# Few-Shot Learning for New User Recommendation in Location-based Social Networks

Ruirui Li
UCLA
rrli@cs.ucla.edu

Xian Wu
Univeristy of Notre Dame
xwu9@nd.edu

Xiusi Chen
UCLA
xiusi@cs.ucla.edu

Wei Wang
UCLA
weiwang@cs.ucla.edu

## ABSTRACT

The proliferation of GPS-enabled devices establishes the prosperity of location-based social networks, which results in a tremendous amount of user check-ins. These check-ins bring in preeminent opportunities to understand users' preferences and facilitate matching between users and businesses. However, the user check-ins are extremely sparse due to the huge user and business bases, which makes matching a daunting task. In this work, we investigate the recommendation problem in the context of identifying potential new customers for businesses in LBSNs. In particular, we focus on investigating the geographical influence, composed of geographical convenience and geographical dependency. In addition, we leverage metric-learning-based few-shot learning to fully utilize the user check-ins and facilitate the matching between users and businesses. To evaluate our proposed method, we conduct a series of experiments to extensively compare with 13 baselines using two real-world datasets. The results demonstrate that the proposed method outperforms all these baselines by a significant margin.

## CCS CONCEPTS

• **Information systems** Location based services.

## KEYWORDS

Customer recommendation; self-attention; few-shot learning

**ACM Reference Format:**
Ruirui Li, Xian Wu, Xiusi Chen, and Wei Wang. 2020. Few-Shot Learning for New User Recommendation in Location-based Social Networks. In *Proceedings of The Web Conference 2020 (WWW '20), April 20–24, 2020, Taipei, Taiwan.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3366423.3379994

## 1 INTRODUCTION

As an increasingly popular application of location-based services, location-based social networks (LBSNs), such as Yelp and Instagram, attract millions of users to share their locations, resulting in a huge amount of user check-ins [25, 40]. The availability of such unprecedented user check-ins brings in great opportunities to understand users' preferences and help businesses identify new users as potential customers. However, new user predictions in LBSNs suffer severely from data sparsity. Therefore, understanding customer preferences and making accurate predictions from the severely sparse data remain a daunting task.


**Figure 1: Geographical convenience influence**

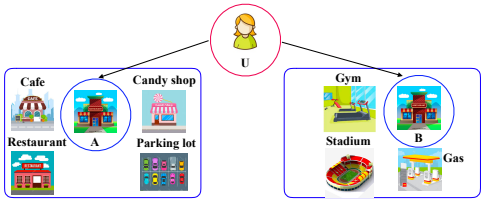



**Figure 2: Geographical dependency influence**

To compensate for the sparse data, various ancillary information, such as geographical influence, social correlations, and temporal patterns, has been leveraged to improve recommendation performances in different manners [4, 10, 19, 21, 29, 35, 38]. For LBSNs, the geographical coordinates, i.e., the latitude and longitude, of businesses are the most accessible ancillary information and they are also the ones that make location-based recommendations unique compared to other recommendation tasks. However, most existing works only investigate the relationship between users and businesses by measuring the distance of a visit between them. This leads to two inherent limitations. First, distance may not be an accurate indicator to distinguish the transportation convenience for different users for a check-in. Second, the inter-dependencies among nearby businesses are not well modeled when making recommendations.

In this work, we highlight that both geographical convenience and dependency should be incorporated in order to comprehensively express the geographical influence. Figure 1 and Figure 2 show two motivating examples for geographical convenience and dependency, respectively. In Figure 1, the two users are both 5 miles away from business *A*. But the distance indicator does not offer too much disciminative power to tell who is more likely to visit from the geographical perspective. If we know that user 1 tends to drive while user 2 relies on walking, we could gauge the actual

The work was done before the first author joined Amazon.


*WWW '20, April 20–24, 2020, Taipei, Taiwan*

ACM ISBN 978-1-4503-7023-3/20/04.
https://doi.org/10.1145/3366423.3379994

transportation efforts more accurately based on the convenience rather than the raw distance. In Figure 2, two businesses $A$ and $B$ provide the same service and they are reachable for user $u$ with equal transportation efforts. Without considering the neighborhood information of the two businesses, the recommendation system can barely distinguish one from the other regarding $u$'s preference. In real-world scenarios, the neighborhood services of two businesses are never the same [9]. If the neighborhood information is modeled when making recommendations, it can provide extra guidance to understand users' decision-making processes more comprehensively and thus make more accurate recommendations.

Beyond embracing geographical convenience and dependency to address the data sparsity issue, we also strive to seek more suitable techniques to fully utilize the user check-ins with the goal of improving recommendations in LBSNs. Few-shot learning aims to learn effectively from limited instances and has demonstrated notable performances in different domains [12, 36, 37]. In this work, we apply a metric-learning-based few-shot learning framework by optimizing two types of instances. More specifically, we construct support instances and query instances, with each instance composed of one user and one business. Support instances are labeled instances and serve as references. Query instances rely on the references to conduct reasoning. The model evolves by iterative reasoning between support and query instances. In this way, the matching between a user and a business is optimized with explicit attention to multiple other related support instances. Therefore, the limited check-ins are comprehensively utilized to make recommendations.

In this paper, we study the problem of potential new user recommendation in LBSNs. The main contributions of this work are as follows:

- We decompose the geographical influence into geographical convenience and dependency. The geographical convenience models the relative transportation efforts of a check-in, while the geographical dependency modeling makes our model neighborhood-aware.
- We apply meta-learning to location-based recommendation tasks and formulate the problem as metric-learning-based few-shot learning.
- We present an empirical evaluation of our approach against 13 recommendation methods on two real-world datasets. The results show that our approach outperforms all baseline methods in suggesting potential new users in different cities.

## 2 PROBLEM FORMULATION & SETTING

In this work, user check-ins are represented as a collection of tuples $\{(b, u)\} \subseteq B \times U$, where $B$ and $U$ are the business set and user set, respectively. The task of new user recommendation is to rank users given a business. The goal is to rank the true new users higher than other candidates. The candidates here are all users who have not checked in this business.

As we mentioned in the introduction, few-shot learning can fully utilize the training instances, which could potentially improve the recommendation performance. Therefore, we formulate the user recommendation task as a metric-learning-baesd few-shot learning problem in this work. Following the few-shot learning settings in [26, 28, 34], we assume access to a set of training tasks, where each training task $T$ corresponds to the new user predictions regarding a business $b$. During training, we aim to learn a generalized similarity metric to compare a set of user-business tuples against some references for each task, with each task designed to simulate the few-shot setting. Tasks are optimized one after another multiple times. For each task, each time $k$ observed check-in tuples are randomly sampled as references, denoted as $R$. An observed check-in tuple refers to a business-user pair $(b, u)$ where the user $u$ did check in business $u$ in the dataset. In addition, two query sets, i.e., a positive query set $Q^+$ and a negative query set $Q^-$, are constructed, with each set made up of $c$ tuples. Each query in $Q^+$ is also an observed check-in tuple regarding $b$, but distinct from the ones in $R$. Each query in $Q^-$ is a fake check-in tuple, where the user in the tuple did not check in $b$. The model thus can be optimized by comparing two types of similarities, one between a positive query and references, and the other one between a negative query and reference for each business. Ranking loss is applied to conduct the model optimizations, where the ranking loss measures how well the model distinguishes a positive query from a negative query regarding a set of references. The optimizations run for multiple iterations and each business is optimized more than one times. Note that for each optimization of a business, we may select different observed check-ins as references and positive queries. Similarly, we may construct different fake check-ins as negative queries. Once trained, the embeddings of all observed check-ins regarding the same business are expected close to each other in the hidden space, while the embedding of a fake check-in is expected faraway from the embeddings of observed check-ins.

## 3 METHODOLOGY

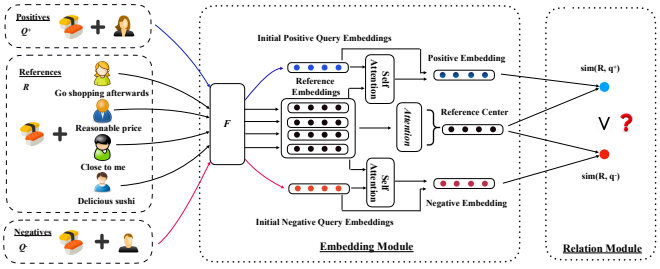



**Figure 3: Few-shot learning framework**

In this section, we explain how to model the recommendation task as few-shot learning and how to incorporate geographical convenience and dependency in detail.

The proposed approach decomposes the recommendation problem into a set of tasks and each task involves the user recommendations with respect to only one business. Figure 3 shows the few-shot learning framework for each task. On the left side, a reference set $R$ and two query sets, a positive query set $Q^+$ and a negative query set $Q^-$, are constructed for a business $b$. The reference set is composed of $k$ random observed check-ins, where $k = 4$ in this example. The positive query set is made up of another random selection of $c$ observed check-ins. But the check-ins in $Q^+$ are mutually exclusive alternatives from the ones in $R$. For illustration simplicity, $c$ is set to

1 in the example. The negative query set is constructed by building $c$ user-business tuples, such that each user in the tuple does not check in business $b$. The reference set functions as the supports in the setting of few-shot learning. The two query sets, based on the reference set, jointly conduct reasoning and inference.

The framework has two modules, an embedding module and a relation module. The embedding module learns the representations of references and queries, while the relation module compares the learnt representations and optimize them in such a manner that representations of positive queries are similar to the ones of references, while representations of negative queries are dissimilar to the ones of references. In the embedding module, $F(\cdot)$ is a layer which learns the initial embeddings of reference, positive query, and negative query tuples. By going through $F$, each reference/positive/negative tuple is represented by a fixed-length vector. Attention is a layer which utilizes the attention mechanism to learn the representative of the references. Self Attention is another attention layer, which takes both initial query and reference embeddings as inputs to generate the relative embeddings for queries. The relative embedding of a query learns to use references to explain the user check-in behavior encoded in the query. In the relation module, based on the learnt embeddings, we match each query in the query set to the reference representative by calculating the similarity between them, denoted as $\text{sim}(R, q)$. Then, we compare the score $\text{sim}(R, q^+)$ of a positive query $q^+$ with the score $\text{sim}(R, q^-)$ of a negative query $q^-$. Ranking loss is generated if a negative query is more similar to the reference representative than a positive query is. The model gets optimized by minimizing such ranking loss.

Given a tuple $(b, u)$, $F(\cdot)$ encodes four types of features: (1) business features, which represent its service, quality, and other business self-related factors; (2) user features, which represent his/her preference; (3) features indicating the geographical convenience of $b$ for user $u$; and (4) features indicating the geographical dependencies of $b$, i.e., the neighborhood information of $b$. These four types of features collectively express how likely the user $u$ will check in the business $b$.

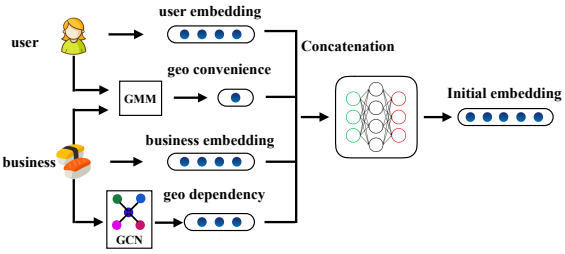



**Figure 4: Feature constructions**

Figure 4 illustrates the initial embedding construction for a user-business tuple. Given a tuple, two vectors, a user embedding vector and a business embedding vector, are utilized to encode user preferences and business self-related features, respectively. A geographical convenience vector is constructed by considering the geographical location of the business and the historical check-in locations of the user. A geographical dependency vector is constructed to encode the neighborhood information of the business. These four types of information are concatenated together and then fed into a fully-connected neural network to derive the initial embedding of a user-business tuple.

## 3.1 Geographical Convenience Modeling

We follow [11] and discuss how to model the geographical convenience of a business $b$ for a user $u$ based on $u$'s historical check-ins.

Gaussian mixture model [23] is applied to model the convenience. A Gaussian mixture model is a weighted aggregation of $M$ component Gaussian densities:

$$p(\mathbf{l}|\Phi) = \sum_{m=1}^{M} \alpha_m g(\mathbf{l}|\mu_m, \Sigma_m), \tag{1}$$

where $\mathbf{l}$ is the location vector, $\alpha_m$ is the mixture weight, and $g(\mathbf{l}|\mu_m, \Sigma_m)$ are the component Gaussian densities. Each component density is a 2-variate Gaussian function. Formally,

$$g(\mathbf{l}|\mu_m, \Sigma_m) = \frac{1}{2\pi|\Sigma_m|^{1/2}} e^{-\frac{1}{2}(\mathbf{l}-\mu_m)'\Sigma_m^{-1}(\mathbf{l}-\mu_m)},$$

where $\Phi$ is the mean location vector and $\Sigma_m$ gives the covariance. The complete Gaussian mixture model is parameterized by the mean location vectors, covariance matrices and mixture weights from all mixture components. $\Phi$ is used to denote these parameters. For a particular customer, given a set of his historical $\mathcal{T}$ check-ins, represented by $\mathcal{T}$ location vectors $L = \{\mathbf{l}_1, .., \mathbf{l}_\mathcal{T}\}$, the GMM likelihood is given by:

$$p(L|\Phi) = \prod_{\tau=1}^{\mathcal{T}} p(\mathbf{l}_\tau|\Phi).$$

We use the Expectation-Maximization algorithm [2] to estimate the parameters. After the GMM construction for a customer $u$, given the geographical location $l_b$ of a business $b$, as shown in Equation 1, $p(\mathbf{l}_b|\Phi)$ gives the geographical convenience of the business $b$ for user $u$. We highlight that the geographical convenience, modeled by GMM, is superior to conventional distance-based metrics, since it captures the relative efforts of a visit, which is capable of distinguishing customers with different traveling flexibility more accurately.

## 3.2 Geographical Dependency Modeling

In this section, we show how to encode geographical neighborhood information using graphs and how to model the dependence relationship among businesses using graph convolutional networks [8].

The geographical correlations among businesses are modeled with a graph $G = (V, E)$, which encodes the geographical proximity. Each vertex $v \in V$ represents a business and an edge $e \in E$ with weight $e^{-\lambda(v_i, v_j)}$ connects every two vertices $v_i$ and $v_j$, where $\lambda(v_i, v_j)$ gives the geographical distance between $v_i$ and $v_j$. Formally, an adjacency matrix $A$ is used to represent $G$ with $A_{i,j} = e^{-\lambda(v_i, v_j)}$.

Graph convolutional network (GCN) is defined over the proximity graph, which allows us to extract and aggregate neighborhood information for each vertex. A graph convolution is defined as:

$$\mathbf{H}^{(\beta+1)} = \text{Sigmoid}(\mathbf{D}^{-\frac{1}{2}}\hat{\mathbf{A}}\mathbf{D}^{-\frac{1}{2}}\mathbf{H}^{(\beta)}\mathbf{W}^{(\beta)}), \tag{2}$$

with $\hat{\mathbf{A}} = \mathbf{A} + \mathbf{I}$, where $\mathbf{I}$ is the Identify matrix, which is added to capture business' own features during feature propagations. $\mathbf{D}$ is the diagonal node degree matrix of $\hat{\mathbf{A}}$. $\mathbf{W}^{(\beta)}$ is the weight matrix for the $\beta$-th layer in GCN, and $\mathbf{H}^{\beta}$ is the output for the $\beta$-th layer.

In particular, $\mathbf{H}^{(0)} = \mathbf{X}$ and $\mathbf{H}^{(\tilde{\beta})} = \mathbf{Z}$, where $\mathbf{X}$ is the initial vertex feature matrix, $\mathbf{Z}$ is the final outputs of GCN, with $\tilde{\beta}$ indicating the number of layers in GCN.

For example, $\mathbf{H}_i^0$ represents the initial features of business $i$. By going through GCN, the information of $i$'s neighbors gets propagated to $\mathbf{H}_i^{\beta}$. Therefore, $\mathbf{H}_i^{\beta}$ not only represents the information of business $i$, but also that of its nearby neighbors.

## 3.3 Embedding and Relation Module

**Algorithm 1:** Few-shot Training

**Input:** Meta-learning task set;
**Output:** A set of model parameters $\Theta$;
**Initialization:** initialize $\Theta$ with Normal distributions;
**foreach** *epoch = 1:N* **do**
  Shuffle the tasks in the task set;
  **foreach** $T_b$ *in the task set* **do**
    Sample a set $R$ of observed user check-ins as references;
    Sample a set $Q^+$ of observed user check-ins as positive queries;
    Construct a set $Q^-$ of fake user check-ins as negative queries;
    Learn reference and query representations;
    Calculate the representative of the references with attention;
    Calculate the similarities between references and the queries in $Q^+$ and $Q^-$;
    Calculate ranking loss $\mathcal{L} = \sum_{(q^+, q^-)} \ell$;
    Update $\Theta$ based on gradients $g \propto \nabla \mathcal{L}$;
**Return** $\Theta$;

Algorithm 1 summarizes the training process. For each training epoch, we go through the tasks one by one. For each task, we aim to distinguish positive queries from negative queries regarding references. For a business $b$, we first sample a set of $k$ observed check-ins as the reference set, $R = \{(b, u_1^r), ..., (b, u_k^r)\}$. Then, we sample another set of $c$ exclusive observed check-ins as the positive query set, $Q^+ = \{(b, u_1^+), ..., (b, u_c^+)\}$. We also construct a third set of $c$ fake check-ins as the negative query set, $Q^- = \{(b, u_1^-), ..., (b, u_c^-)\}$. After the constructions of references, positive and negative queries, we calculate the similarity between each query in $Q^+ \cup Q^-$ and the references. We expect that positive queries are closer to references, while negative queries are faraway from references in the hidden space. The representations of queries and references are learnt through two attention mechanisms. The closeness/similarity between a query and a set of references is calculated by comparing the query embedding with the embedding of the reference representative. For each optimization of a business, we randomly pair up positive queries with negative queries. Ranking loss is adopted if a negative query is closer to the references than a positive query is. In the following paragraphs, we discuss the representation learning of both queries and references, the query-reference similarity calculation, and the ranking loss function in detail sequentially.

**Query embedding**: the query embedding is constructed by incorporating two types of information. One encodes the user business interaction behavior itself and the other one encodes the representation with attention to the references. In other words, we attempt to use references to explain the user business interaction behavior in the query. $F(\cdot)$ in Figure 3 yields the first part, while the second part is achieved by introducing a self-attention mechanism. The scaled dot-product attention [27] is defined as:

$$\text{Attention}(\tilde{\boldsymbol{Q}}, \tilde{\boldsymbol{K}}, \tilde{\boldsymbol{V}}) = \text{softmax}(\frac{\tilde{\boldsymbol{Q}}\tilde{\boldsymbol{K}}^T}{\sqrt{d_Q}})\tilde{\boldsymbol{V}}, \tag{3}$$

where $\tilde{\boldsymbol{Q}}$, $\tilde{\boldsymbol{K}}$, and $\tilde{\boldsymbol{V}}$ represent the queries, keys, and values in the attention mechanism, respectively. The attention operation calculates a weighted sum of all values, where the weight between query $i$ and value $j$ relates to the interaction between query $i$ and key $j$. $d_Q$ is the feature dimension of $\tilde{\boldsymbol{Q}}$ and $\sqrt{d_Q}$ serves as a scale factor.

In our case, the self-attention operation takes the query embeddings $\boldsymbol{Q} \in \mathbb{R}^{c \times d}$ and the reference embeddings $\boldsymbol{R} \in \mathbb{R}^{k \times d}$ as inputs, converts them to three matrices through linear projections, and feeds them into an attention layer:

$$\boldsymbol{Q}^R = \text{Attention}(\boldsymbol{Q}\boldsymbol{W}^Q, \boldsymbol{R}\boldsymbol{W}^K, \boldsymbol{R}\boldsymbol{W}^V), \tag{4}$$

where the projection matrices $\boldsymbol{W}^Q$, $\boldsymbol{W}^K$, and $\boldsymbol{W}^V \in \mathbb{R}^{d \times d}$. The self-attention result $\boldsymbol{Q}^R$ learns the embedding of a query by comparing the closeness between the query and all references. $\boldsymbol{Q}^R$ is a weighted sum of reference embeddings, where each weight gauges the behavior similarity between the query and a reference. In this way, $\boldsymbol{Q}^R$ encodes the user-business behavior of the query explained by references. We employ residual shortcut connection [5] to derive the final representations for queries $Q$, denoted as $Q^{com}$, as follows:

$$\boldsymbol{Q}^{com} = \boldsymbol{Q}^R + \boldsymbol{Q}. \tag{5}$$

**Reference embedding**: for the references, we calculate the reference representative $\bar{R}$ as a weighted sum of each reference, where the weights can be derived from a second attention mechanism.

$$\alpha_i = \text{softmax}(\sigma(\boldsymbol{W}_A\boldsymbol{R}_i + \boldsymbol{b}_A)\boldsymbol{V}_C^T). \tag{6}$$

$$\bar{\boldsymbol{R}} = \sum_i \alpha_i \boldsymbol{R}_i. \tag{7}$$

Equations 6 and 7 summarize the representative calculation. Each reference $\boldsymbol{R}_i$ is first fed into a one-layer neural network, the outputs of which, together with the context vector $\boldsymbol{V}_C$, are further used to generate the importance weight $\alpha_i$ for each reference $R_i$ through a softmax function. The representative $\bar{\boldsymbol{R}}$ is calculated as a weighted sum of the references based on the derived weights.

**Similarity and loss function:** Given a set of references $R$ and a query $q$, the similarity $\text{sim}(R, q)$ between $R$ and $q$ is defined as the dot product between $\bar{\boldsymbol{R}}$ and $\boldsymbol{q}^{com}$. Formally,

$$\text{sim}(R, q) = \bar{\boldsymbol{R}} \cdot \boldsymbol{q}^{com}. \tag{8}$$

We apply hinge loss to gauge the ranking error defined on references $R$, a positive query $q^+$, and a negative query $q^-$:

$$\ell = \max\{0,\ \text{sim}(R, q^-) - \text{sim}(R, q^+) + \gamma\}, \tag{9}$$

where $\gamma$ is the margin. Losses are generated only when the negative query is closer to the references than the positive query regarding a margin $\gamma$.

# 4 EXPERIMENTS

In this section, we conduct extensive experiments on two real-world datasets to evaluate the performance of the proposed method.

## 4.1 Datasets and Experimental Settings

The experiments are conducted on two datasets. One is the dataset from the Yelp challenge. The other is the Foursquare dataset. For the Yelp dataset, we investigate the recommendation tasks in six large cities. The Foursquare dataset contains interactions between

customers and businesses in Los Angeles and New York. Table 1 shows the statistics for the eight cities in the two datasets.

## 4.2 Baselines

To compare our approach with others, the following 13 methods are adopted as baselines. The proposed method, which utilizes SElf-Attention and few-shoT LEarning, is denoted as *SEATLE*.

Recommendation methods without considering geographical influence:

- **WRMF** minimizes the square error loss by assigning both observed and fake check-ins with different weights [7].
- **MMMF** minimizes the hinge loss based on matrix factorization [31].
- **BPRMF** optimizes pairwise bpr losses of observed check-ins and sampled fake check-ins [22].
- **CofiRank**, [30] optimizes the estimation of a ranking loss based on normalized discounted cumulative gain.
- **CLiMF**, [24] optimizes a different ranking-oriented mean reciprocal rank loss.

Conventional methods with geographical influence involved:

- **USG**, [39] is a collaborative filtering method. It utilizes distances between users and businesses as extra guidance to conduct recommendations.
- **GeoMF**, [16] explicitly learns user activity areas (distance-based) and business influences areas via matrix factorization.
- **Rank-GeoFM**, ranking-based geographical factorization [14] incorporates business neighborhood information via matrix factorization.
- **ASMF**, [10] mainly leverages social network information to improve recommendations.
- **ARMF**, [10] extends ASMF by further optimizing ranking losses.
- **CORALS**, [11] models geographical convenience and business reputation to improve recommendations.

Deep learning methods with geographical influence involved:

- **SAE-NAD**, self-attentive encoder and neighbor-aware decoder [21] applies auto-encoders to make recommendations.
- **PACE**, preference and context embedding [35], a deep neural architecture that jointly learns the embeddings of users and businesses by building a context graph.

## 4.3 Recommendation Performance

In this section, we evaluate the performances of SEATLE against the 13 baseline methods. Mean Average Precision (MAP) is adopted as the evaluation metric, which is also used in [10, 11, 13].

Figure 5 shows the recommendation performances of different methods on the eight cities from the two datasets. Figures from 5a to 5f show the performances based on the six cities in the Yelp dataset, while the last two Figures 5g and 5h show the performances based on the two cities in the Foursquare dataset.

Among methods which do not consider geographical influence, MMMF, BPRMF, CofiRank, and CLiMF achieve better recommendation performances than WRMF in general. This demonstrates that point-wise methods, such as WRMF, which achieve low prediction errors, do not necessarily have high recommendation accuracy. In other words, directly optimizing the predicted check-ins may not provide the best recommendation lists to businesses.

After leveraging geographical influence, Rank-GeoMF, ASMF, ARMF, GeoMF, CORALS, SAE-NAD, and PACE outperform the five above methods, which do not incorporate any ancillary features. It verifies that modeling ancillary information can offer extra guidance and compensate for the sparsity issue in location-based recommendation tasks. USG, with geographical influence modeled, does not perform as well as expected in some cities, such as Charlotte, Las Vegas, etc. This is due to its oversimplified model design, which is a straightforward linear combination of user preference and geographical distance scores without proper optimizations. The performances of ASMF and ARMF on the foursquare dataset are not as good as it usually achieves on the Yelp dataset. ASMF and ARMF mainly focus on leveraging social network information. This demonstrates that social network information may not always be reliable, while comprehensively utilizing geographical influence might be a better choice to improve performances in location-based recommendation tasks. In general, CORALS outperforms Rank-GeoMF, GeoMF, SAE-NAD, and PACE. This mainly results from the geographical convenience incorporated in CORALS rather than distance-based metrics employed in other models.

Among the deep learning-based methods, SEATLE achieves the best performance. The reasons could be explained as follows. PACE models the geographical influence by a context graph, which does not explicitly model the user reachability to businesses. SAE-NAD captures the geographical dependency through a neighbor-aware auto-encoder, but it fails to incorporate geographical convenience. In general, SEATLE outperforms all baseline methods in the eight cities over the two datasets. SEATLE models both geographical convenience and dependency, which collaboratively express the power of geographical influence. Moreover, SEATLE adopts few-shot learning, designed for learning with limited data, as the framework to fully utilize the sparse training instances. These appropriate designs make SEATLE a good fit for new user recommendations in LBSNs.

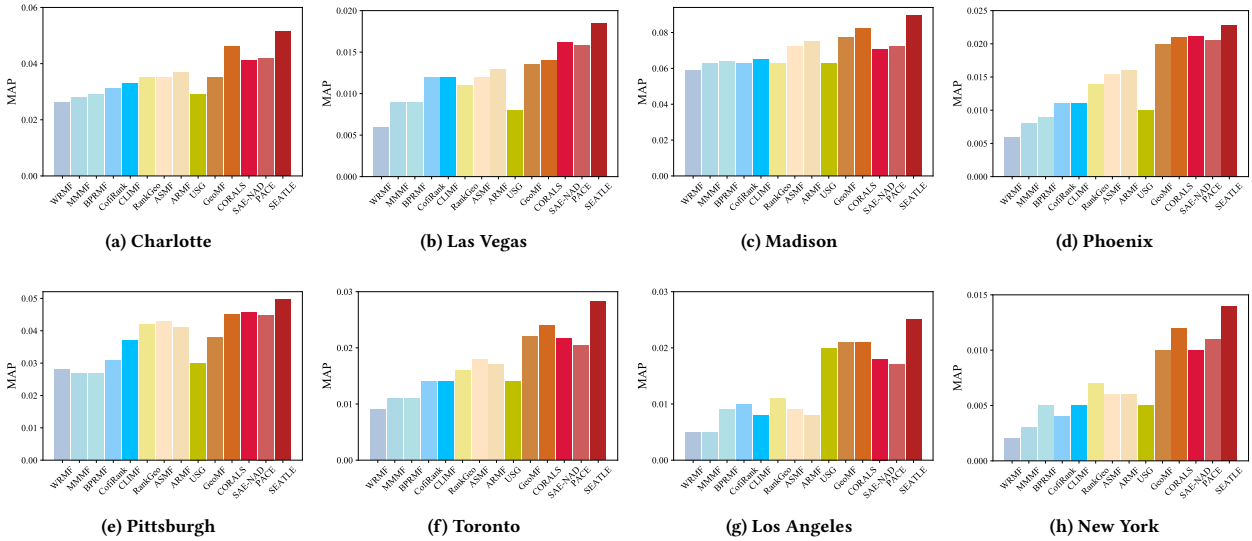
## 4.4 Geographical Influence Analysis

In this section, we investigate the effectiveness of geographical convenience and dependency modelings. We develop SEATLE$_{con-}$ and SEATLE$_{dep-}$ by removing the convenience and dependency feature from SEATLE, respectively. To compare convenience-based and distance-based influence, we further develop SEATLE$_{dist}$ by replacing the convenience feature by a distance-based kernel metric, adopted from SAE-NAD. More precisely, the metric is given by $\exp(\gamma|\boldsymbol{l}_i - \boldsymbol{l}_j|)$, where $\boldsymbol{l}_i$ and $\boldsymbol{l}_j$ are both location coordinates, and $\gamma$ is used to control the correlation level of the two locations.

Figure 6 shows the MAP performances. We observe that when the geographical convenience and dependency features are removed from SEATLE, MAP performance drops correspondingly. The performance decreases more when eliminating geographical convenience as compared to eliminating geographical dependency. This observation applies to all eight cities. Therefore, we can safely conclude that both geographical convenience and dependency modelings improve the recommendation performance and the geographical convenience contributes more. We further compare SEATLE$_{dist}$

**Table 1: Business and customer statistics**

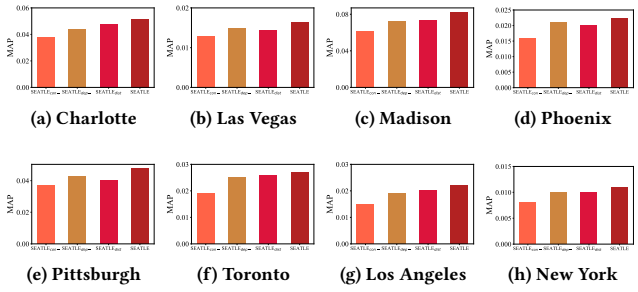
| Dataset | Yelp | | | | | | Foursquare | |
|---|---|---|---|---|---|---|---|---|
| City | Charlotte | Las Vegas | Madison | Phoenix | Pittsburgh | Toronto | Los Angeles | New York |
| # of Customers | 69,005 | 432,399 | 26,083 | 314,610 | 51,422 | 58,377 | 501,940 | 717,382 |
| # of Businesses | 10,652 | 282,204 | 3,895 | 43,482 | 8,037 | 20,849 | 215,614 | 206,416 |




(a) Charlotte (b) Las Vegas (c) Madison (d) Phoenix

(e) Pittsburgh (f) Toronto (g) Los Angeles (h) New York

**Figure 5: MAP performances of different methods over eight cities**




(a) Charlotte (b) Las Vegas (c) Madison (d) Phoenix

(e) Pittsburgh (f) Toronto (g) Los Angeles (h) New York

**Figure 6: Geographical influence analysis over eight cities**

and SEATLE. SEATLE$_{dist}$ models distance-based geographical features, while SEATLE is convenience-based. We notice that SEATLE outperforms SEATLE$_{dist}$ on all eight cities. This demonstrates the advantage of convenience-based geographical modeling since it gauge users' actual transportation efforts more accurately.

## 5 RELATED WORK

To address the check-in sparsity issue, various ancillary information is incorporated when building recommendation models, such as POI popularity, social influence, temporal patterns, textual and visual contents [3, 4, 6, 10, 15, 18, 29, 32, 33, 39, 41, 43]. In this part, we focus on investigating geographical influence oriented works.

To leverage geographical influence to improve recommendation performances, Cheng et al. [1] first detect multiple centers for each customer based on their check-in histories. Recommendations are made by referring to the distance between the location of the business and the nearest customer center. In [39], geographical influence is modeled by a power-law distribution between the check-in probability and the pair-wise distance of two check-ins. [17, 42] utilize the kernel density estimation to study customers' check-ins and avoid employing a specific distribution. [20] exploits geographical neighborhood information by assuming that customers have similar preferences on neighboring POIs and POIs in the same region may share similar user preferences. PACE [35] explores the use of deep neural networks to learn location embeddings and user preferences over POIs. SAE-NAD [21] applies auto-encoder to learn POI recommendations. APOIR [44] employs an adversarial generative model to make POI recommendations. SACRA [13] constructs dynamic adversarial examples to make robust recommendations in LBSNs.

The proposed method, SEATLE, differs from most of the above work. SEATLE models both geographical convenience and dependency. Moreover, SEATLE employs few-shot learning to fully utilize the training instances and strives to improve recommendations from limited data.

## 6 CONCLUSION

In this work, we study the problem of recommending new users to businesses in LBSNs. We formulate the recommendation problem as a metric-learning-based few-shot learning problem in order to fully utilize the sparse training instances. We look into the geographical influence and decompose it into geographical convenience and geographical dependency modeling. Extensive experiments are conducted, which demonstrates the effectiveness of SEATLE on two real-world datasets.

# REFERENCES


[1] Chen Cheng, Haiqin Yang, Irwin King, and Michael R. Lyu. 2012. Fused Matrix Factorization with Geographical and Social Influence in Location-Based Social Networks. In *AAAI '12, July 22-26, 2012, Toronto, Ontario, Canada.*

[2] Arthur P Dempster, Nan M Laird, and Donald B Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the royal statistical society. Series B (methodological)* (1977).

[3] Huiji Gao, Jiliang Tang, Xia Hu, and Huan Liu. 2013. Exploring temporal effects for location recommendation on location-based social networks. In *RecSys '13, Hong Kong, China, October 12-16, 2013.*

[4] Huiji Gao, Jiliang Tang, Xia Hu, and Huan Liu. 2015. Content-Aware Point of Interest Recommendation on Location-Based Social Networks. In *AAAI '15, January 25-30, 2015, Austin, Texas, USA.* 1721–1727.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016.* 770–778.

[6] Bo Hu and Martin Ester. 2014. Social Topic Modeling for Point-of-Interest Recommendation in Location-Based Social Networks. In *ICDM '14, Shenzhen, China, December 14-17, 2014.* 845–850.

[7] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative Filtering for Implicit Feedback Datasets. In *(ICDM '08), December 15-19, 2008, Pisa, Italy.*

[8] Thomas N. Kipf and Max Welling. 2016. Semi-Supervised Classification with Graph Convolutional Networks. *CoRR* abs/1609.02907 (2016). arXiv:1609.02907

[9] Wolters Kluwer. 2018. Evaluating the Neighborhood When Choosing a Business Facility. https://www.bizfilings.com/toolkit/research-topics/office-hr/evaluating-the-neighborhood-when-choosing-a-business-facility.

[10] Huayu Li, Yong Ge, Richang Hong, and Hengshu Zhu. 2016. Point-of-Interest Recommendations: Learning Potential Check-ins from Friends. In *SIGKDD '16, San Francisco, CA, USA, August 13-17, 2016.*

[11] Ruirui Li, Jyunyu Jiang, Chelsea Ju, and Wei Wang. 2019. CORALS: Who are My Potential New Customers? Tapping into the Wisdom of Customers' Decisions. In *WSDM '19, Melbourne, Australia, February 11-15, 2019.*

[12] Ruirui Li, Jyun-Yu Jiang, Jiahao Liu, Chu-Cheng Hsieh, and Wei Wang. 2020. Automatic Speaker Recognition with Limited Data. In *Proceedings of WSDM, Houston, Texas, USA, February 3-7.*

[13] Ruirui Li, Xian Wu, and Wei Wang. 2019. Adversarial Learning to Compare: Self-Attentive Prospective Customer Recommendation in Location based Social Networks. In *WSDM '20, Houston, TEXAS, February 3-7, 2020.*

[14] Xutao Li, Gao Cong, Xiaoli Li, Tuan-Anh Nguyen Pham, and Shonali Krishnaswamy. 2015. Rank-GeoFM: A Ranking based Geographical Factorization Method for Point of Interest Recommendation. In *SIGIR '15, Santiago, Chile, August 9-13, 2015.* 433–442.

[15] Defu Lian, Yong Ge, Fuzheng Zhang, Nicholas Jing Yuan, Xing Xie, Tao Zhou, and Yong Rui. 2015. Content-Aware Collaborative Filtering for Location Recommendation Based on Human Mobility Data. In *ICDM '15, Atlantic City, NJ, USA, November 14-17, 2015.* 261–270.

[16] 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 *KDD, New York, USA, August 24 - 27, 2014.*

[17] Moshe Lichman and Padhraic Smyth. 2014. Modeling human location data with mixtures of kernel densities. In *KDD '14, New York, USA - August 24 - 27, 2014.*

[18] Bin Liu, Yanjie Fu, Zijun Yao, and Hui Xiong. 2013. Learning geographical preferences for point-of-interest recommendation. In *KDD '13, Chicago, IL, USA, August 11-14, 2013.*

[19] Yanchi Liu, Chuanren Liu, Bin Liu, Meng Qu, and Hui Xiong. 2016. Unified Point-of-Interest Recommendation with Temporal Interval Assessment. In *SIGKDD '16, San Francisco, CA, USA, August 13-17, 2016.*

[20] Yong Liu, Wei Wei, Aixin Sun, and Chunyan Miao. 2014. Exploiting Geographical Neighborhood Characteristics for Location Recommendation. In *CIKM, Shanghai, China, November 3-7, 2014.* 739–748.

[21] Chen Ma, Yingxue Zhang, Qinglong Wang, and Xue Liu. 2018. Point-of-Interest Recommendation: Exploiting Self-Attentive Autoencoders with Neighbor-Aware Influence. In *CIKM.* ACM, 697–706.

[22] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *UAI '09, Montreal, QC, Canada, June 18-21, 2009.*

[23] Douglas A. Reynolds. 2009. Gaussian Mixture Models. In *Encyclopedia of Biometrics.*

[24] Yue Shi, Alexandros Karatzoglou, Linas Baltrunas, Martha Larson, Nuria Oliver, and Alan Hanjalic. 2012. CLiMF: learning to maximize reciprocal rank with collaborative less-is-more filtering. In *RecSys, Ireland, September 9-13, 2012.*

[25] Craig Smith. 2016. By the numbers: 20 important Foursquare Stats. http://expandedramblings.com/index.php/by-the-numbers-interesting-foursquare-user-stats/.

[26] Jake Snell, Kevin Swersky, and Richard S. Zemel. 2017. Prototypical Networks for Few-shot Learning. In *NIPS, 4-9 December 2017, Long Beach, CA, USA.* 4080–4090.

[27] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *NIPS, 4-9 December 2017, Long Beach, CA, USA.* 6000–6010.

[28] Oriol Vinyals, Charles Blundell, Tim Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. 2016. Matching Networks for One Shot Learning. In *NIPS, December 5-10, 2016, Barcelona, Spain.* 3630–3638.

[29] Suhang Wang, Yilin Wang, Jiliang Tang, Kai Shu, Suhas Ranganath, and Huan Liu. 2017. What Your Images Reveal: Exploiting Visual Contents for Point-of-Interest Recommendation. In *Proceedings of WWW, Perth, Australia, April 3-7, 2017.* 391–400.

[30] Markus Weimer, Alexandros Karatzoglou, Quoc V. Le, and Alexander J. Smola. 2007. COFI RANK - Maximum Margin Matrix Factorization for Collaborative Ranking. In *NIPS, Vancouver, British Columbia, Canada, December 3-6, 2007.*

[31] Markus Weimer, Alexandros Karatzoglou, and Alexander J. Smola. 2015. Improving maximum margin matrix factorization. *Machine Learning* 72, 3 (2015).

[32] Xian Wu, Yuxiao Dong, Baoxu Shi, Ananthram Swami, and Nitesh V. Chawla. 2018. Who will Attend This Event Together? Event Attendance Prediction via Deep LSTM Networks. In *Proceedings of the 2018 SIAM International Conference on Data Mining, SDM 2018, May 3-5, 2018, San Diego, CA, USA.* 180–188.

[33] Xian Wu, Baoxu Shi, Yuxiao Dong, Chao Huang, Louis Faust, and Nitesh V. Chawla. 2018. RESTFul: Resolution-Aware Forecasting of Behavioral Time Series Data. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018, Torino, Italy, October 22-26, 2018.* 1073–1082.

[34] Wenhan Xiong, Mo Yu, Shiyu Chang, Xiaoxiao Guo, and William Yang Wang. 2018. One-Shot Relational Learning for Knowledge Graphs. *CoRR* abs/1808.09040 (2018). arXiv:1808.09040 http://arxiv.org/abs/1808.09040

[35] 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 SIGKDD, Halifax, NS, Canada, August 13 - 17, 2017.* 1245–1254.

[36] Huaxiu Yao, Yiding Liu, Ying Wei, Xianfeng Tang, and Zhenhui Li. 2019. Learning from Multiple Cities: A Meta-Learning Approach for Spatial-Temporal Prediction. In *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019.* 2181–2191.

[37] Huaxiu Yao, Xian Wu, Zhiqiang Tao, Yaliang Li, Bolin Ding, Ruirui Li, and Zhenhui Li. 2020. Automated Relational Meta-learning. In *International Conference on Learning Representations.* https://openreview.net/forum?id=rklp93EtwH

[38] Zijun Yao, Yanjie Fu, Bin Liu, Yanchi Liu, and Hui Xiong. 2016. POI Recommendation: A Temporal Matching between POI Popularity and User Regularity. In *ICDM '16, December 12-15, 2016, Barcelona, Spain.*

[39] Mao Ye, Peifeng Yin, Wang-Chien Lee, and Dik Lun Lee. 2011. Exploiting geographical influence for collaborative point-of-interest recommendation. In *SIGIR '11, Beijing, China, July 25-29, 2011.*

[40] Yelp. 2017. Yelp for Business Owners. https://biz.yelp.com/support/what_is_yelp.

[41] Quan Yuan, Gao Cong, Zongyang Ma, Aixin Sun, and Nadia Magnenat-Thalmann. 2013. Time-aware point-of-interest recommendation. In *SIGIR '13, Dublin, Ireland - July 28 - August 01, 2013.*

[42] Jia-Dong Zhang and Chi-Yin Chow. 2013. iGSLR: personalized geo-social location recommendation: a kernel density estimation approach. In *SIGSPATIAL '13, Orlando, FL, USA, November 5-8, 2013.*

[43] Jia-Dong Zhang and Chi-Yin Chow. 2015. GeoSoCa: Exploiting Geographical, Social and Categorical Correlations for Point-of-Interest Recommendations. In *Proceedings of SIGIR, Santiago, Chile, August 9-13, 2015.* 443–452.

[44] Fan Zhou, Ruiyang Yin, Kunpeng Zhang, Goce Trajcevski, Ting Zhong, and Jin Wu. 2019. Adversarial Point-of-Interest Recommendation. In *WWW, San Francisco, CA, USA, May 13-17, 2019.* 3462–34618.