Personalized Context-Aware Point of Interest Recommendation

MOHAMMAD ALIANNEJADI and FABIO CRESTANI, Università della Svizzera italiana (USI), Switzerland

Personalized recommendation of Points of Interest (POIs) plays a key role in satisfying users on Location-Based Social Networks (LBSNs). In this paper, we propose a probabilistic model to find the mapping between user-annotated tags and locations' taste keywords. Furthermore, we introduce a dataset on locations' contextual appropriateness and demonstrate its usefulness in predicting the contextual relevance of locations. We investigate four approaches to use our proposed mapping for addressing the data sparsity problem: one model to reduce the dimensionality of location taste keywords and three models to predict user tags for a new location. Moreover, we present different scores calculated from multiple LBSNs and show how we incorporate new information from the mapping into a POI recommendation approach. Then, the computed scores are integrated using learning to rank techniques. The experiments on two TREC datasets show the effectiveness of our approach, beating state-of-the-art methods.

CCS Concepts: • **Information systems** → **Recommender systems**;

Additional Key Words and Phrases: user modeling, contextual suggestion, point of interest recommendation, content-based recommendation, location-based social networks

1 INTRODUCTION

Nowadays with the super fast growth in the amount of information available on the world wide web and with the increasing popularity of mobile devices equipped with easy access to the Internet, it is essential to assist users to find relevant and useful information according to their needs and context. Recommender systems aim to filter information in order to satisfy the users' information needs and minimize the effort made by the user to find relevant information. More specifically, recommender systems focus on suggesting items that can potentially be attractive to users [37]. The availability of *Location-Based Social Networks* (LBSNs) together with a *Global Positioning System* (GPS) and an always-on Internet access on mobile devices encourages many users to check in at various *Points of Interest* (POIs) using their favorite platform. Among the many available LBSNs, we can point out

some of the most popular ones. Yelp[1] is one of the most popular LBSNs in the US and mainly relies on users' reviews. Users leave very creative and usually long reviews about almost any type of POI on this service. According to the company's factsheet[2], 73% of searches on Yelp were performed on a mobile device[3]. Foursquare[4] introduces "tips" as short informative reviews and "tastes" as a way for users to express their expectations and preferences. TripAdvisor[5], on the other hand, focuses on reviews and content related to travel and travel companies. The popularity of such services enables them to gather various types of information about users including users' mobility, feedback and context. A key factor in satisfying the users' needs is being able to personalize the system to recommend POIs taking into account personal preference and contextual constraints [15].

Much work has been carried out to model users taking into consideration their preference history and current vicinity to POIs to help them explore new and interesting locations[6]. More specifically, POI recommendation tries to ensure user's satisfaction by suggesting her the most interesting locations, taking into account her preferences and contextual constraints [12, 15]. In fact, among the many proposed works, there has been several successful approaches [20, 26, 36, 40], addressing different problems of POI recommendation, taking into account various aspects. However, many challenges in this area are not resolved yet. A major challenge in POI recommendation is data sparsity [12, 43]. Normally, users visit a very limited number of locations, however, LBSNs feature a relatively huge number of locations with a large variety. Consequently, the user-item matrix used in *Collaborative Filtering* (CF) by many papers is sparse [48]. Several studies seek to address the data sparsity problem incorporating additional information into the model [12, 15]. For instance, Zhang et al. [46] derived virtual ratings from users' reviews and studied the impact of fusing them into CF. Zheng et al. [47], on the other hand, modeled users adopting a tensor representation, and introduced a matrix decomposition and regularized tensor to better address the data sparsity problem. Moreover, since users spend most of their time in their home town [30], the data sparsity problem is aggravated when a user visits a new city where she has no history of visited locations [21].

Apart from personal preference and interest, the user behavior is influenced and, in many cases, constrained by local and contextual preferences [20]. For instance, a user may be a big fan of *nightlife spots*. However, when traveling with her family, she may prefer not to visit such locations. Hence, it is crucial to consider a user's context when recommending locations to her. It is also important to note that the user's context often introduces new constraints, not necessarily inclined to her opinion and interest. To this end, the main focus of the *Text REtrieval Conference* (TREC) *Contextual Suggestion* (TREC-CS) track[7] in 2015 [18] and 2016 [25] was to improve location recommendation with the aid of contextual information. However, not many successful participants took into account context in their proposed approaches. Thus, applying contextual constraints still remains a challenge for context-aware POI recommendation.

Given the easy access to the Internet and availability of mobile devices such as cell phone, smart watches and tablets, users tend to leave their check-in data more often. However, writing a long review on such devices is not as trivial as is using a desktop computer. As a consequence, the majority of users rate locations without writing a review. Reviews contain critical information regarding a user's opinion and view about a location; for example a user's opinion about a location's

---

[1]http://www.yelp.com

[2]Yelp Inc. 2017. An Introduction to Yelp Metrics as of June 30, 2016. http://web.archive.org/web/20160825213451/https://www.yelp.com/factsheet. (2017). Accessed: 2017-03-23.

[3]As of June 30, 2016

[4]http://www.foursquare.com

[5]http://www.tripadvisor.com

[6]In this paper, we use the terms *location*, *venue* and *POI* interchangeably.

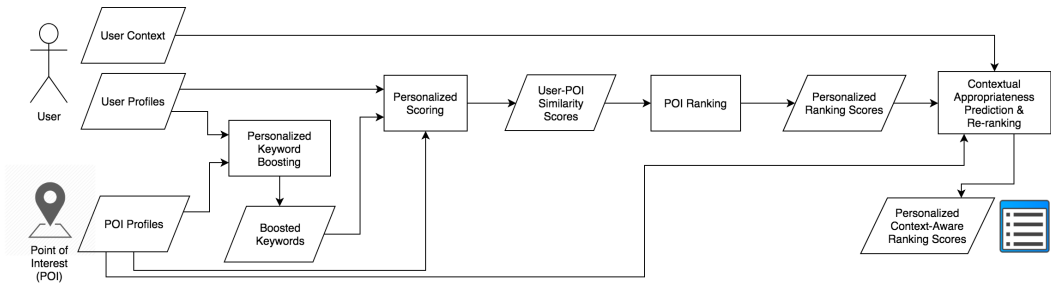[7]https://sites.google.com/site/treccontext/

Fig. 1. Overview of the proposed method.

view or staff. In order to compensate for the absence of such information, an LBSN can assist a user with a few related predefined *tags*, from which the user can conveniently select those expressing her opinion. Predefined tags come very handy especially on smaller devices such as smart watches enabling users to express themselves with the aid of a couple of taps. Modeling users with such tags is very challenging since user tags are much more sparse compared to user ratings. Thus the traditional CF approach could not be applied for user tag modeling. Furthermore, in real-world POI recommendation scenarios for mobile devices, the top 10 locations are usually interesting to users [16], because of the screen size of a typical mobile device and limited effort a typical user spends to go through the recommendation list. Therefore, providing a personalized ranking to the user is crucial, making this task a *top-k recommendation* task.

In this paper, we aim to answer the following research questions:

- RQ1: How can we model a user's interest and opinion based on her check-in record?
- RQ2: How can we leverage user tags to model users' interests and opinions more effectively to ultimately improve top-k POI recommendation effectiveness?
- RQ3: Could we use user tags to create a personalized model to reduce data dimensionality and address the data sparsity problem?
- RQ4: Could we model users to predict how they would tag a new location?
- RQ5: How can we incorporate users' context into top-k POI recommendation to improve the performance?
- RQ6: How can we integrate different aspects of information to generate personalized ranking considering both users' personal interests and contextual constraints?

In an effort to address these research questions, the contributions of this paper can be summarized as follows:

(1) We introduce a set of relevance scores for measuring the similarity between a user's history and a location considering location's content and reviews.
(2) We present a probabilistic generative approach to find the mapping between location taste keywords and user tags thus modeling the personalized opinion of users about venues more accurately.
(3) We address the sparsity problem by performing personalized boosting of location keywords in a user's history.
(4) We explore different machine learning models to predict user tags and evaluate the prediction effectiveness in terms of both tag prediction and recommendation effectiveness.
(5) We introduce a brand new dataset for predicting contextually appropriate locations and show how to predict the contextually appropriate locations given the user's current context and evaluate its effectiveness on recommendation.

(6) We evaluate several learning to rank techniques to incorporate boosting and tag prediction into our POI recommendation model using information from multiple LBSNs.

Figure 1 shows an overview of our proposed method. In the first step, user and POI profiles are analyzed to perform personalized keyword boosting, resulting in a list of boosted keywords for each user (see Section 2). Then the list of boosted keywords together with user and POI profiles are fed to the personalized scoring component to calculate the similarity scores between a given user and a POI. The scores are then passed to the ranking model (i.e., learning to rank) to produce a personalized ranked list of POIs for each user (see Section 4). Finally, given the user's current context, the level of contextual appropriateness of every venue is predicted (see Section 3) and used to re-rank the personalized ranking scores, resulting in a personalized context-aware ranked list of POIs.

This paper extends over three years of past work on personalized context-aware suggestion that resulted in a number of published papers. In particular, in [6] we first proposed a user model enrichment process leveraging user reviews from well-known LBSNs to support the venue suggestion process. In [4] we tackled the POI appropriateness prediction problem, proposing a novel approach based on a user model enrichment process. In [10] we combined multimodal information from multiple LBSNs to recommend POIs and studied how personalized mapping of location keywords to user tags could improve POI recommendation. Finally, in [8] we made publicly available a contextual dataset that we created using crowdsourcing to incorporate the contextual constraints in our personalized recommender system. All this work originated from our participation [3, 7] to TREC-CS 2015 and TREC-CS 2016 where we consistently obtained the best run, demonstrating the effectiveness of our approaches under a common evaluation framework. This paper draws together content from all the above papers, presenting a coherent synthesis of this full line of research. In addition, this paper extends our previous works as follows:

- It introduces a unified recommendation framework, combining our previous works and showing how this can improve recommendation.
- It presents deeper theoretical details of the proposed models, providing more discussion about keyword mapping, while elaborating on how to model user tag prediction as a sequence tagging problem.
- It provides a more extensive experimentation on different datasets evaluating the proposed models from various perspectives. In particular, we evaluate our POI recommendation method in terms of personalized dimensionality reduction, user tag prediction, and contextual appropriateness prediction. We furthermore analyze the results pertaining each information component and reducing the number of visited POIs.

This paper demonstrates that our proposed approach is able to outperform state-of-the-art strategies. In fact, experiments show that combining multimodal information from multiple LBSNs improves POI recommendation significantly. Moreover, we show that the proposed mapping of location keywords to user tags enables us to predict user tagging behavior effectively. We also show that predicting contextually appropriate locations and reranking suggestions according to their contextual appropriateness results in a more accurate top-k location recommendation.

The remainder of the paper is organized as follows. Before explaining our proposed recommendation approach, we describe two critical components of our model in Sections 2 and 3. We first describe how we model the statistical mapping between user tags and venue taste keywords in Section 2.1. Then, we explore two directions to use this information. First, we explain how we use the computed mapping to reduce the dimensionality of venue taste keywords in Section 2.3. Second, we describe how we use the computed mapping as training data to learn the sequential tagging of user tags in Section 2.4. The trained tagging models are then used to predict user tags for unseen

venues. Section 3 describes our proposed contextual dataset and elaborates our proposed approach to predict contextual appropriateness of locations. After describing the two main components of our model, in Section 4 we describe how we integrate them with other similarity measures in order to recommend POIs to users. Section 5 describes the evaluation protocol and Section 6 presents our experimental evaluation, while Section 7 reviews related work. We conclude our study in Section 8 proposing possible directions as future work.

2 PERSONALIZED KEYWORD BOOSTING

In this section we propose a probabilistic approach by which we map location keywords to user tags. Furthermore we propose two possible approaches of utilizing such additional information in order to enhance location recommendation. First, we propose to use the mapping as an additional information to reduce the dimensionality of location keyword space in order to address the data sparsity problem. Second, we use the mapping to train a sequence labeling model to predict user tags for a new location. We use the outcome of both directions to estimate the similarity of a location to a user in Section 4.

2.1 Personalized Keyword-Tag Mapping

In this section we present a probabilistic approach to map location keywords to user tags. We aim to find a meaningful correlation between the location content (e.g., keywords) and user tags since users annotate locations with tags based on both their personal views and locations' characteristics. We assume that the key characteristics that trigger the user's mind to annotate a location with a specific tag are of those listed in the location keywords. For example, tagging a location as *healthy-food* is a result of user's personal view reflected in the location's characteristics (e.g., keywords). Hence, a user who believes *vegan* foods are healthy may tag a *vegan* location as *healthy-food*, whereas another user with a different view may tag a *sushi* place as *healthy-food*. Therefore, user tags are dependent on both users views and locations' characteristics. That is why we need to find a meaningful mapping between user tags and location keywords to take into account locations' characteristics. The mapping needs to be personalized to model users' personal views. Figure 2 depicts a real example mapping with a set of two user tags and four location keywords. Our ultimate goal is to determine the most likely mapping of location keywords to user tags, personalized for each user.

For a given user u, let $\mathbf{f}^J = \langle f_1 \ldots f_j \ldots f_J \rangle$ be a sequence of location keywords. We aim to find the sequence of user tags $\mathbf{t}^I = \langle t_1 \ldots t_i \ldots t_I \rangle$. Note that $\mathbf{t}^I$ refers to a sequence named $\mathbf{t}$ with the length of I. Hence, $\mathbf{t}^i$ denotes a set with the size of i ($\mathbf{t}^i = \langle t_1 \ldots t_i \rangle$), whereas t_i refers to the i-th item of a given sequence. Our aim is to find a user tag sequence maximizing $Pr(\mathbf{t}^I | \mathbf{f}^J)$:

$$\hat{\mathbf{t}}^I = \underset{\mathbf{t}^I}{\operatorname{argmax}} \{ Pr(\mathbf{t}^I | \mathbf{f}^J) \} = \underset{\mathbf{t}^I}{\operatorname{argmax}} \{ Pr(\mathbf{f}^J | \mathbf{t}^I) Pr(\mathbf{t}^I) \} , \tag{1}$$

where $Pr(\mathbf{t}^I)$ models user tags. In fact, given $\mathbf{t}^I$, this function determines to what extend $\mathbf{t}^I$ is likely to be generated by a specific user. It basically models the user's behavior of tag annotation regardless of location keywords. We fairly assume that users annotate locations with a specific tag independent of other tags. In other words, we assume zero-order dependence of user tags. Hence, we rewrite $Pr(\mathbf{t}^I)$ as follows:

$$Pr(\mathbf{t}^I) = p(I) \prod_{i=1}^{I} p(t_i | \mathbf{t}^{i-1}, I) = p(I) \prod_{i=1}^{I} p(t_i | I) , \tag{2}$$
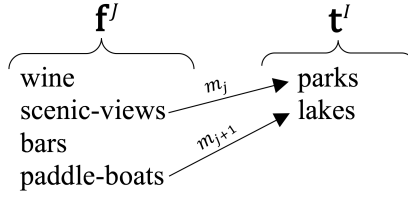
Fig. 2. An example of mapping of $J = 4$ location keywords to $I = 2$ user tags.

where $\mathbf{t}^{i-1} = \langle t_1 \ldots t_{i-1} \rangle$.

$Pr(\mathbf{f}^J|\mathbf{t}^I)$ in (1) models location keywords given a sequence of user tags $\mathbf{t}^I$. We need to find the optimum mapping between location keywords and user tags to optimally model location keywords given $\mathbf{t}^I$. Therefore, we marginalize the probability $Pr(\mathbf{f}^J|\mathbf{t}^I)$ over $\mathbf{m}^J$. We introduce $\mathbf{m}^J$ as the latent variable defining how location keywords are *mapped* to user tags: $\mathbf{m}^J = \langle m_1 \ldots m_j \ldots m_J \rangle$, with $m_j \in \{1, \ldots, I\}$:

$$Pr(\mathbf{f}^J|\mathbf{t}^I) = \sum_{\mathbf{m}^J} Pr(\mathbf{f}^J, \mathbf{m}^J|\mathbf{t}^I) , \tag{3}$$

where

$$
\begin{aligned}
Pr(\mathbf{f}^J, \mathbf{m}^J|\mathbf{t}^I) &= p(\mathbf{m}^J|\mathbf{t}^I, I, J)p(\mathbf{f}^J|\mathbf{m}^J, \mathbf{t}^I, I, J) \\
&= p(J|\mathbf{t}^I) \prod_{j=1}^{J} [p(m_j|\mathbf{m}^{j-1}, J, \mathbf{t}^I, I)p(f_j|\mathbf{f}^{j-1}, \mathbf{m}^J, J, \mathbf{t}^I, I)] ,
\end{aligned}
\tag{4}
$$

where $\mathbf{m}^{i-1} = \langle m_1 \ldots m_{i-1} \rangle$. We also assume a zero-order dependence for both m_j's and f_j's. Note that given the limited amount of data and its sparsity, we make some assumptions in order to reduce the number of parameters. Therefore, we consider $p(J|\mathbf{t}^I)$ only depends on J and m_j is only dependent on the length of the user tag sequence I. We also assume that f_j depends only on t_{m_j}, i.e., the user tag associated to f_j according to the mapping. Notice that since $\mathbf{f}^J$ denotes the sequence of location keywords of size J and $\mathbf{t}^I$ denotes the sequence of user tags of size I, therefore $Pr(\mathbf{f}^J, \mathbf{m}^J|\mathbf{t}^I)$ also depends on the length of both sequences. Consequently, (4) is simplified as follows:

$$Pr(\mathbf{f}^J|\mathbf{t}^I) = p(J) \sum_{\mathbf{m}^J} \prod_{j=1}^{J} p(m_j|I)p(f_j|t_{m_j}) . \tag{5}$$

2.2 Parameter Estimation Based on Expectation-Maximization

Assume that we have N pairs of training samples as in $S = \{(\mathbf{f}_{(1)}, \mathbf{t}_{(1)}), \ldots, (\mathbf{f}_{(n)}, \mathbf{t}_{(n)}), \ldots, (\mathbf{f}_{(N)}, \mathbf{t}_{(N)})\}$, the log-likelihood function for the training samples would be:

$$F(\vartheta) = \sum_{n=1}^{N} \sum_{j=1}^{J_n} \log \sum_{i=0}^{I_n} p(i|I_n)p(f_{jn}|t_{is}) , \tag{6}$$

where $\vartheta := \{p(i|I), p(f|t)\}$ are the free parameters. To solve the parameter estimation problem of (6), we follow the Maximum Likelihood (ML) criterion subject to the constraint $\sum_f p(f|t) = 1$, for each user tag t. We use Lagrange multipliers to make the optimization problem unconstrained.

However, since we introduced hidden variables into our model (Equation (3)), there is no closed-form solution to this optimization problem. Therefore, we follow the iterative process of the *Expectation-Maximization* (EM) algorithm.

To be able to follow the EM algorithm, we first define $Q(\vartheta, \hat{\vartheta})$ as:

$$
\begin{aligned}
Q(\vartheta, \hat{\vartheta}) &= Q\Big(\{p(i|I), p(f|t)\}; \{\hat{p}(i|I), \hat{p}(f|t)\}\Big) \\
&= \sum_{n=1}^{N} \sum_{j=1}^{J_n} \sum_{i=0}^{I_n} \gamma_n(i|j, J_n, I_n) \log\Big\{\hat{p}(i|I_n)\hat{p}(f_{jn}|t_{in})\Big\} \\
&= \sum_{n=1}^{N} \sum_{j=1}^{J_n} \sum_{i=0}^{I_n} \frac{p(i|I_n)p(f_{js}|t_{is})}{\sum_{i'=0}^{I_n} p(i'|I_n)p(f_{js}|t_{i's})} \log\Big\{\hat{p}(i|I_n)\hat{p}(f_{jn}|t_{in})\Big\},
\end{aligned}
\tag{7}
$$

where $\gamma_n(i|j, J_n, I_n)$ is the posterior probability, defined as:

$$
\gamma_n(i|j, J_n, I_n) = \frac{p(i|I_n)p(f_{js}|t_{is})}{\sum_{i'=0}^{I_n} p(i'|I_n)p(f_{js}|t_{i's})} \ .
$$

According to the EM algorithm, we follow an iterative procedure for parameter estimation. After defining the relative objective function, $Q(\vartheta, \hat{\vartheta})$, we follow the usual steps of the algorithm:

(1) E-step: calculate $Q(\vartheta, \hat{\vartheta})$ for all training samples in S with the previous estimate of ϑ.
(2) M-step: optimize $Q(\vartheta, \hat{\vartheta})$ over $\hat{\vartheta}$.

We start the algorithm with uniform values for the parameters and follow the EM steps until convergence.

In the following we describe two possible directions to use the computed mapping.

2.3 Location Keywords Boosting

After finding the optimum mapping between the user tags and location keywords, we aim to use this additional knowledge in our system to address the sparsity problem and eventually enhance the recommendation performance. Take Figure 3 as a real example of such mapping from our dataset. As we can see, 19 taste keywords from one location are illustrated together with tags for the same location by 3 different users . Not surprisingly, the 3 sets of tags have some in common such as "beer" and "cocktails." However, each user has her own personal opinion and therefore her personal set of tags. The lines and the numbers in parentheses represent the result of our proposed mapping for these 3 users. Every mapped item is based on the user personal preference and behavior with respect to all locations in her history. As an example, we take one of the user tags that is common between the three users: "cocktails." What is interesting about this tag is that each user maps it to a different location keyword. For User1 "cocktails" is mapped to "good-for-a-late-night," for User2 to "cocktails" and for User3 to "lemoncello." All three location keywords are good candidates to be mapped to "cocktails" user tag, however, as we argued *each user has her own reasons to tag the same location with a different tag*.

After the observation of Figure 3, we assume that among the I location keywords, we can determine J keywords that are mapped to user tags and presumably are more interesting to the user. As in the example of Figure 3, the number of location keywords ($I = 19$) is much higher than user tags ($J_{user1} = 4$). Therefore, by boosting the mapped location keywords in our model we achieve two main goals: 1) reduce the location keyword space dimensions drastically (e.g., $19 \rightarrow 4$) and; 2) use the valuable information given by the users to detect those location keywords that are more interesting to each users.
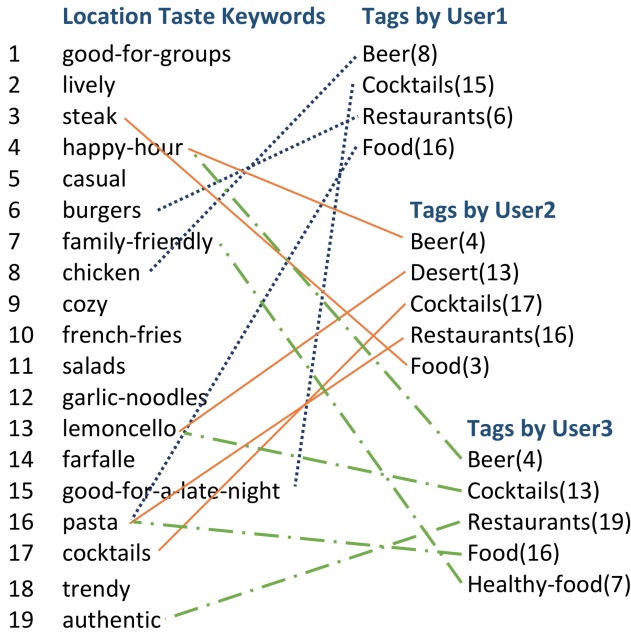
Fig. 3. A sample of tags from three different users assigned to one single location and the calculated mapping. The lines connect each user tag to their mapped location keywords. Also, the index number of the mapped location keywords is written in parentheses for more convenient reading.

Formally, let $\mathbf{f}^I = \langle f_1 \ldots f_I \rangle$ be the set of keywords of a location and $\hat{\mathbf{f}} \in \mathbf{f}^I$ be the set of location keywords which are mapped to user tags. According to the result of our probabilistic mapping, we assume that there is a strong correlation between $\hat{\mathbf{f}}$ and the user's interest. In other words, the keywords in $\hat{\mathbf{f}}$ correlate more to the user's interest as opposed to the other ones in $\mathbf{f}^I$. Hence, we boost $\hat{\mathbf{f}}$ to model the user's interests, reducing the data dimensionality from I to $|\hat{\mathbf{f}}|$. This helps us to address the data sparsity problem. The personalized boosted location keywords are used for POI recommendation (see Section 4).

2.4 User Tag Prediction

As an alternative approach, we explore three models to predict user tags. We utilize the result of our mapping model (**m**) between location keywords and user tags to train a model being able to predict user tags for a new location. We predict user tags for an unseen location as an alternative to keyword boosting. We explore this direction for two reasons: 1) to see how we can predict a user behavior in terms of tag annotation and; 2) to compare the effect of user tag prediction against location keyword boosting to see which strategy is able to enhance the recommendation more effectively. We follow two approaches to predict user tags: 1) we use the maximum likelihood criterion with our estimated parameters to generate the most likely set of user tags given a set of location keywords and; 2) we model the user tag prediction as a sequence labeling problem enabling us to apply different sequence labeling models.

Maximum Likelihood. Here we describe how we follow the Maximum Likelihood to leverage the learned mapping parameters in order to predict user tags for a new POI. As we mentioned in relation to (1) in Section 2.1, given a set of location keywords from Foursquare, $\mathbf{f}^J$, we aim to

compute the most probable set of user tags, $\hat{\mathbf{t}}^I$. Once the model parameters are estimated using EM (see Section 2.2), one approach to predict user tags given a set of location keywords is to follow the maximum likelihood criterion (see (1)) to generate the most likely set of user tags. We search the space of user tag probabilities to find the optimum sequence of user tags following a Viterbi-like algorithm.

Sequence Labeling. In the following, we begin with explaining how we model user tag prediction as a sequence labeling problem. Furthermore, we introduce the set of features we choose to train the tagging models as well as the tagging models we adopt. Assuming we have N sample mapped pairs of user tags and location keywords for each user: $S = \{(\mathbf{f}_{(1)}, \mathbf{t}_{(1)}), \ldots, (\mathbf{f}_{(n)}, \mathbf{t}_{(n)}), \ldots, (\mathbf{f}_{(N)}, \mathbf{t}_{(N)})\}$ with N corresponding mappings. That is, $M = \{\mathbf{m}_{(1)}, \ldots, \mathbf{m}_{(n)}, \ldots, \mathbf{m}_{(N)}\}$. We should model the tag prediction problem as a sequence labeling problem: given a sequence of location keywords we aim to predict the most probable sequence of user tags. In order to do this, we need to adapt the form of the training data.

As in a general sequence labeling problem, we need to assign a label from the target space to each item in the source space. Therefore, we should assign a label to all location keywords even if they are not mapped to any user tag. To this end, we automatically annotate location keywords following these steps:

(1) For each $f_i \in \mathbf{f}_{(n)}$ mapped to a user tag with $\mathbf{m}_{(n)}$, we annotate f_i with its corresponding user tag m_j.
(2) For each $f_i \in \mathbf{f}_{(n)}$ not mapped to a user tag with $\mathbf{m}_{(n)}$, we annotate f_i with "null."

As for the example of Figure 3, a sample training sequence would be as follows:

$$\underbrace{\text{burgers}}_{restaurants}, \underbrace{\text{chicken,}}_{beer} \underbrace{\text{good-for-a-late-night,}}_{cocktails} \underbrace{\text{pasta,}}_{food} \underbrace{\text{good-for-groups,}}_{null} \underbrace{\text{lively,}}_{null} \underbrace{\text{steak,}}_{null} \underbrace{\text{happy-hour,}}_{null} \ldots$$

As we can see, each of the location keywords is annotated with a tag. Therefore, it is straightforward to use this data as training samples for a sequence tagger. We adopt two models as taggers: *Conditional Random Fields* (CRF) [29] and another tagger based on *Support Vector Machines* (SVM) [28]. The main advantage of these models is that they are discriminative. Discriminative tagging models have proven to be more effective for sequence labeling mainly because they normalize the model over the whole training set, resulting in a more generalized model. Furthermore, discriminative tagging models accept a wider range of features, something that is of great importance to some applications.

As features of the sequence labeling models, for a user tag at position j, t_j, we only consider the location keyword at the same position, f_j. That is to follow our assumption that user tags only depend on one location keyword and since we assume that user tags are independent, we use a zero-order tagger model.

In this section, we first introduced a probabilistic framework for finding the mapping between location content (i.e., keywords) and user-annotated tags. Then, we described how this information can be leveraged to reduce the dimensionality of location keywords and hence to address the data sparsity problem. We also explored modeling this problem as a sequence labeling problem and used some state-of-the-art techniques to predict user tags given a new POI's keywords. In Section 4, we show how we use the two mentioned directions to enhance POI recommendation.

3 CONTEXTUAL APPROPRIATENESS PREDICTION

In this section, we first define the problem of predicting the contextual relevance of locations. Then we present the set of features that we use to train the appropriateness classifier and introduce the

Table 1. Description of different contextual information dimensions.

Context	Value	Short Reference	Description
Duration	Day Trip	day-trip	The duration of the trip is one day.
	Night Out	night-trip	The duration of the trip is a night out.
	Weekend Trip	weekend-trip	The trip lasts for a weekend.
	Longer	longer-trip	The trip lasts longer than a weekend.
Group	Alone	alone	The person travels alone.
	Friends	with-friends	The person travels with her friends.
	Family	with-family	The person travels with her family.
	Other	with-others	The person travels with a group other than family and friends.
Type	Business	business-trip	The type of the trip is business.
	Holiday	holiday-trip	The type of the trip is holiday.
	Other	other-trip	The type of the trip is other than holiday and business, e.g., medical.

Table 2. Examples of contextual features generated using crowdsourcing.

Category	Context	$F_{\text{app}}(cat, cxt)$
Beach	Trip type: Holiday	1.0
College & University	Trip duration: Weekend	−1.0
Shop & Service	Trip type: Holiday	0.71
Museum	Trip type: Business	−0.66
Pet Store	Trip duration: Weekend	−0.18
Medical Center	Trip type: Other	0.0

dataset that we collected to train the classifier. The computed contextual relevance scores are then used to re-rank a ranked list of POIs in Section 4.

Let $V = \{v_1, \ldots, v_n\}$ be a set of locations and $C_x = \{cx_1, \ldots, cx_m\}$ a set of contextual descriptors. Our aim is to predict whether it is appropriate for a user to visit a location $v_i \in V$ under a given context C_x. Different contextual dimensions define user's preferences, constraints, or requirements and are listed as follows: **Trip type** (holiday, business, other), **Trip duration** (day trip, night out, weekend trip, longer) and **Group type** (alone, family, friends, other). More information could be found in Table 1. We model the problem as binary classification considering location categories and contextual descriptors as classification features.

3.1 Contextual Features

In this section, we describe the features that we used to train the appropriateness classifier. The degrees of appropriateness between location categories and contextual descriptors constitute our features.

We define a contextual feature function as follows:

Definition 3.1. A **contextual feature**, $F_{app}(cat, cxt)$, is a function determining the relevance of a POI category, cat, to a contextual dimension, cxt. $F_{app}(cat, cxt)$ ranges between -1 and $+1$ with -1 representing absolute inappropriateness and $+1$ absolute appropriateness.

For instance, assume a user wants to visit a location with category *nightlife-spot*, and her context is described as follows: holiday-trip, with-family, weekend-trip. The three features of this example are the appropriateness value between the location category and each of the contextual dimensions. Therefore, the features are F_{app}(nightlife-spot, holiday-trip), F_{app}(nightlife-spot, with-family), and F_{app}(nightlife-spot, weekend-trip).

In many cases, determining such contextual features is intuitive and can be done by one human annotator. However, there are several features that human annotators cannot agree on, like for example F_{app}(office, with-friends), F_{app}(food-and-drink-shop, business-trip), or F_{app}(stadium, night-out-trip). Hence, we define two classes of features: objective and subjective. *Objective* features are those that the annotators quickly agree on. This suggests that a user would be very likely to agree with the annotators on the objective features. Therefore, we can conclude that objective features potentially influence a user's decision of visiting a location. As in the previous example, supposedly, everyone would consider going to a nightlife spot with family is *not* appropriate. Thus a user who regularly goes to nightlife spots might change her mind when she is traveling with her family. As we saw in this example, such *objective* features can directly change users' decisions adding contextual constraints to the model. *Subjective* features, in contrast, have less impact as they mainly depend on the user's opinion and personal preferences. If the annotators did not agree on a feature, we would not be able to predict a user's opinion. Therefore, we cannot predict the influence of subjective features on a user's decision.

We determined the level of subjectivity or objectivity of features via a crowdsourcing task. In the task, we asked the workers to judge if a location category is appropriate for a context descriptor (e.g., cat = nightlife-spot and cxt = with-family). We asked at least five different assessors to judge each category-context pair. In the context of this paper, we define those pairs with high agreement rate between the workers as *objective*, while we consider those lacking assessors agreement as *subjective*. More details on how we created the dataset can be found in our previous work [8].

Table 2 lists some example features from our dataset. As we can see in this table, lower values for $|F_{app}(cat, cxt)|$ mean that the features are more subjective. We created the contextual features for all pairs of 11 contextual dimensions and the 177 most frequent categories of Foursquare category tree[8]. Overall we generated 1,947 contextual features from 11,487 judgments[9].

3.2 Training the Classifier

As described earlier, we formulate contextual appropriateness as binary classification. In Section 3.1 we explained how we created the contextual features. Here, we describe another dataset for training the appropriateness classifier using our features. We randomly selected 10% of the data from TREC-CS 2016 dataset. We created another crowdsourcing task for annotating the training data. We asked workers to assess if it is appropriate that a user with a full description of context (e.g., Holiday, Friends, Weekend) to visit a location category (e.g., Bar). Each instance in the dataset is considered as appropriate only if at least two of the three assessors voted for their appropriateness. We train the contextual appropriateness classifier on 10% of the data from TREC-CS 2016 to predict the remaining 90% of TREC-CS 2016 and the whole TREC-CS 2015 dataset. We applied a wide range of classifiers for this task. However, we only report the best results that were obtained with

---

[8]https://developer.foursquare.com/categorytree
[9]The dataset if freely available on request.

SVM [17]. The predicted contextual relevance score is denoted as S_{cxt} and we use it to re-rank the personalized location ranking (see Section 4).

4 RECOMMENDATION BASED ON INFORMATION FROM MULTIPLE LOCATION-BASED SOCIAL NETWORKS

After explaining the two major components of our proposed approach in Sections 2 and 3, here, we explain our way of performing POI recommendation, exploiting the scores from multiple LBSNs. We describe two sets of scores: the frequency-based and review-based scores. We use the frequency-based score to incorporate the boosted keywords (Section 2.3), the predicted user tags (Section 2.4), as well as other types of information. We also demonstrate how we combine different scores to produce the personalized location ranking using learning to rank.

4.1 Frequency-based Score

We base the frequency-based scores on the assumption that users prefer the type of locations that they like more frequently and rate them positively[10]. Therefore, we create positive and negative profiles considering the content of locations in the user's check-in history and calculate the normalized frequencies as they appear in her profile. Then we compute a similarity score between the user's profile and a new location. For simplicity, we only explain how to calculate the frequency-based score using location categories. The method can be easily generalized to calculate the score for other types of information.

Let u be a user and $h_u = \{v_1, \ldots, v_n\}$ her history of check-ins. Each location has a list of categories $C(v_i) = \{c_1, \ldots, c_k\}$. We define the user category profile as follows:

Definition 4.1. A **Positive-Category Profile** is the set of all unique categories belonging to locations that user u has previously rated positively. A **Negative-Category Profile** is defined analogously for locations that are rated negatively.

Each category in the positive/negative category profile is assigned with a user-level normalized frequency. We define the user-level normalized frequency for a category as follows:

Definition 4.2. A **User-level Normalized Frequency** for an item (e.g., category) in a profile (e.g., positive-category profile) for user u is defined as:

$$\mathrm{cf}_u^+(c_i) = \frac{\sum_{v_k \in h_u^+} \sum_{c_j \in C(v_k), c_j = c_i} 1}{\sum_{v_k \in h_u} \sum_{c_j \in C(v_k)} 1} \, ,$$

where h_u^+ is the set of locations that u rated positively. We calculate user-level normalized frequency for negative categories, cf_u^-, analogously.

We create positive/negative category profiles for each user based on Definitions 4.1 and 4.2 . Given a user u and candidate location v, the frequency-based similarity score based on location categories, $S_{cat}(u, v)$, is calculated as follows:

$$S_{cat}(u, v) = \sum_{c_i \in C(v)} \mathrm{cf}_u^+(c_i) - \mathrm{cf}_u^-(c_i) \, . \tag{8}$$

We follow the same procedure to calculate a frequency-based score based on other types of information as listed below:

- S_{key}: We consider location taste keywords (instead of categories) to compute the similarity score between a given user's profile and a candidate location.

---

[10]We consider reviews with rating [4, 5] as positive, 3 as neutral, and [1, 2] as negative.

- S_{boost}: As for boosting, for each user we follow Definition 4.1 to create positive and negative *boosted location taste keyword profiles*. We consider only the location taste keywords that are mapped to user tags (see Section 2.3). Given a candidate location, we calculate boosted keywords similarity score according to Definition 4.2 and (8).
- S_{ml}: Following Definitions 4.1 and 4.2, we create positive and negative *user tag profiles* for each user. However, since a candidate location does not naturally have user-assigned tags, we use our ML-based approach to predict user tags. The predicted user tags are then compared with the user's profile following (8) to calculate S_{ml} similarity score.
- S_{crf}: We calculate S_{crf} score similar to S_{ml}. We predict user tags for a candidate location using the trained CRF model. Then we follow (8) to calculate the S_{crf} score comparing predicted user tags with user profile.
- S_{svm}: This score is also calculated like S_{ml}. For a candidate location, we predict user tags using our trained SVM-based tagging model. The predicted user tags are then compared with the user's profile using (8) resulting in S_{svm}.

4.2 Review-Based Score

Modeling a user only on locations' content is general and does not determine why the user enjoyed or disliked a POI. The content of locations is often used to infer "which type" of POIs a user likes. On the other hand, reviews express the reasons of users' ratings [39]. Since there could be a lack of explicit reviews from the user, we tackle this sparsity problem using reviews of other users who gave a similar rating to the location. We follow the same idea of Yang et al. [39], that is, a user's opinion regarding a location could be learned based on the opinions of other users who rated the same location similarly.

We calculate the review-based score using a binary classifier. We model this problem as binary classification since a user, before visiting a new city or location, would get a positive or negative impression of the location after reading the online reviews of other users. We assume that a user compares the characteristics of a location and the opinions which are expressed by other users in their reviews to her expectations and interests. A user would be convinced of visiting a particular location if the reviews satisfy her expectations up to a certain point. An alternative to binary classification would be a regression model. However, we assume that users behave similarly to a binary classifier when they read online reviews before deciding on whether to visit a venue or not. For example, assume a user reads a few positive and negative online reviews about a POI and measures how similar the mentioned qualities are to her expectations. Finally, depending on the balance between the positive remarks and the negative ones, she makes a binary decision (i.e., whether to go or not). We see this behavioral pattern similar to that of a binary classifier: it learns from the positive and negative samples and compares the learned parameters with a test sample and assigns its label accordingly. Furthermore, due to data sparsity, grouping ratings as positive and negative aids us to model users more effectively.

We train binary classifier using the reviews from the locations in a user's check-in history. The positive training samples for user u are positive reviews of locations that were liked by u. Likewise, the negative reviews of locations that u disliked constitute the negative training samples. We decided to ignore the negative reviews of liked locations and positive reviews of disliked locations since they are not supposed to contain any useful information.

We consider TF-IDF score of terms in reviews as features. We trained many classifiers but SVM outperformed all other models. Therefore, we choose SVM and consider the value of its decision function as the review-based score and refer to it as S_{rev}. The decision function gives us an idea on how relevant a location is to a user profile.

Table 3. Four proposed models using different combination of similarity scores.

	Category	Review	Keywords	Context
UT-ML	S_{cat}	S_{rev}	S_{key}, S_{ml}	S_{cxt}
UT-CRF	S_{cat}	S_{rev}	S_{key}, S_{crf}	S_{cxt}
UT-SVM	S_{cat}	S_{rev}	S_{key}, S_{svm}	S_{cxt}
PK-Boosting	S_{cat}	S_{rev}	S_{key}, S_{boost}	S_{cxt}

4.3 Location Ranking

After defining the mentioned relevance scores, here we explain how we combine them. Given a user and a list of candidate locations, we calculate the mentioned scores for each location and combine them to create a ranked list of locations. We adopt several learning to rank[11] techniques to rank the candidate locations since they have proven to be effective for similar tasks [32, 39]. In particular, we examine the following learning to rank techniques: AdaRank, CoordinateAscent, RankBoost, MART, LambdaMART, RandomForest, RankNet, and ListNet. We introduce four models using different combinations of the scores as mentioned in Table 3.

5 EXPERIMENTAL SETTINGS

In this section, we present the experimental settings including the datasets we used, compared methods, and evaluation process.

5.1 Datasets

Recommendation Effectiveness. We evaluate our approach on two benchmark collections, published by TREC. The collections are those used in the *Batch Experiments/Phase 2* of the TREC-CS track 2015 [18] and 2016 [25]. The task was to rank a list of candidate locations in a new city for a user, given her history of check-ins in other cities. The datasets were collected using crowdsourcing where each user rated 30 to 60 locations in one or two cities. In addition, each user may have tagged locations to explain why she likes them (i.e., user tags). Later, the same users were called to rate new POIs in another city as well as the contextual factors of their trip. To get more information about locations, we crawled Yelp and Foursquare for reviews, categories, and taste keywords[12]. More specifically, for each location in the dataset, we formed a query from the location's name and city to find the corresponding Yelp and Foursquare profiles. To avoid irrelevant information, we verified the title and location of each result. Yelp is crawled mainly for reviews, whereas Foursquare mainly for location taste keywords. It is worth noting that we were able to find additional information for most of the locations from both LBSNs. Table 4 lists the key attributes of our crawled dataset. More details on the crawling process and structure of our dataset can be found in our previous work [8].

Dimensionality Reduction. We evaluate the dimensionality reduction effectiveness on the same datasets as we do for recommendation effectiveness. We compare the performance of our proposed model to a well-known dimensionality reduction model in terms of recommendation effectiveness. Therefore, we use the same datasets used to evaluate recommendation effectiveness, however, we provide more details and discussion related to dimensionality reduction.

User Tag Prediction. Since the test set in TREC-CS 2015 and TREC-CS 2016 do not include user tags for locations, we need to evaluate the user tag prediction on the training datasets. Therefore, we

---

[11]We use RankLib implementation of learning to rank: https://sourceforge.net/p/lemur/wiki/RankLib/
[12]The dataset is available on request.

Table 4. Statistical details of datasets

	TREC-CS 2015	TREC-CS 2016
Number of requests	211	442
Number of requests evaluated by TREC	211	58
Number of locations	8,794	18,808
Number of locations crawled from Yelp	6,290	13,604
Number of locations crawled from Foursquare	5,534	13,704
Average reviews per location	117.34	66.82
Average categories per location	1.63	1.57
Average taste keywords per location	8.73	7.89
Average user tags per user	1.46	3.61
Number of distinct user tags	186	150

Table 5. Statistical details of user tagging dataset

	Training Set	Test Set
Number of instances	20,148	5,037
Number of non-null tags	102,667	25,541
Number of null tags	54,444	13,954
Number of unique user tags	156	121
Number of unique location keywords	2,676	1,398
Average user tags per location	1.85	1.82
Average keywords per location	5.10	5.07

randomly split the TREC-CS 2015 and TREC-CS 2016 training sets into: training, development, and test set. We train the taggers using the new training set, tune them using the new development set and evaluate them with the new test set. As part of the evaluating recommendation effectiveness we show how different taggers can improve the recommendation; however, the aim of this experiment is to show how accurately we can model the user interests and tagging behavior. The statistical details of the tags dataset is listed in Table 5.

5.2 Compared Methods

Recommendation Effectiveness. We consider the best performing system in TREC-CS 2015 as our baseline. Moreover, we compare our proposed method with state-of-the-art context-aware POI recommendation methods. We also compare our proposed PK-Boosting with other models based on user tag prediction (i.e., UT-ML, UT-CRF, and UT-SVM).

- *LinearCatRev* is our previous work [6] which is the best performing model of TREC-CS 2015. It extracts information from different LBSNs and uses it to calculate category-based and review-based scores. Then, it combines the scores using linear interpolation. We choose this baseline for two reasons, firstly because it is the best performing system of TREC-CS 2015, and secondly because it also uses scores derived from different LBSNs.
- *GeoSoCa* exploits geographical, social, and categorical correlations for POI recommendation [45]. GeoSoCa models the geographical correlation using a kernel estimation method with an adaptive bandwidth determining a personalized check-in distribution. It models

the categorical correlation by applying the bias of a user on a POI category to weigh the popularity of a POI in the corresponding category modeling the weighted popularity as a power-law distribution. We used the implementation of GeoSoCa released in [34]. We did not include the social correlation component since such information does not exits in the datasets.

- *n-Dimensional Tensor Factorization (nDTF)* [27] generalizes matrix factorization to allow for integrating multiple contextual features into the model. We used the publicly available implementation of nDTF[13]. Regarding the features, we include two types of features: (1) location based: category, keywords, average rating on Yelp, and number of ratings on Yelp (as an indicator of its popularity); (2) user based: age group and gender.

- *UT-ML* differs from PK-Boosting in one score. For UT-ML, instead of the keyword boosting score (S_{boost}), we use the score based on the predicted user tags following maximum likelihood criterion. As we described in Section 2.4, we also explored three different models as alternatives to PK-Boosting. Our aim is to study the impact of predicting user tags on recommendation effectiveness, compared to PK-Boosting. The other two alternative approaches are listed as follows.

- *UT-CRF* predicts user tags using a trained CRF model. Then, for each venue-user pair, it computes the similarity between the predicted user tags and the user profile. Finally, it replaces the boosting score with the computed similarity score (see Section 2.4). We used CRFSuite[14] implementation of CRF.

- *UT-SVM* predicts user tags given a user-venue pair using an SVM-based tagging model. The boosting score is replaced by the similarity score between the user profile and predicted user tags (see Section 2.4). We used YamCha[15] implementation of the SVM-based tagging model.

Dimensionality Reduction. In order to evaluate the keyword boosting approach from the perspective of dimensionality reduction, we also apply the following well-known dimensionality reduction method to reduce the location keywords dimension. In particular, we use *PK-PCA*. PK-PCA uses *Principal Component Analysis* (PCA) to reduce the dimensionality of location keywords. The corresponding score of keyword boosting is replaced by the score computed based on PCA.

User Tag Prediction. As user tags contain very crucial information explicitly described by users, we aim to evaluate the effectiveness of user tag models. In particular, we evaluate the following models:

- *Conditional Random Fields (CRF) Tagger* [29] models the sequence tagging problem in a discriminative manner. The tagger is based on binary features that are extracted from the text and optimized for the training data.

- *SVM-based Tagger* [28] is also a discriminative approach that trains one SVM classifier per tag. The model is an ensemble of all SVM classifiers.

5.3 Evaluation Metrics

We evaluate the recommendation effectiveness as well as the dimensionality reduction for the top-k recommendation. We also evaluate the effectiveness of user tag prediction methods.

Recommendation Effectiveness. In both TREC-CS 2015 and TREC-CS 2016 datasets, for each user u, the data, $S(u)$, is split into two sets: a number of locations visited in one or two cities constitute the training set and a number of locations in a new city constitute the test set. Given

---

[13]https://github.com/VincentLiu3/TF
[14]http://www.chokkan.org/software/crfsuite/
[15]http://chasen.org/~taku/software/yamcha/

a user u, if a recommended location in the test set is marked by the user as *relevant*, it is a "hit," otherwise it is a "miss." To perform a fair comparison, we choose the official evaluation protocol and metrics of TREC-CS for this task, which are P@5 (Precision at 5), nDCG@5 (Normalized Discounted Cumulative Gain at 5), and MRR (Mean Reciprocal Rank). Since the main focus in this task was to improve the location rankings, such evaluation metrics serve as perfect metrics.

The relevance assessments for test sets are slightly different in TREC-CS 2015 and TREC-CS 2016. In TREC-CS 2015 relevance of a location to a user is defined with a binary value with 0 as irrelevant and 1 as relevant, whereas in TREC-CS 2016 users rated locations in the range of -2 to $+2$. This also explains why the main evaluation metric in TREC-CS 2015 is P@5 as opposed to nDCG@5 in TREC-CS 2016. Both P@k and nDCG@k metrics are evaluated over k top locations on the ranked list. Let U be the set of users and r_u^p be the rating score assigned by user u to the location at the ith rank of the list. Precision and nDCG values are calculated at the kth position as follows:

$$P_u@k = \frac{\#hits_u@k}{k} \ ,$$

$$nDCG_u@k = Z_u \sum_{i=1}^{k} \frac{2^{r_u^i} - 1}{\log(1+i)} \ ,$$

where u is the given user, Z_u is a normalization factor and $\#hits_u@k$ is the number of relevant locations for user u in the top-k locations of the ranked list. nDCG@k and P@k are the mean of nDCG$_u$@k and P$_u$@k over U respectively. MRR is also calculated as follows:

$$MRR = \frac{1}{|U|} \sum_{u=1}^{|U|} \frac{1}{rank_u} \ ,$$

where $rank_u$ is the ranking of the first relevant location for user u. We conduct a 5-fold cross validation on the training data to tune our model. We determine the statistically significant differences using the two-tailed paired t-test at a 95% confidence interval ($p < 0.05$).

Dimensionality Reduction. Since there is no ground truth data to evaluate dimensionality reduction methods, we evaluate the recommendation effectiveness using different dimensionality reduction methods to see how they enhance the overall recommendation. Therefore, we use the same evaluation metrics that we used for evaluating recommendation effectiveness.

User Tag Prediction. Since we modeled the user tag prediction problem as a sequence-labeling problem, we evaluate the effectiveness of user tag prediction using the same metrics used for evaluating typical sequence-labeling problems such as *Part of Speech* (POS) tagging. Therefore, we report Precision, Recall, and F-Measure for this experiment. Let T_p the number of true positive and F_p the number false positive non-null predicted tags. Then precision is calculated as follows:

$$Precision = \frac{T_p}{T_p + F_p} \ .$$

Given the number of false negatives, F_n, we also calculated recall as follows:

$$Recall = \frac{T_p}{T_p + F_n} \ .$$

F-measure is then defined as follows:

$$F - Measure = 2 \times \frac{Precision \times Recall}{Precision + Recall} \ .$$

6 EXPERIMENTAL RESULTS

In this section, we first present the results for recommendation effectiveness. We also show the results for location keyword dimensionality reduction and user tag prediction. Furthermore, we study the effect of different sources and scores on recommendation effectiveness.

6.1 Recommendation Effectiveness

Tables 6 and 7 demonstrate the performance of our approach compared with other methods for the TREC-CS 2015 and TREC-CS 2016 datasets, respectively. We choose the best performing learning to rank technique for each model we adopt the best performing learning to rank technique according to Tables 8 and 9. It is worth noting that the best learning to rank technique for PK-Boosting is ListNet [14]. Tables 6 and 7 show that PK-Boosting outperforms the competitors in terms of the three evaluation metrics. This indicates that the proposed PK-Boosting approach improves the performance of POI recommendation. This happens because the proposed approach for boosting location keywords addresses the data sparsity problem, while at the same time it captures user preferences more accurately. In contrast, the models UT-ML, UT-CRF, and UT-SVM introduce a prediction error, when predicting user tags for a candidate location. This error is then propagated to location ranking and subsequently degrades the models' performances. As we can see, GeoSoCa and nDTF exhibit the worst performance among all compared methods. This happens mainly because these methods rely on user-POI check-in associations among the training and test sets. In other words, there should be enough common POIs appearing in both the training and test sets, otherwise they fail to recommend unseen POIs. Hence, they suffer from the high level of sparsity on these datasets. In particular, the intersection of POIs in the training and test sets is 771 (out of 8,794) and 4 (out of 18,808) in TREC-CS 2015 and 2016, respectively.

To compute the review-based classifier, we used various classifiers such as Naïve Bayes and k-NN; however, the SVM classifier exhibited a better performance by a large margin. The SVM classifier is a better fit for this problem since it is more suitable for text classification, which is a linear problem with weighted high dimensional feature vectors. Also, we observed a significant difference between the number of positive reviews and negative reviews per location. Generally, locations receive more positive reviews than negative reviews and, in our case, this results in a unbalanced training set. Most of the classification algorithms fail to deal with the problem of unbalanced data. This is mainly due to the fact that those classifiers try to minimize an overall error rate. Therefore, given an unbalanced training set, the classifier is usually trained in favor of the dominant class to minimize the overall error rate. However, SVM does not suffer from this, since it does not try to directly minimize the error rate but instead tries to separate the two classes using a hyperplane maximizing the margin. This makes SVM more intolerant of the relative size of each class. Another advantage of linear SVM is that the execution time is very low and there are very few parameters to tune.

Impact of Different Learning to Rank Techniques. In this experiment we aim to show how the recommendation effectiveness is affected by applying different learning to rank techniques to combine the scores. Tables 8 and 9 report P@5 applying different learning to rank techniques for TREC-CS 2015 and TREC-CS 2016 respectively. We report the performance for UT-ML, UT-CRF, UT-SVM, and PK-Boosting. As we can see, ListNet in many cases outperforms other learning to rank techniques. More specifically, for TREC-CS 2015, ListNet exhibits the best performance for all models except for UT-ML. It is very interesting that RankNet exhibits the best performance for UT-ML, and both ListNet and RankNet are based on artificial neural networks. As for TREC-CS 2016, RankNet performs better for UT-ML and UT-SVM while ListNet performs better for other models. As we can observe, applying different learning to rank techniques can potentially have a

Table 6. Performance evaluation on TREC-CS 2015. Bold values denote the best scores and the superscript *
denotes significant differences compared to LinearCatRev. Δ values (%) express the relative improvement,
compared to LinearCatRev. For each model we report the scores using the best learning to rank technique
(Table 8).

	P@5	Δ(%)	nDCG@5	Δ(%)	MRR	Δ(%)
LinearCatRev	0.5858	-	0.6055	-	0.7404	-
GeoSoCa	0.5147*	−12.14	0.5404*	−10.75	0.6918*	−6.56
nDTF	0.5232*	−10.96	0.5351*	−11.63	0.6707*	−9.41
UT-ML	0.6224*	6.25	0.6320*	4.38	0.7496	1.24
UT-CRF	0.6249*	6.67	0.6285	3.80	0.7434	0.41
UT-SVM	0.6219*	6.16	0.6339*	4.69	0.7553	2.01
PK-Boosting	**0.6259***	**6.85**	**0.6409***	**5.85**	**0.7704***	**4.05**

Table 7. Performance evaluation on TREC-CS 2016. Bold values denote the best scores and the superscript *
denotes significant differences compared to LinearCatRev. For each model we report the scores using the
best learning to rank technique (Table 9).

	P@5	Δ(%)	nDCG@5	Δ(%)	MRR	Δ(%)
LinearCatRev	0.4897	-	0.3213	-	0.6284	-
GeoSoCa	0.4207*	−14.09	0.2958	−7.94	0.6497	3.39
nDTF	0.4172*	−14.80	0.2663*	−17.12	0.6167	−1.86
UT-ML	0.5138	4.92	0.3357	4.48	0.6389	1.67
UT-CRF	0.5138	4.92	0.3410	6.13	0.6765	7.65
UT-SVM	0.5207	6.33	0.3389	5.48	0.6510	3.60
PK-Boosting	**0.5310**	**8.43**	**0.3526***	**9.74**	**0.6800**	**8.21**

big impact on recommendation results. Therefore, it is critical to apply the best technique for the
scores.

Impact of Using Information from Multiple LBSNs. Tables 10 and 11 evaluate the perfor-
mance of the examined models before and after removing information from each LBSN. In this
set of experiments, we also report the relative performance drop of different models when using
information from the two different LBSNs. As we can see in almost all cases, when a source of
information is removed from the model, we observe a drop in the performance. The average drop
for TREC-CS 2015 is −4.90% and for TREC-CS 2016 is −6.00% which confirms the effectiveness of
exploiting information from different LBSNs. This indicates that using multimodal information
from different LBSNs is a key to improve POI recommendation. For all different runs, the best
performing method is the proposed PK-Boosting, that uses a combination of information from both
LBSNs.

Impact of Using Different Scores. In this experiment, we try to demonstrate the effectiveness
of each score. We remove each score and analyze our model's performance without it (but we do
not remove more than one score at a time). The results are reported in Table 12. The first line (*All*)
shows the results for P@5, nDCG@5, and MRR using all scores. The second line ($−S_{cat}$) shows the
results without the location categories, and so on for the other lines.

Table 8. Effect on P@5 for different learning to rank techniques in TREC-CS 2015. Bold values denote the best learning to rank technique per model.

	UT-ML	UT-CRF	UT-SVM	PK-Boosting
MART	0.5911	0.6008	0.5958	0.6010
RankNet	**0.6224**	0.6190	0.6155	0.6190
RankBoost	0.6030	0.6086	0.6088	0.6146
AdaRank	0.6028	0.6121	0.6117	0.5893
CoordinateAscent	.06115	0.5858	0.5918	0.5997
LambdaMART	0.6022	0.6077	0.6061	0.6135
ListNet	0.6069	**0.6249**	**0.6219**	**0.6259**
RandomForests	0.5836	0.5966	0.5920	0.5963

Table 9. Effect on P@5 for different learning to rank techniques in TREC-CS 2016.

	UT-ML	UT-CRF	UT-SVM	PK-Boosting
MART	0.4653	0.4103	0.3931	0.4483
RankNet	**0.5138**	0.5103	**0.5237**	0.5103
RankBoost	0.3414	0.4241	0.4345	0.4586
AdaRank	0.3414	0.3414	0.3414	0.3414
CoordinateAscent	0.5021	0.4931	0.4931	0.5000
LambdaMART	0.3793	0.3931	0.3793	0.4931
ListNet	0.5103	**0.5138**	0.5103	**0.5310**
RandomForests	0.4207	0.4069	0.4345	0.4310

The results show a decrease of the model's performance after removing each of the scores exhibiting an average relative drop of -4.31%. It indicates that our system is able to capture different aspects of information and combine them to create a better personalized ranking model for POI recommendation. The S_{cat} score models the types of locations a user is interested in visiting, while S_{rev} models the reasons the user likes/dislikes different locations belonging to the same category. S_{key} tries to incorporate the most important keywords extracted from the reviews and to describe a location and its characteristics. S_{boost} boosts the most important keywords that interest a user and the contextual relevance is measured by S_{cxt}. Our model exhibits its largest decrease in performance when S_{rev} is removed from the model. This suggests that the review-based score is the most important score in our model. We think this is because it captures users' opinions. In fact, it is crucial to realize why a user rates two locations in the same category differently.

Impact of Number of Visited Locations. We report P@5 of all models on TREC-CS 2015 and TREC-CS 2016 in Figure 4. In this set of experiments, we vary the number of locations to find the mapping between the taste keywords and the user tags. We calculate the scores of Section 4 with different number of locations and train the ranking model. Figure 4 shows that PK-Boosting is the winning method when compared with other models for all different number of locations. This result indicates that PK-Boosting is more robust when the training set is smaller, whereas the prediction models ML and SVM are not very well trained using such a small data and their performance gets worse.

Table 10. Performance evaluation after removing information provided by Foursquare (F) and Yelp (Y) in the TREC-CS 2015 dataset. The superscript * denotes significant differences compared to the performance each model has when using information from the two different LBSNs. Δ values (%) express the relative drop, compared to the performance each model has when using information from the two different LBSNs. (Average drop= −4.90%)

	F	Y	P@5	Δ(%)	nDCG@5	Δ(%)	MRR	Δ(%)
LinearCatRev	✓	✓	0.5858	-	0.6055	-	0.7404	-
	✓	✗	0.5649	−3.57	0.5860	−3.22	0.7263	−1.90
	✗	✓	0.5697	−2.75	0.5917	−2.28	0.7341	−0.85
UT-ML	✓	✓	0.6224	-	0.6320	-	0.7496	-
	✓	✗	0.5288*	−15.04	0.5307*	−16.03	0.6487*	−13.46
	✗	✓	0.5787*	−7.02	0.5746*	−9.08	0.6833*	−8.84
UT-CRF	✓	✓	0.6249	-	0.6285	-	0.7434	-
	✓	✗	0.5960*	−8.95	0.5930*	−5.65	0.7301	−1.79
	✗	✓	0.6055	−3.10	0.6238	−0.75	0.7503	0.93
UT-SVM	✓	✓	0.6219	-	0.6339	-	0.7553	-
	✓	✗	0.5728*	−7.90	0.5921*	−6.59	0.7388	−2.18
	✗	✓	0.6129	−1.45	0.6250	−1.40	0.7497	−0.74
PK-Boosting	✓	✓	**0.6259**	-	**0.6409**	-	**0.7704**	-
	✓	✗	0.5731*	−8.44	0.6010*	−6.23	0.7602	−1.32
	✗	✓	0.6044	−3.44	0.6227	−2.84	0.7613	−1.18

Table 11. Performance evaluation after removing information provided by Foursquare (F) and Yelp (Y) in the TREC-CS 2016 dataset. The superscript * denotes significant differences compared to the performance each model has when using information from the two different LBSNs. (Average drop= −6.00%)

	F	Y	P@5	Δ(%)	nDCG@5	Δ(%)	MRR	Δ(%)
LinearCatRev	✓	✓	0.4897	-	0.3213	-	0.6284	-
	✓	✗	0.4172*	−14.49	0.2705*	−15.81	0.6222	−0.99
	✗	✓	0.4759	−2.46	0.3072	−4.39	0.6032	−4.01
UT-ML	✓	✓	0.5138	-	0.3357	-	0.6389	-
	✓	✗	0.4862	−5.37	0.3079	−8.28	0.6038	−5.49
	✗	✓	0.5034	−2.02	0.3313	−1.31	0.6393	0.06
UT-CRF	✓	✓	0.5138	-	0.3410	-	0.6765	-
	✓	✗	0.5069	−1.34	0.3336	−2.17	0.6531	−3.46
	✗	✓	0.4793	−6.71	0.3133	−8.12	0.6268	−7.35
UT-SVM	✓	✓	0.5207	-	0.3389	-	0.6510	-
	✓	✗	0.4724	−9.28	0.3057*	−9.80	0.6260	−3.84
	✗	✓	0.4793	−7.95	0.3158	−6.82	0.6512	0.03
PK-Boosting	✓	✓	**0.5310**	-	**0.3526**	-	**0.6800**	-
	✓	✗	0.4793*	−9.74	0.3210	−8.39	0.6542	−3.79
	✗	✓	0.4759*	−10.38	0.3177*	−9.90	0.6354	−6.56

Table 12. Performance of PK-Boosting using all the scores (*All*) and after removing each score at a time. The superscript * denotes significant differences compared to the model using all scores (*All*). Percentages in bold represent the highest decrease in performance when the corresponding score is removed (Average relative drop = −4.31%).

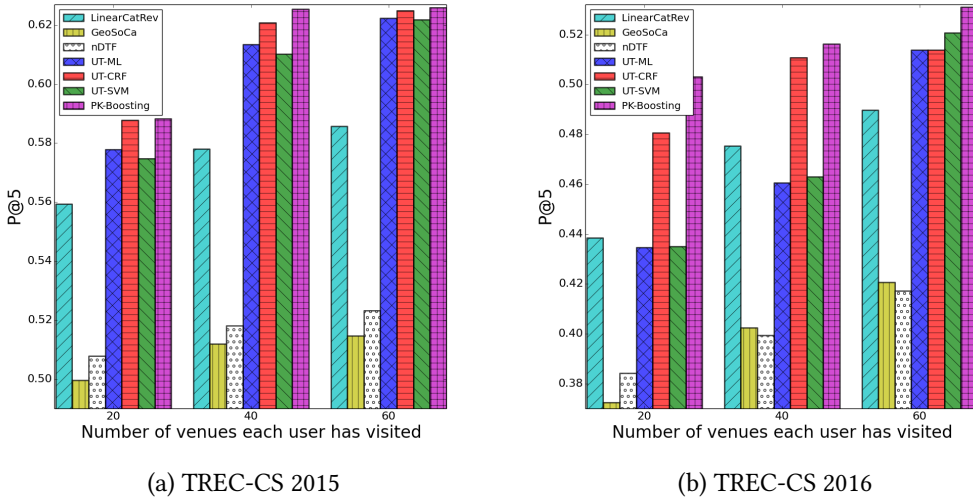|	P@5	$\Delta(\%)$	nDCG@5	$\Delta(\%)$	MRR	$\Delta(\%)$
All	0.6259	-	0.6409	-	0.7704	-
$-S_{cat}$	0.6009*	−3.99	0.6124*	−4.45	**0.7324***	**−4.93**
$-S_{rev}$	**0.5555***	**−11.25**	**0.5837***	**−8.92**	0.7383*	−4.17
$-S_{key}$	0.6009*	−3.99	0.6113	−3.35	0.7443	−3.10
$-S_{boost}$	0.6190	−1.10	0.6312	−1.51	0.7610	−1.22
$-S_{cxt}$	0.5962*	−4.75	0.6126*	−4.42	0.7437	−3.47

(a) TREC-CS 2015   (b) TREC-CS 2016

Fig. 4. Effect on P@5 by varying the number of locations that each user has visited for (a) TREC-CS 2015 and (b) TREC-CS 2016.

Impact of Visiting POIs from a Single City vs. Two Cities. In this experiment, we intend to see how the number of visited locations from one single city affects the performance of our model as opposed to the same number of locations from different cities. In order to do that, we consider at maximum 2 cities, and we train our model using 10, 15, ..., 60 of them for each user as their history of preferences. To make sure that the order of selected locations does not affect our experiment, we shuffle the list of previously visited locations in two ways: 1) we make sure that the first 30 locations are from a city and the second 30 are from the other city; 2) we shuffle the order of visited locations for each city and interleave them. For example, v_1 would be a location from City1, v_2 a location from City2, v_3 a location from City1, and so on. We conduct this experiment with 5 differently shuffled lists and report the average of the results.

The first ordering method ensures that the first half of the locations, visited by a user, are from a particular city. The second ordering, on the other hand, makes sure that for a given number of visited locations n, $n/2$ of them are locations from City1 and $n/2$ are from City2 . We intend to
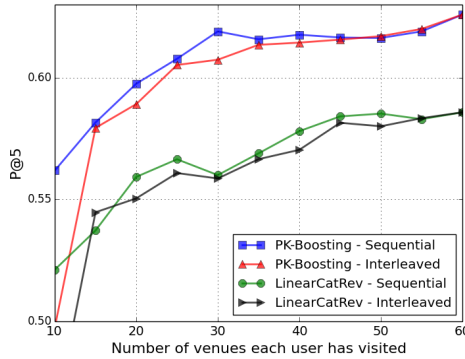
Fig. 5. Our model's performance in terms of P@5 with different number of locations as users' history of preferences compared to LinearCatRev. We have chose the order of locations in two different manners. *Sequential*: the first 30 locations are from one single city, the second 30 are from another city, *Interleaved*: the list of locations is interleaved based on their cities.

examine how our model performs when we have information about users from only one city as opposed to two cities. Moreover, in cases where we have a low number of visited locations in the user's history, it is interesting to see how our model performs in both scenarios (i.e., all locations from a single city vs. from multiple cities).

Figure 5 demonstrates our system performance in terms of P@5 with different number of locations in the user's history compared to LinearCatRev. As we can see in the figure, the model shows a large improvement up to the first 30 locations, decreasing in size after we add 40 locations in both orderings. However, it is interesting to see that the sequential order always performs better than the interleaved one. This difference is more evident when the number of locations is smaller than 20. It suggests that when we have limited number of locations as the user's history, it is better to have them all about the same city. This can be observed when there are 30 locations and all from one single city (denoted as *Sequential*), we get a much better performance as compared to training the model using 30 locations with half from one city and the other half from another one (denoted as *Interleaved*).

6.2 Dimensionality Reduction

In this experiment we compare our personalized keyword boosting method with the well-known dimensionality reduction method PCA. Since keyword boosting is a kind of personalized dimensionality reduction, we choose to compare our method with PCA. As Tables 13 and 14 show, our personalized keyword boosting method is able to beat PCA with respect to recommendation effectiveness in terms of both $P@k$ and $nDCG@k$. It suggests that the proposed probabilistic model is able to effectively reduce the dimensionality of location keywords taking into account user personal preferences as well as interests. In fact, in TREC-CS 2015 the average location keyword per user is 277 and our proposed approach is able to reduce it to 41 ($-\%85$), whereas PCA reduces it to 25 ($-\%91$). Moreover, the average location keywords per user in TREC-CS 2016 is 302 and PK-Boosting reduces it to 105 ($-\%65$) compared with PCA reducing it to 16 ($-\%95$).

It is worth noting that PCA produces $\min(n, m-1)$ principal components, where n is the number of data samples and m is the number of data dimensions. Since in both datasets $n \ll m$, the number of principal components is bounded by n and thus PCA reduces data dimensionality more than

Table 13. Performance comparison on TREC-CS 2015 on dimensionality reduction. The superscript * denotes significant differences compared to LinearCatRev.

	Avg. Dim.	P@5	Δ(%)	nDCG@5	Δ(%)	MRR	Δ(%)
LinearCatRev	277	0.5858	-	0.6055	-	0.7404	-
PK-PCA	25	0.6030*	4.37	0.6157	3.07	0.7366	−0.32
PK-Boosting	41	**0.6259***	**6.85**	**0.6409***	**5.85**	**0.7704***	**4.05**

Table 14. Performance comparison on TREC-CS 2016 on dimensionality reduction. The superscript * denotes significant differences compared to LinearCatRev.

	Avg. Dim.	P@5	Δ(%)	nDCG@5	Δ(%)	MRR	Δ(%)
LinearCatRev	302	0.4897	-	0.3213	-	0.6284	-
PK-PCA	16	0.5106	4.21	0.3406	6.02	0.6424	2.23
PK-Boosting	105	**0.5310**	**8.43**	**0.3526***	**9.74**	**0.6800**	**8.21**

PK-Boosting. This difference for TREC-CS 2016 is even bigger due to the fact that users have smaller number of previously visited locations. It is interesting to note the difference between the dimensionality reduction that PK-Boosting exhibits on TREC-CS 2015 and TREC-CS 2016 (i.e., 41 vs. 105). This is due the difference between the average number of user tags in the two datasets. As we can see in Table 4, the average user tag per user for TREC-CS 2015 is 1.46 as compared with 3.61 for TREC-CS 2016. Since PK-Boosting personalizes dimensionality reduction problem according to user tags, the average number of user tags can potentially have an impact on the average number of reduced dimensions. The results of Tables 13 and 14 show that PK-Boosting outperforms PCA in terms of recommendation effectiveness even though PCA is able to reduce more dimensions. This suggests that incorporating personal information for dimensionality reduction is effective and hence PK-Boosting performs better.

6.3 User Tag Prediction

In this experiment we evaluate the effectiveness of different user tag prediction methods. The aim of this experiment is to show how effective user tag prediction is in terms of user tag prediction accuracy. In previous experiments we showed how user tag prediction can improve the overall recommendation effectiveness; however, it is crucial to know how effective is the prediction model so that we can further analyze and improve the prediction accuracy in order to achieve better recommendation. Table 15 reports the performance of different user tag prediction models. Note that CRF and SVM-based taggers are trained using the same feature set for fair comparison. As we can see in this table the SVM based model is able to beat all other models. In fact, the SVM based model benefits highly from the features that are extracted using the proposed mapping and therefore it can beat ML.

7 RELATED WORK

In this section, we review some of the existing related works on POI recommendation, and context-aware POI recommendation.

POI Recommendation. Recommender systems play an important role in satisfying users' expectations for many online services such as e-commerce, LBSN and social network websites. CF-based approaches are based on the core idea that users with similar behavioral history tend to act similarly

Table 15. Performance comparison of user tag prediction models.

	Precision	Recall	F-Measure
ML	0.3982	0.2421	0.3011
SVM-Based	**0.7923**	**0.8110**	**0.8016**
CRF	0.7646	0.7573	0.7609

in the future [23]. A large body of research has been done following this idea [21, 24, 42, 46]. CF-based approaches often suffer from data sparsity since there are a lot of available locations, and a single user can visit only a few of them. As a consequence, the user-item matrix of CF becomes very sparse, leading to poor performance of recommender systems in cases that there is no significant association between users and items. Many studies have tried to address the data sparsity problem of CF by incorporating additional information into the model [41, 44]. More specifically, Ye et al. [41] argued that users' check-in behavior is affected by the spatial influence of locations and proposed a unified location recommender system incorporating spatial and social influence to address the data sparsity problem. Yuan et al. [44], on the other hand, proposed a time-aware collaborative filtering approach. More specifically, they recommended locations to users at a particular time of the day by mining historical check-ins of users in LBSNs. Yin et al. [42] proposed a model which captures user interests as well as local preferences to recommend locations or events to users when they are visiting a new city. Ference et al. [21] took into consideration user preference, geographical proximity, and social influences for POI recommendation. Griesner et al. [24] also proposed an approach integrating temporal and geographic influences into matrix factorization.

Other works follow a review-based strategy, constructing rich user profiles based on their reviews [39, 46]. Reviews reveal the underlying reasons of users' ratings related to a particular location. In fact, as argued by Chen et al. [15], online reviews significantly aid a system to deal with the data sparsity problem. Zhang et al. [46] fused virtual ratings derived from online reviews into CF. Yang et al. [39] created rich user profiles aggregating online reviews from other users and measured the similarity between a new location and a user profile. We exploit online reviews similarly, however, we employ more sophisticated machine learning models to learn users' preferences and opinions. Aliannejadi and Crestani [5] studied various strategies for reducing the number of reviews in users' profiles while maintaining the model's efficiency.

Context-Aware POI Recommendation. Another line of research tries to leverage context to enhance the performance of a recommender system. Context-aware recommendation has been categorized into three types [2]: (1) *pre-filtering*: data selection is done based on context; (2) *post-filtering*: recommendation is done using a traditional approach and context is used to filter them; (3) *contextual modeling*: contextual information is incorporated into the model. Our work aims at modeling the contextual information by re-ranking the recommendations. Adomavicius et al. [1] proposed a multidimensional context pre-filtering model based on the online analytical processing for decision support. Park et al. [36] computed a weighted sum of the conditional probabilities of restaurants' attribute values. They automatically detected users' physical contexts such as the time of the day, the position, and the weather and used a Bayesian network for expressing their probabilistic influences. Levi et al. [31] developed a weighted context-aware recommendation algorithm to address the cold start problem for hotel recommendation. More specifically, they defined context groups based on hotel reviews and followed a user's preferences in trip intent and hotel aspects as well as the user's similarity with other users (e.g., nationality). Other works focused on time as context [19, 20, 22, 44]. Gao et al. [22] developed a time-aware recommendation

model. Fang et al. [20] proposed a model which takes into account both spatial and temporal context to address the data sparsity problem. Deveaud et al. [19] modeled locations popularity in the immediate future utilizing time series. They leveraged the model to make time-aware POI recommendation. Aliannejadi et al. [9] recommended POIs while considering, as users' contexts, the traveling group as well as the season in which the trip occurred. Braunhofer et al. [13] used various complex contextual factors such as budget, companion, and crowdedness to overcome the cold start problem. They developed an active learning strategy and a context-aware recommendation algorithm using an extended matrix factorization model.

TREC Contextual Suggestion. The TREC-CS track [25] aimed to encourage research on context-aware POI recommendation. In fact, the task was to produce a ranked list of locations for each user in a new city, given the user's context and history of preferences in 1-2 other cities. The contextual dimensions were the trip duration, the season, the trip type, and the type of group with whom the user was traveling. These contextual dimensions were introduced in TREC-CS 2015. Since then, among the top runs, few approaches tried to leverage such information. Arampatzis and Kalamatianos [11] studied the performance of various content-based, collaborative, and hybrid fusion methods on TREC-CS and they found that content-based methods performed best among these methods. Yang and Fang [38] introduced some handcrafted rules for filtering locations based on their appropriateness to a user's current context. According to them, applying such filters degrades the performance of the system. Hence, we conclude that contextual appropriateness is not a simple problem of applying some deterministic rules to filter locations. Manotumruksa et al. [35] introduced a set of temporal, term-based, and categorical features to train a set of classifiers to predict contextually appropriate locations. In our opinion, such features are not general enough to be applied to similar problems. Moreover, similar to our work, they collected the contextual relevance dataset using crowdsourcing, however, since they asked the workers to assess the appropriateness of a particular location to a given user's context, this could result in biased assessments. In contrast, we attempt to build a contextual relevance dataset that is not biased and can be used for similar problems.

8 CONCLUSIONS

In this paper, we presented a probabilistic model to find the mapping between user tags and location taste keywords. This mapping enabled us to exploit various directions to address the data sparsity problem for POI recommendation. In particular, we followed two directions: 1) a PK-Boosting model to reduce the dimensionality of location taste keywords and 2) three models to predict user tags for a new location, as alternatives to PK-Boosting. Moreover, we described how to incorporate the new information into POI recommendation, calculating different scores from information from multiple LBSNs. In addition, we also created a dataset to measure the contextual appropriateness of locations and explained how we used the dataset to improve our model. Following learning to rank techniques, the final POI recommendation ranking is obtained based on the computed scores. The experimental results on two TREC collections demonstrate that our method outperforms state-of-the-art strategies. This confirms that the proposed approach, PK-Boosting, addresses the data sparsity problem capturing user preferences accurately.

As future work, we plan to extend our contextual model to capture the time dimension and perform time-aware POI recommendation [33].

REFERENCES

[1] Gediminas Adomavicius, Ramesh Sankaranarayanan, Shahana Sen, and Alexander Tuzhilin. 2005. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Trans. Inf. Syst.* 23, 1 (2005), 103–145.

[2] Gediminas Adomavicius and Alexander Tuzhilin. 2011. Context-Aware Recommender Systems. In *Recommender Systems Handbook*. Springer, 217–253.

[3] Mohammad Aliannejadi, Seyed Ali Bahrainian, Anastasia Giachanou, and Fabio Crestani. 2015. University of Lugano at TREC 2015: Contextual Suggestion and Temporal Summarization Tracks. In *Proceedings of the 24th Text REtrieval Conference, TREC '15, Gaithersburg, MD, USA*. NIST.

[4] Mohammad Aliannejadi and Fabio Crestani. 2017. Venue Appropriateness Prediction for Personalized Context-Aware Venue Suggestion. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '17, Shinjuku, Tokyo, Japan*. ACM, 1177–1180.

[5] Mohammad Aliannejadi and Fabio Crestani. 2018. Venue Suggestion Using Social-Centric Scores. *CoRR* abs/1803.08354 (2018).

[6] Mohammad Aliannejadi, Ida Mele, and Fabio Crestani. 2016. User Model Enrichment for Venue Recommendation. In *Proceedings of the 12th Asia Information Retrieval Societies Conference, AIRS '16, Beijing, China*. Springer, 212–223.

[7] Mohammad Aliannejadi, Ida Mele, and Fabio Crestani. 2016. Venue Appropriateness Prediction for Contextual Suggestion. In *Proceedings of the 25th Text REtrieval Conference, TREC '16, Gaithersburg, MD, USA*. NIST.

[8] Mohammad Aliannejadi, Ida Mele, and Fabio Crestani. 2017. A Cross-Platform Collection for Contextual Suggestion. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '17, Shinjuku, Tokyo, Japan*. ACM, 1269–1272.

[9] Mohammad Aliannejadi, Ida Mele, and Fabio Crestani. 2017. Personalized ranking for context-aware venue suggestion. In *Proceedings of the 32nd Symposium on Applied Computing, SAC '17, Marrakech, Morocco*. ACM, 960–962.

[10] Mohammad Aliannejadi, Dimitrios Rafailidis, and Fabio Crestani. 2017. Personalized Keyword Boosting for Venue Suggestion Based on Multiple LBSNs. In *Proceedings of the 39th European Conference on IR Research, ECIR '17, Aberdeen, UK*. Springer, 291–303.

[11] Avi Arampatzis and Georgios Kalamatianos. 2017. Suggesting Points-of-Interest via Content-Based, Collaborative, and Hybrid Fusion Methods in Mobile Devices. *ACM Trans. Inf. Syst.* 36, 3 (2017), 23:1–23:28.

[12] Jie Bao, Yu Zheng, David Wilkie, and Mohamed F. Mokbel. 2015. Recommendations in location-based social networks: a survey. *GeoInformatica* 19, 3 (2015), 525–565.

[13] Matthias Braunhofer, Mehdi Elahi, and Francesco Ricci. 2014. Techniques for cold-starting context-aware mobile recommender systems for tourism. *Intelligenza Artificiale* 8, 2 (2014), 129–143.

[14] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th International Conference, ICML '07, Corvallis, OR, USA*. ACM, 129–136.

[15] Li Chen, Guanliang Chen, and Feng Wang. 2015. Recommender systems based on user reviews: the state of the art. *User Model. User-Adapt. Interact.* 25, 2 (2015), 99–154.

[16] Chen Cheng, Haiqin Yang, Irwin King, and Michael R. Lyu. 2016. A Unified Point-of-Interest Recommendation Framework in Location-Based Social Networks. *ACM Trans. Intell. Syst. Technol.* 8, 1 (2016), 10:1–10:21.

[17] Corinna Cortes and Vladimir Vapnik. 1995. Support-Vector Networks. *Machine Learning* 20, 3 (1995), 273–297.

[18] Adriel Dean-Hall, Charles L. A. Clarke, Jaap Kamps, Julia Kiseleva, and Ellen M. Voorhees. 2015. Overview of the TREC 2015 Contextual Suggestion Track. In *Proceedings of the 24th Text REtrieval Conference, TREC '15, Gaithersburg, MD, USA*. NIST.

[19] Romain Deveaud, M-Dyaa Albakour, Craig Macdonald, and Iadh Ounis. 2015. Experiments with a Venue-Centric Model for Personalisedand Time-Aware Venue Suggestion. In *Proceedings of the 24th ACM International Conference on Information and Knowledge Management, CIKM '15, Melbourne, VIC, Australia*. ACM, 53–62.

[20] Quan Fang, Changsheng Xu, M. Shamim Hossain, and Ghulam Muhammad. 2016. STCAPLRS: A Spatial-Temporal Context-Aware Personalized Location Recommendation System. *ACM Trans. Intell. Syst. Technol.* 7, 4 (2016), 59:1–59:30.

[21] Gregory Ference, Mao Ye, and Wang-Chien Lee. 2013. Location recommendation for out-of-town users in location-based social networks. In *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management, CIKM'13, San Francisco, CA, USA*. ACM, 721–726.

[22] Huiji Gao, Jiliang Tang, Xia Hu, and Huan Liu. 2013. Exploring temporal effects for location recommendation on location-based social networks. In *Proceedings of the 7th ACM Conference on Recommender Systems, RecSys '13, Hong Kong, China*. ACM, 93–100.

[23] David Goldberg, David A. Nichols, Brian M. Oki, and Douglas B. Terry. 1992. Using Collaborative Filtering to Weave an Information Tapestry. *Commun. ACM* 35, 12 (1992), 61–70.

[24] Jean-Benoît Griesner, Talel Abdessalem, and Hubert Naacke. 2015. POI Recommendation: Towards Fused Matrix Factorization with Geographical and Temporal Influences. In *Proceedings of the 9th ACM Conference on Recommender*

Systems, RecSys '15, Vienna, Austria. ACM, 301–304.

[25] Seyyed Hadi Hashemi, Charles L. A. Clarke, Jaap Kamps, Julia Kiseleva, and Ellen M. Voorhees. 2016. Overview of the TREC 2016 Contextual Suggestion Track. In *Proceedings of the 25th Text REtrieval Conference, TREC '16, Gaithersburg, MD, USA.* NIST.

[26] Tieke He, Hongzhi Yin, Zhenyu Chen, Xiaofang Zhou, Shazia Sadiq, and Bin Luo. 2016. A Spatial-Temporal Topic Model for the Semantic Annotation of POIs in LBSNs. *ACM Trans. Intell. Syst. Technol.* 8, 1 (2016), 12:1–12:24.

[27] Alexandros Karatzoglou, Xavier Amatriain, Linas Baltrunas, and Nuria Oliver. 2010. Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In *Proceedings of the 4th ACM Conference on Recommender Systems, RecSys '10, Barcelona, Spain.* ACM, 79–86.

[28] Taku Kudo and Yuji Matsumoto. 2003. Fast Methods for Kernel-Based Text Analysis. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics, ACL '03, Sapporo, Japan.* ACL, 24–31.

[29] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of the 18th International Conference on Machine Learning, ICML '01, Williamstown, MA, USA.* Morgan Kaufmann, 282–289.

[30] Justin J. Levandoski, Mohamed Sarwat, Ahmed Eldawy, and Mohamed F. Mokbel. 2012. LARS: A Location-Aware Recommender System. In *Proceedings of the 28th IEEE International Conference on Data Engineering, ICDE '12, Washington, DC, USA.* IEEE Computer Society, 450–461.

[31] Asher Levi, Osnat Mokryn, Christophe Diot, and Nina Taft. 2012. Finding a needle in a haystack of reviews: cold start context-based hotel recommender system demo. In *Proceedings of the 6th ACM Conference on Recommender Systems, RecSys '12, Dublin, Ireland.* ACM, 305–306.

[32] Tie-Yan Liu. 2009. Learning to Rank for Information Retrieval. *Foundations and Trends in Information Retrieval* 3, 3 (2009), 225–331.

[33] Yanchi Liu, Chuanren Liu, Bin Liu, Meng Qu, and Hui Xiong. 2016. Unified Point-of-Interest Recommendation with Temporal Interval Assessment. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16, San Francisco, CA, USA.* ACM, 1015–1024.

[34] Yiding Liu, Tuan-Anh Pham, Gao Cong, and Quan Yuan. 2017. An Experimental Evaluation of Point-of-interest Recommendation in Location-based Social Networks. *PVLDB* 10, 10 (2017), 1010–1021.

[35] Jarana Manotumruksa, Craig MacDonald, and Iadh Ounis. 2016. Predicting Contextually Appropriate Venues in Location-Based Social Networks. In *Proceedings of the 7th International Conference of the CLEF Association, CLEF '16, Évora, Portugal.* Springer, 96–109.

[36] Moon-Hee Park, Jin-Hyuk Hong, and Sung-Bae Cho. 2007. Location-Based Recommendation System Using Bayesian User's Preference Model in Mobile Devices. In *Proceedings of the 4th International Conference Ubiquitous Intelligence and Computing, UIC '07, Hong Kong, China.* Springer, 1130–1139.

[37] Francesco Ricci, Lior Rokach, and Bracha Shapira. 2015. Recommender Systems: Introduction and Challenges. In *Recommender Systems Handbook.* Springer, 1–34.

[38] Peilin Yang and Hui Fang. 2015. University of Delaware at TREC 2015: Combining Opinion Profile Modeling with Complex Context Filtering for Contextual Suggestion. In *Proceedings of the 24th Text REtrieval Conference, TREC '15, Gaithersburg, MD, USA.* NIST.

[39] Peilin Yang, Hongning Wang, Hui Fang, and Deng Cai. 2015. Opinions matter: a general approach to user profile modeling for contextual suggestion. *Inf. Retr. Journal* 18, 6 (2015), 586–610.

[40] Mao Ye, Dong Shou, Wang-Chien Lee, Peifeng Yin, and Krzysztof Janowicz. 2011. On the semantic annotation of places in location-based social networks. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '11, San Diego, CA, USA.* ACM, 520–528.

[41] Mao Ye, Peifeng Yin, Wang-Chien Lee, and Dik Lun Lee. 2011. Exploiting geographical influence for collaborative point-of-interest recommendation. In *Proceeding of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR'11, Beijing, China.* ACM, 325–334.

[42] Hongzhi Yin, Bin Cui, Yizhou Sun, Zhiting Hu, and Ling Chen. 2014. LCARS: A Spatial Item Recommender System. *ACM Trans. Inf. Syst.* 32, 3 (2014), 11:1–11:37.

[43] Hongzhi Yin, Bin Cui, Xiaofang Zhou, Weiqing Wang, Zi Huang, and Shazia W. Sadiq. 2016. Joint Modeling of User Check-in Behaviors for Real-time Point-of-Interest Recommendation. *ACM Trans. Inf. Syst.* 35, 2 (2016), 11:1–11:44.

[44] 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, SIGIR '13, Dublin, Ireland.* ACM, 363–372.

[45] 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, SIGIR '15, Santiago, Chile.* ACM, 443–452.

[46] Weishi Zhang, Guiguang Ding, Li Chen, Chunping Li, and Chengbo Zhang. 2013. Generating virtual ratings from chinese reviews to augment online recommendations. *ACM Trans. Intell. Syst. Technol.* 4, 1 (2013), 9:1–9:17.

[47] Vincent Wenchen Zheng, Bin Cao, Yu Zheng, Xing Xie, and Qiang Yang. 2010. Collaborative Filtering Meets Mobile Recommendation: A User-Centered Approach. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence, AAAI '10, Atlanta, GA, USA.* AAAI Press, 236–241.

[48] Vincent Wenchen Zheng, Yu Zheng, Xing Xie, and Qiang Yang. 2010. Collaborative location and activity recommendations with GPS history data. In *Proceedings of the 19th International Conference on World Wide Web, WWW '10, Raleigh, NC, USA.* ACM, 1029–1038.