential influence in addition to social and geographical influence by FCF, KDE and additive Markov chain.
- **ASMF** and **ARMF** [20]: 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 $\mathcal{H}$, 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 $\mathcal{S}$, we set the path length $|S|$ to 10, window size d to 3; for the embedding layer in $\mathcal{E}$, 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, we set their parameters to the values suggested in the original works or implementations.

[8] https://www.yelp.com/dataset_challenge

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.

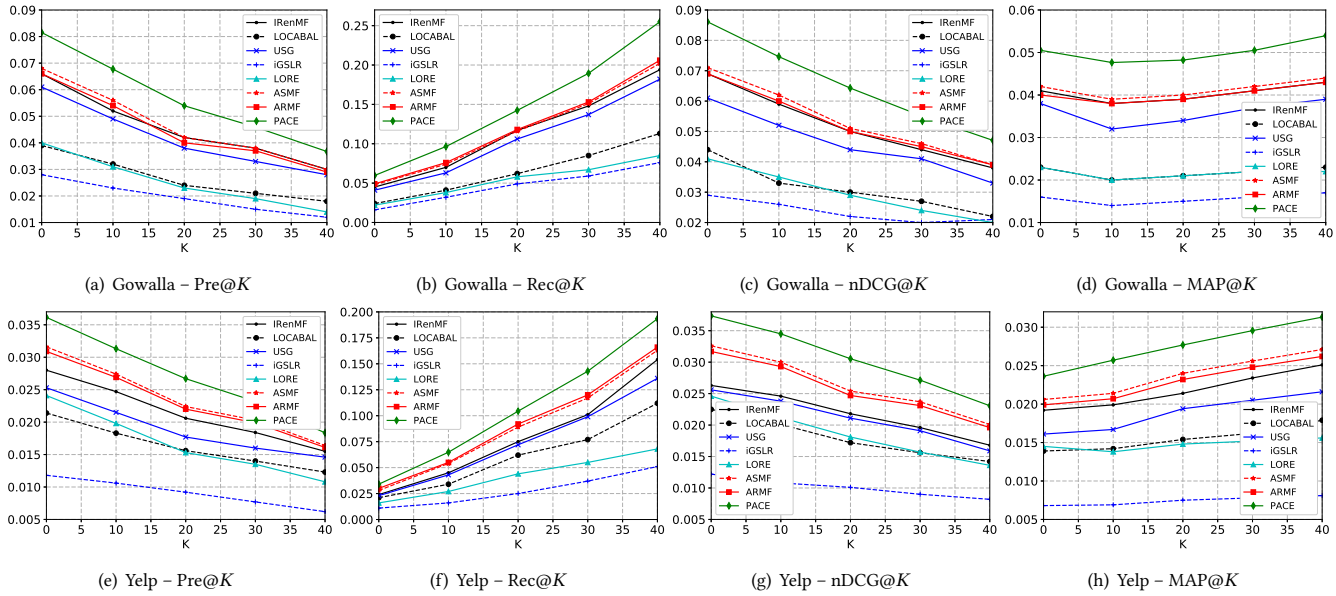
4.3 Parameter Study

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.

Capacity	Q = 0	Q = 1	Q = 2	Q = 3	Q = 4
Gowalla dataset					
4	0.565	0.813	0.818	0.824	**0.829**
8	*0.566*	0.825	0.843	0.852	**0.868**
16	*0.566*	0.834	0.850	0.859	**0.869**
32	0.564	*0.838*	*0.857*	*0.864*	***0.871***
Yelp dataset					
4	0.425	0.587	0.601	0.612	**0.619**
8	*0.427*	0.603	0.614	0.626	**0.628**
16	0.425	0.618	0.629	0.634	**0.635**
32	0.426	*0.632*	*0.636*	***0.641***	*0.640*

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



(a) Gowalla – Pre@K (b) Gowalla – Rec@K (c) Gowalla – nDCG@K (d) Gowalla – MAP@K

(e) Yelp – Pre@K (f) Yelp – Rec@K (g) Yelp – nDCG@K (h) Yelp – MAP@K

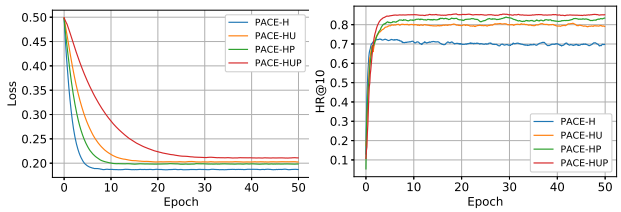
Figure 3: Performance compared with the baselines on the two datasets as K increases.

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 are 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 applying context smoothing (on both user and POI sides) and joint training leads to better model performance and robustness.



(a) Training Loss (b) HR@10

Figure 4: Training loss and prediction performance curves.

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 our approach. 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. In addition, as the framework of PACE is general, it is feasible to leverage other neural networks such as LSTM and tackle tasks like online POI recommendation.

REFERENCES

[1] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. 2006. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of machine learning research* 7, Nov (2006), 2399–2434.

[2] Robert M Bell and Yehuda Koren. 2007. Lessons from the Netflix prize challenge. *Acm Sigkdd Explorations Newsletter* 9, 2 (2007), 75–79.

[3] Dimitri P Bertsekas. 2014. *Constrained optimization and Lagrange multiplier methods*. Academic press.

[4] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*. 2787–2795.

[5] Ting Chen and Yizhou Sun. 2017. Task-Guided and Path-Augmented Heterogeneous Network Embedding for Author Identification. In *WSDM*. ACM.

[6] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, and others. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*. ACM, 7–10.

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

[8] Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*. ACM, 160–167.

[9] Ali Mamdouh Elkahky, Yang Song, and Xiaodong He. 2015. A multi-view deep learning approach for cross domain user modeling in recommendation systems. In *Proceedings of the 24th International Conference on World Wide Web*. ACM, 278–288.

[10] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. 2010. Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence* 32, 9 (2010), 1627–1645.

[11] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 855–864.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 770–778.

[13] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. 2016. Fast matrix factorization for online recommendation with implicit feedback. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 549–558.

[14] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, and others. 2012. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine* 29, 6 (2012), 82–97.

[15] Liangjie Hong, Aziz S Doumith, and Brian D Davison. 2013. Co-factorization machines: modeling user interests and predicting individual decisions in twitter. In *Proceedings of the sixth ACM international conference on Web search and data mining*. ACM, 557–566.

[16] 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 Databases*. Springer, 570–586.

[17] D Kinga and J Ba Adam. 2015. A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.

[18] Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 426–434.

[19] Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*. 2177–2185.

[20] Huayu Li, Yong Ge, and Hengshu Zhu. 2016. Point-of-Interest Recommendations: Learning Potential Check-ins from Friends. In *Proceedings of the 22th ACM SIGKDD international conference on on Knowledge discovery and data mining. ACM*.

[21] Sheng Li, Jaya Kawale, and Yun Fu. 2015. Deep collaborative filtering via marginalized denoising auto-encoder. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM, 811–820.

[22] Xutao Li, Gao Cong, Xiao-Li Li, Tuan-Anh Nguyen Pham, and Shonali Krishnaswamy. 2015. Rank-geofm: A ranking based geographical factorization method for point of interest recommendation. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 433–442.

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

[24] Dawen Liang, Laurent Charlin, James McInerney, and David M Blei. 2016. Modeling user exposure in recommendation. In *Proceedings of the 25th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 951–961.

[25] Yong Liu, Wei Wei, Aixin Sun, and Chunyan Miao. 2014. Exploiting geographical neighborhood characteristics for location recommendation. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. ACM, 739–748.

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

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

[28] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. 2007. Restricted Boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on Machine learning*. ACM, 791–798.

[29] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*. ACM, 285–295.

[30] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. 2015. Autorec: Autoencoders meet collaborative filtering. In *Proceedings of the 24th International Conference on World Wide Web*. ACM, 111–112.

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

[32] Jiliang Tang, Xia Hu, Huiji Gao, and Huan Liu. 2013. Exploiting Local and Global Social Context for Recommendation.. In *IJCAI*. 264–269.

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

[34] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative deep learning for recommender systems. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1235–1244.

[35] Xinxi Wang and Ye Wang. 2014. Improving content-based and hybrid music recommendation using deep learning. In *Proceedings of the 22nd ACM international conference on Multimedia*. ACM, 627–636.

[36] Jason Weston, Frédéric Ratle, Hossein Mobahi, and Ronan Collobert. 2012. Deep learning via semi-supervised embedding. In *Neural Networks: Tricks of the Trade*. Springer, 639–655.

[37] Derry Wijaya, Partha Pratim Talukdar, and Tom Mitchell. 2013. Pidgin: ontology alignment using web text as interlingua. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. ACM, 589–598.

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

[39] Zhilin Yang, William Cohen, and Ruslan Salakhutdinov. 2016. Revisiting semi-supervised learning with graph embeddings. In *ICML*.

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

[41] Quan Yuan, Gao Cong, Zongyang Ma, Aixin Sun, and Nadia Magnenat Thalmann. 2013. Time-aware point-of-interest recommendation. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*. ACM, 363–372.

[42] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative knowledge base embedding for recommender systems. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 353–362.

[43] Jia-Dong Zhang and Chi-Yin Chow. 2013. iGSLR: personalized geo-social location recommendation: a kernel density estimation approach. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 334–343.

[44] Jia-Dong Zhang and Chi-Yin Chow. 2015. GeoSoCa: Exploiting geographical, social and categorical correlations for point-of-interest recommendations. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 443–452.

[45] Jia-Dong Zhang, Chi-Yin Chow, and Yanhua Li. 2014. Lore: Exploiting sequential influence for location recommendations. In *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 103–112.

[46] Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schölkopf. 2003. Learning with local and global consistency.. In *NIPS*, Vol. 16. 321–328.

[47] Xiaojin Zhu, Zoubin Ghahramani, John Lafferty, and others. 2003. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*, Vol. 3. 912–919.