3M2-IOS-3b-6

Contextual Restaurant Recommendation Utilizing Implicit Feedback

Wei-Ti Kuo *1 Yen-Ling Kuo *1 Jane Yung-jen Hsu *1 Richard Tzong-Han Tsai *2

*1 National Taiwan University, Taipei, Taiwan *2 Yuan Ze University, Taoyuan, Taiwan

Selecting a good restaurant for an event is a great problem for most people. In addition to intrinsic factors of restaurants (e.g. food style, price, and taste), a good recommendation system should also consider users' context information such as purpose of gathering and the type of companions. Although there are many context-aware restaurant recommenders, most of them only focus on location information. This research aims to incorporate more useful contexts into recommendation process.

Unlike extensive work on explicit user ratings of restaurants, this paper utilizes users' restaurant booking history to recommend restaurants. Each booking record contains the dining contexts, e.g. dining date and number of companion, a user leaves when he reserve the restaurant. In this paper, we proposed to use PITF-BPR algorithm to model the context contained in restuarant booking history. Experiments were conducted using the three-year booking history from EZTable, the largest online restaurant booking service in Taiwan, to see the improvements of recommender after we modeled the event contexts.

1. Introduction

As the Internet and social media sites emerges, many systems, e.g. Yelp *1 and OpenTable *2, are launched to help users find restaurants based on others reviews or book restaurants. As a result of overwhelming restaurant information, a good restaurant recommendation system is becoming more and more important.

In addition to intrinsic factors of restaurants (e.g. food style, price, and taste), people also take other factors such as other people's opinions and context information into consideration. The decision of choosing a restaurant is more inconsistent and personalized than before. While other factors have been well studied [7, 12], context information has received great attentions recently [14].

Because of the popularity of mobile recommendation services, most current context-aware restaurant recommendation studies focused on exploiting location information [5, 8, 10, 17] to recommend services to users. However, in real situation, a user chooses restaurants based on several other context types, such as purposes and companions.

The goal of this paper is to incorporate more contextual information into restaurant recommendation. The main challenges in context-aware restaurant recommendation is asking users to provide explicit restaurant ratings for different contexts. Fortunately, these context information are easy captured by users' restaurant booking history. In each booking record, users are required to provide context information, such as purposes, dates, number of companions, etc, to reserve their tables. The identified context are served as a personalized tag to the booked restaurant.

Unlike the most recent work that considers users' explicit rating of restaurants, users' booking history contains only positive classes, i.e. how many times a user book a restaurant. In order to utilize these implicit feedback in real world, we (1) turn users' booking history to a cube that is consisted of user, restaurant, and context dimensions and (2) propose to use PITF-BPR [20] algorithm to optimize the restaurant ranking to each user under different contexts.

The rest of this paper is organized as follows. We start by reviewing previous works on context-aware recommender system, ranking from implicit feedback, and restaurant recommendation. Following the problem definition of context-aware restaurant recommender, the proposed modification of PITF-BPR algorithm are introduced. We then present the experimental set-up to recommend restaurants using three-year booking data from EZTable *3, the largest online restaurant booking service in Taiwan. To evaluate the results, we empirically compare our proposed methods with several baseline recommendation algorithms. This paper concludes with a discussion on the effect of different context information and future work.

2. Background and Related Work

In this section, we will briefly introduce the three approaches of context-aware recommendation at first. Next, we will present the characteristics and corresponding algorithms of implicit feedback. Finally, the restaurant recommender research in the past will be illustrated.

2.1 Context-Aware Recommender Systems

Context-aware recommendation can be classified into three approaches: pre-filtering, post-filtering, and contextual modeling [1]. The pre-filtering approach drives data selection for the specific data, which will easily generate the sparsity problem. Hence, researchers often relax the filter criteria when using the pre-filtering approach. For example, in the taxi demand hotspots prediction research [2], the context constraints of the filter will be relaxed to a super-concept according to the context ontology when the amount of the dataset is not large enough. For instance, 7:30 AM Monday can be relaxed to Monday morning rush hour, which is from 7:00 AM to 9:00 AM.

Contact: Wei-Ti Kuo, National Taiwan University, Taipei, Taiwan, 886-2-33664888, 886-2-23628167, weiti.kuo@gmail.com

∗1 http://www.yelp.com

∗2 http://www.opentable.com

∗3 http://www.eztable.com

The post-filtering approach generates initial recommendations first by traditional methods, then readjusts the initial list for each user using the contextual information. Panniello et al. compares the pre-filtering method versus two different post-filtering methods [16], and he doesn't find clear winners. Thus, this indicates that the best approach to use would depend on each given application. Compare to the previous two methods, the contextual modeling approach uses contextual information directly in the whole modeling process [13]. For example, Yu et al. introduce context as additional modeling dimensions and uses hybrid recommendation techniques [23].

2.2 Ranking from Implicit Feedback

Most of the time, when a model is created based on user's profile, a distinction can be found between the two data collection forms, explicit and implicit. When asking a user to rate an item on a sliding scale, or rank a collection of items from the most favorite to the least favorite, etc. The strategy that requires gathering information from users directly is called explicit feedback. However, if you don't pay money to the workers, it becomes difficult to collect explicit data from users. Thus, some researchers would try using an alternative strategy that would only apply data from implicit feedback [6]. For example, inferring users' preferences from product viewing times according to the psychological evidence [18], observing the items that a user would view [4], etc.

Although implicit measures are generally assumed to be less accurate than explicit measures, but as large quantities of implicit data can be gathered at no extra cost to users, they are attractive alternatives. The prime characteristic of implicit feedback is no negative feedback [4]. For example, a user that did not book a restaurant might have done so because he disliked the restaurant or just because he didn't know about the restaurant or was not available to book it because the booking quota of the restaurant was fulfilled at that time. Hence, it's hard to reliably infer which restaurants a user didn't like. Furthermore, implicit feedback is inherently noisy. For example, we may view the historical dining records for an individual, but this doesn't necessary indicate a positive impression of the restaurant. The restaurant may have been booked for others rather than the user-self, or perhaps the user was disappointed with the restaurant. Therefore, we can only guess the user's preference while we track their dining records passively.

Algorithms that utilize explicit feedback directly are not suitable for implicit feedback due to the characteristics mentioned above. Pan proposes three strategies to treat the missing value with implicit feedback [15]:

- AMAU(treat all missing as unknown): When adopting this strategy, we ignore all the missing examples and utilize the positive examples only. There is no bias because we don't use any missing value. However, we can't use any modeling approach because there are only one-class information.

- AMAN(treat all missing as negative): When adopting this strategy, we can use most of the modeling approaches easily. However, it biases the recommendation results because some of the missing data might be positive.

- Improved method to treat the missing value: To discover techniques that can outperform the AMAU and AMAN strategies, researchers try to find some methods in between the two extreme strategies. For example, Pan proposes two methods for the one-class collaborative filtering problem. The first approach is based on weighted low rank approximation. The second is based on negative example sampling. They both utilize the information contained in unknown data and correct the bias of treating them as negative examples. In short, they are regularized methods of point-wise learning.

Different from the regularized methods of point-wise learning, Rendle assumes that a user would prefer a purchased item over all other non-purchased items, and presents Bayesian personalized ranking(BPR). In this research, we will explore the concept of pair-wise learning.

2.3 Restaurant Recommenders

Currently, most context-aware restaurant recommendation studies focus on exploiting location information due to the popularity of mobile recommendation services, which recommend users services based on their current or given locations. Kitamura proposes a competitive recommendation system which consists of multiple animated agents that would recommend their items competitively [7], and each agent would recommend restaurants based on its view point for the user. It's effective when the user does not have clear requirements or preferences, but it's time-consuming and the user usually becomes tired during a long transaction. Tung illustrates a scenario of recommending restaurants to tourists by having them interact with a personalized agent on a mobile device [22], which merely considers the users' preferences, time and spatial contexts. Kodama extends the notion of spatial skyline queries to incorporate not only distance information, but also categorical preference information to select limited number of restaurants for location-based services [8].

In addition to the studies mentioned above, there are much restaurant research also focus on location [17] [10], and very few restaurant recommendation research would use event contexts. Oku uses C-SVM and C-SVM-CF in restaurant recommendation system to examine the effectiveness when considering restaurant environment parameters and context parameters [13]. In the research, he focuses on the quality of classifying the restaurants into the binary relation(relevant or irrelevant) using the context information. Our goal differ from him, we would like to improve the quality of the total order rank. Hence, there are still room for improvement under context-aware restaurant recommendation.

According to our survey, most restaurant recommendations rely on explicit feedback in the past. However, users usually are not willing to spend a lot of time writing these

questionnaires. Moreover, most of the research use simulated ways to collect data and evaluate the performance instead since dining records are not easy to obtain. However, the conviction of the simulation is not eloquent compare to the actual restaurant dining records. To not cause burden on users, we would recommend restaurants by utilizing implicit feedback of the actual dining records. Furthermore, a user chooses restaurants based on several context types except for the location information, so we would also use the event contexts to improve the performance of recommendations. In short, we focus on the context-aware restaurant recommendation which utilize implicit feedback in this research. Despite the lack of explicit feedback, we will strive to recommend a suitable restaurant list under the given context information.

3. Context-aware Restaurant Recommender

The architecture of a context-aware restaurant recommender system is shown in Figure 1. Given a set of restaurants I and the context c of user u's request, the system should recommend a list of suitable restaurants for the user u. The order of the output restaurant list should reflect the user's preference in that context. Let $y_{u,c,i}$ be the preference score of user u to restaurant i under the context c after the recommendation process. The preference order of output list $O = \{o_1, o_2, \ldots, o_p\}$, where p is the number of output list, should be $y_{u,c,o_1} \geq y_{u,c,o_2} \geq \ldots \geq y_{u,c,o_p}$.

In our definition, a context c is a vector of context tags, and the value of each context tag t is a binary number, either 1 or 0. Namely, $c = \{t_1, t_2, \ldots, t_k\}$, where k is the number of context tags. For example, the context "season" should contain tags "spring", "summer", "fall", and "winter." In the booking history, every dining record r contains a context c. We use $r_{u,c,i}$ to represent the times user u visits restaurant i under the context c. For example, $r_{u_1,c_1,i_1} = 3$ means that user u_1 has gone to restaurant i_1 under the context c_1 three times. Let R be a set of dining records, where each record $r_{u,c,i} > 0$. Let $V = \{v_{u_1,c_1,i_1}, v_{u_1,c_2,i_1}, \ldots, v_{u_n,c_s,i_m}\}$ be a set of visiting records, where $s = 2^k$ is the number of all possible c. Each visiting record presents whether the user u has gone to the restaurant i under the context value c or not. In other words,

$$v_{u,c,i} = \begin{cases} 1, & \text{if } r_{u,c,i} > 0 \\ 0, & \text{otherwise.} \end{cases}$$

In context-aware restaurant recommendation, our goal is to recommend a list of restaurants in accordance with user u's preference under context c. Therefore, we formulate it as a ranking problem. Given a user-context pair (u, c), we would like to predict a total order $> u, c \subset I \times I$ over restaurants, and each ranking $> u, c$ satisfies:

$$\forall i_1, i_2 \in I : i_1 \neq i_2 \Rightarrow i_1 >_{u,c} i_2 \vee i_2 >_{u,c} i_1 \quad (1)$$

$$\forall i_1, i_2 \in I : i_1 >_{u,c} i_2 \wedge i_2 >_{u,c} i_1 \Rightarrow i_1 = i_2 \quad (2)$$

$$\forall i_1, i_2, i_3 \in I : i_1 >_{u,c} i_2 \wedge i_2 >_{u,c} i_3 \Rightarrow i_1 >_{u,c} i_3 \quad (3)$$

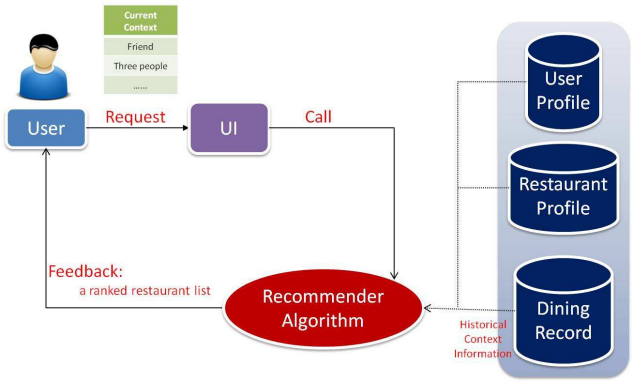

Figure 1: Context-aware recommender system architecture

where equation (1) is totality, equation (2) is antisymmetry, and equation (3) is transitivity. In this paper, our proposed model predicts a scoring function $\hat{y} : U \times C \times I \rightarrow \mathbb{R}$ to derive an order that satisfies antisymmetry and transitivity. In order to ensure totality, we randonly put one restaurants in higher ranking if two restaurants get the same score for the same pair (u, c).

Most of time, only the top restaurants are important to user because user is impatient to browse all ranked restaurant list. Hence, we only show the top N restaurants in the output. The top N restaurants are defined as the following equation:

$$Top(u, c, N) := \arg\max_{i \in I}^{N} \hat{y}_{u,c,i} \quad (4)$$

4. Methodology

Since there are only booking records in the EZTable dataset without negative feedback, we would take the approaches of implicit feedback. Due to the performance of AMAU and AMAN are not good [19], we would use a method in between the two extreme approaches. Assuming that a user would prefer a visited restaurant over all other non-visited restaurants in a specified scenario, and using the pairwise learning to learn a model. According to our survey, we find that the PITF-BPR algorithm is more suitable, and modifying the algorithm slightly to use the context information. In this section, we will introduce the concept of BPR first, then present the PITF-BPR algorithm with context information.

4.1 BPR Optimization Criterion and Learning Algorithm

The Bayesian formulation of finding the best ranking $>_{u,c} \subset I \times I$ for a given pair (u, c) is to maximize the following posterior probability:

$$p(\Theta| >_{u,c}) \propto p(>_{u,c} |\Theta) p(\Theta)$$

where Θ represents the parameters of an arbitrary model. All pairs are assumed independent of each other, it brings about the maximum a posterior (MAP) estimator of the model parameters:

$$\arg\max_{\Theta} \prod_{(u,c) \in U \times C} p(>_{u,c} |\Theta) p(\Theta) \quad (5)$$

The ordering of each pair of restaurants (i_i, i_j) for a specific pair (u, c) is also assumed independent of the ordering of every other pair, and $i_i >_{u,c} i_j$ is a Bernoulli experiment. Therefore, we can infer:

$$\prod_{(u,c)\in U\times C} p(>_{u,c}|\Theta) = \prod_{(u,c,i_i,i_j)\in U\times C\times I^2} p(i_i >_{u,c} i_j|\Theta)^{\delta((u,c,i_i,i_j)\in D_S)} \cdot(1-p(i_i >_{u,c} i_j|\Theta))_{\delta((u,c,i_j,i_i)\in D_S)}$$

where δ is an indicator function:

$$\delta(b) := \begin{cases} 1, & \text{if b is true} \\ 0, & \text{else} \end{cases}$$

D_S is the training data, we will explain in next subsection.

Owing to the totality and antisymmetry of the target function, the above formula can be simplified to:

$$\prod_{(u,c)\in U\times C} p(>_{u,c}|\Theta) = \prod_{(u,c,i_i,i_j)\in D_S} p(i_i >_{u,c} i_j|\Theta) \quad (6)$$

Then we define the probability that a user really prefers restaurant i over restaurant j:

$$p(i_i >_{u,c} i_j|\Theta) := \sigma(\hat{y}_{u,c,i_i,i_j}(\Theta)) \quad (7)$$

where σ is the logistic sigmoid function $\sigma(x) := \frac{1}{1+e^{-x}}$, and $\hat{y}_{u,c,i_i,i_j}(\Theta)$ is a function $\hat{Y} : U \times C \times I^2 \to \mathbb{R}$ which captures the special relationship between user u, context c, restaurant i_i and restaurant i_j by the model parameters Θ. In the following, we will write $\hat{y}_{u,c,i_i,i_j}$ for $\hat{y}_{u,c,i_i,i_j}(\Theta)$ for convenience. By combining the equation (6) and (7), we can get:

$$\prod_{(u,c)\in U\times C} p(>_{u,c}|\Theta) = \prod_{(u,c,i_i,i_j)\in D_S} \sigma(\hat{y}_{u,c,i_i,i_j}) \quad (8)$$

In order to complete BPR, we assume that the model parameters are drawn from a normal distribution $\Theta \sim N(0, \sigma_\Theta^2 I)$ for the prior $p(\Theta)$.

Now, we fill the equation (8) and the prior $p(\Theta)$ into the MAP estimator (5) to derive the optimization criterion for Bayesian Personalized Ranking ($BPR-OPT$):

$$\begin{aligned} BPR-OPT &:= ln\ p(>_{u,c}|\Theta)p(\Theta) \\ &= ln \prod_{(u,c,i_i,i_j)\in D_S} \sigma(\hat{y}_{u,c,i_i,i_j})p(\Theta) \\ &= \sum_{(u,c,i_i,i_j)\in D_S} ln\ \sigma(\hat{y}_{u,c,i_i,i_j}) - \lambda_\Theta \| \Theta \|_F^2 \end{aligned} \quad (9)$$

where λ_Θ are model specific regularization parameters.

For BPR-OPT, Rendle proposed LEARNBPR [19], which is a stochastic gradient descent algorithm shown in Algorithm 1 to optimize the model parameters Θ of $\hat{y}_{u,c,i_i,i_j}$. Because the number of quadruples in D_S is very large, computing the full gradient descent is very slow and not feasible, therefore we draw (u, c, i_i, i_j) by bootstrap sampling. In general, this is a good approach for the class imbalance problem. For example, a restaurant i_i is often visited, then for many combinations (u, c) the restaurant i_i is compared against all non-visited restaurants i_j. Thus the gradient for model parameters learned by i_i would dominate largely if we update the same (u, c, i_i) consecutively. Furthermore, there are many quadruples overlap in three dimension. This means that performing stochastic gradient descent on the drawn case will also help many other related cases. Hence, the bootstrap sampling approach with replacement is suggested here.

Given a case (u, c, i_i, i_j), the gradient of BPR-OPT with respect to a model parameter Θ is:

$$\begin{aligned} &\frac{\partial}{\partial\theta}(ln\ \sigma(\hat{y}_{u,c,i_i,i_j}) - \lambda_\Theta \| \Theta \|_F^2) \\ &\propto (1-\sigma(\hat{y}_{u,c,i_i,i_j})) \cdot \frac{\partial}{\partial\theta}\hat{y}_{u,c,i_i,i_j} - \lambda_\Theta\Theta \end{aligned}$$

That means, we only have to compute the gradient $\frac{\partial}{\partial\theta}\hat{y}_{u,c,i_i,i_j}$ to apply LEARNBPR.

Algorithm 1 LEARNBPR

Require: D_S: the training data; Θ: the model parameters; α: the learning rate; λ: the regularization term;
Ensure: $\hat{\Theta}$: the final model parameters;
1: initial Θ
2: **repeat**
3: draw (u, c, i_i, i_j) uniformly from D_S
4: $\Theta \leftarrow \Theta + \alpha\frac{\partial}{\partial\theta}(ln\sigma(\hat{y}_{u,c,i_i,i_j}) - \lambda_\Theta \| \Theta \|_F^2)$
5: **until** convergence
6: return $\hat{\Theta}$

4.2 Pairwise Interaction Tensor Factorization(PITF) with BPR

According to the recommender research of the past, we know that the collaborative filtering(CF) is very popular [21]. Most of the time, CF can find users' preferences effectively and create a good recommendation list. Besides, as mentioned before, we believe that the context information is more useful for restaurant recommender compare to other domains. Hence, we would like to find a way to combine CF with context information, and then create a better recommendation.

PITF-BPR algorithm was originally used for personalized tag recommendation [20], and it improved the performance obviously. We modify the PITF-BPR by exchanging the item dimension to the context dimension, and exchanging the tag dimension to the restaurant dimension. Then using the modified PITF-BPR algorithm to model the two-way interactions between users, context information, and restaurants.

As mentioned in the previous section, the context value c is composed of a list of context tags, and each context tag t is a binary number, either 1 or 0. Namely, $c = \{t_1, t_2, \ldots, t_k\}$, where k is the number of context tags, and the number of all possible value c will be 2^k. As c is large, the tensor V(visiting record) will be very large and sparse. In order to ease the problem, we transform the context dimension C into the context tag dimension T, namely, transform $V \subseteq U \times C \times I$ into $S \subseteq U \times T \times I$ at first. As S is

a ternary relation over categorical variables, it can be seen as a three-dimensional tensor which showed in Figure 2.

Next, we create the restaurant-to-context tag table for every user. Then based on the personalized restaurant-to-context tag table, we create restaurant-to-restaurant table for every combination of user and context tag. The concept of PITF-BPR is showed in Figure 3. Given user-context tag pair (u, t), we assume that restaurant i_i is preferred over another restaurant i_j iff (u, t, i_i) has been visited and (u, t, i_j) has not been visited. In short, the training data D_S for pairwise constraints is defined as:

$$D_S := \{(u, t, i_i, i_j) | S_{u,t,i_i} = 1 \wedge S_{u,t,i_j} = 0\}$$

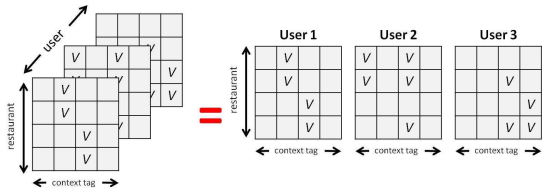

Figure 2: A ternary relation S between users U, context tags T and restaurants I

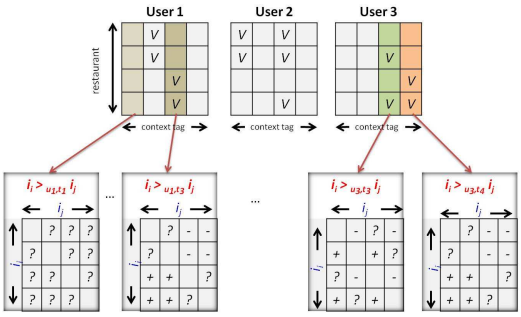

Figure 3: Infer pairwise preferences of restaurants for a given user-context tag pair

The factorization model predict a scoring function $\hat{Y} : U \times T \times I \to \mathbb{R}$ which can be seen as a three-dimensional tensor Y where the value of entry (u, t, i) is the score $\hat{y}_{u,t,i}$. That is to say, we sort the restaurants with respect to $\hat{y}_{u,t,i}$ for ranking within a combination (u, t). For applying BPR optimization, we set:

$$\hat{y}_{u,t,i_i,i_j} := \hat{y}_{u,t,i_i} - \hat{y}_{u,t,i_j}$$

By factorizing each of the three relationship between users, context tags, and restaurants, PITF-BPR models the two-way interactions among them explicitly:

$$\hat{y}_{u,t,i} = \sum_f \hat{u}_{u,f} \cdot \hat{i}^U_{i,f} + \sum_f \hat{t}_{t,f} \cdot \hat{i}^T_{i,f} + \sum_f \hat{u}_{u,f} \cdot \hat{t}_{t,f} \quad (10)$$

Given a combination (u, t), we can find that the user-context tag score $\sum_f \hat{u}_{u,f} \cdot \hat{t}_{t,f}$ is identical for all restaurants. Hence, the user-context tag interaction vanishes for predicting ranking and for BPR optimization. Then the final PITF model:

$$\hat{y}_{u,t,i} = \sum_f \hat{u}_{u,f} \cdot \hat{i}^U_{i,f} + \sum_f \hat{t}_{t,f} \cdot \hat{i}^T_{i,f} \quad (11)$$

with model parameters:

$$\hat{U} \in \mathbb{R}^{|U| \times K}, \hat{T} \in \mathbb{R}^{|T| \times K},$$
$$\hat{I^U} \in \mathbb{R}^{|I| \times K}, \hat{I^T} \in \mathbb{R}^{|I| \times K}$$

where K is the latent feature number.

As mentioned before, we use stochastic gradient descent(SGD) to optimize the model parameters, and the gradients for the PITF model are:

$$\frac{\partial \hat{y}_{u,t,i}}{\partial \hat{u}_{u,f}} = \hat{i}^U_{i,f}, \frac{\partial \hat{y}_{u,t,i}}{\partial \hat{t}_{t,f}} = \hat{i}^T_{i,f},$$
$$\frac{\partial \hat{y}_{u,t,i}}{\partial \hat{i}^U_{i,f}} = \hat{u}_{u,f}, \frac{\partial \hat{y}_{u,t,i}}{\partial \hat{i}^T_{i,f}} = \hat{t}_{t,f}$$

The complete PITF-BPR algorithm is shown in Algorithm 2.

Algorithm 2 Pairwise Interaction Tensor Factorization with BPR

Require: U: the set of user; T: the set of context tag; I: the set of restaurant; $S \in \mathbb{R}^{|\mathbb{U}| \times |\mathbb{T}| \times |\mathbb{I}|}$: the set of context tag visiting record; $\hat{U} \in \mathbb{R}^{|\mathbb{U}| \times \mathbb{K}}$: user latent feature matrix; $\hat{T} \in \mathbb{R}^{|\mathbb{T}| \times \mathbb{K}}$: context tag latent feature matrix; $\hat{I^U} \in \mathbb{R}^{|\mathbb{I}| \times \mathbb{K}}$: latent feature matrix between restaurant and user; $\hat{I^T} \in \mathbb{R}^{|\mathbb{I}| \times \mathbb{K}}$: latent feature matrix between restaurant and context tag; K: the number of latent features; α: the learning rate; λ: the regularization term;

Ensure: the final matrix of $\hat{U}$, $\hat{T}$, $\hat{I^U}$, $\hat{I^T}$;

initial $\hat{U}$, $\hat{T}$, $\hat{I^U}$, $\hat{I^T}$
repeat
  draw (u, t, i_i, i_j) from D_S
  $\hat{y}_{u,t,i} = \hat{u}_u \cdot \hat{i}^U_i + \hat{t}_t \cdot \hat{i}^T_i + \hat{u}_u \cdot \hat{t}_t$
  $\hat{y}_{u,t,j} = \hat{u}_u \cdot \hat{i}^U_j + \hat{t}_t \cdot \hat{i}^T_j + \hat{u}_u \cdot \hat{t}_t$
  $\hat{y}_{u,t,i,j} = \hat{y}_{u,t,i} - \hat{y}_{u,t,j}$
  $\delta \leftarrow (1 - \sigma(\hat{y}_{u,t,i,j}))$
  for $f = 1$ to K **do**
    $\hat{u}_{uf} \leftarrow \hat{u}_{uf} + \alpha[\delta * (\hat{i}^U_{if} - \hat{i}^U_{jf}) - \lambda * \hat{u}_{uf}]$
    $\hat{t}_{tf} \leftarrow \hat{t}_{tf} + \alpha[\delta * (\hat{i}^T_{if} - \hat{i}^T_{jf}) - \lambda * \hat{t}_{tf}]$
    $\hat{i}^U_{if} \leftarrow \hat{i}^U_{if} + \alpha(\delta * \hat{u}_{uf} - \lambda * \hat{i}^U_{if})$
    $\hat{i}^U_{jf} \leftarrow \hat{i}^U_{jf} + \alpha(-\delta * \hat{u}_{uf} - \lambda * \hat{i}^U_{jf})$
    $\hat{i}^T_{if} \leftarrow \hat{i}^T_{if} + \alpha(\delta * \hat{t}_{tf} - \lambda * \hat{i}^T_{if})$
    $\hat{i}^T_{jf} \leftarrow \hat{i}^T_{jf} + \alpha(-\delta * \hat{t}_{tf} - \lambda * \hat{i}^T_{jf})$
  end for
until convergence
return $\hat{U}$, $\hat{T}$, $\hat{I^U}$, $\hat{I^T}$

5. Experimental Design and Result

In this section, we will introduce the EZTable system and its data distribution first. Then we will present the experimental setup which includes the split of training set and

Context Information	*Context Tags*
Purpose	family gathering, friend gathering, date, business meeting, birthday party, others
Season	Spring, Summer, Fall, Winter
Day of Week	Sunday, Weekday { Monday ∼ Thursday }, Friday, Saturday
People	1, 2, 3 ∼ 6, Greater than or equal to 7, Unknown
Dining Time	0:00 ∼ 9:59, 10:00 ∼ 13:59, 14:00 ∼ 16:59, 17:00 ∼ 23:59
Holiday	Father's Day, Mother's Day, Thanksgiving, Chinese New Year's Eve, Saint Valentine's Day, White Day, Chinese Valentine's Day Halloween, New Year's Eve, Christmas, Moon Festival, Chinese New Year

Table 1: Context tags corresponding to the context information

testing set, evaluation metrics, and the compared baseline algorithms. The experimental result with interesting discoveries will be illustrated in the last part of this section.

5.1 EZTable Dataset

In this work, we use the EZTable data between August 27, 2008 and December 6, 2011 about three years, Figure 4 shows the booking distribution. There are 120480 users, 359 restaurants, and 167702 bookings in the dateset. 'Active User' is defined as any user who books more than two different restaurants, and there are only 10316 Active Users in 120480 users. 'Active Booking' is defined as bookings by the Active User, and there are only 35177 Active Bookings in 167702 bookings. Because we use CF as the main recommender algorithm, only the Active Bookings by the Active Users are useful.

Every record r includes Purpose, Season, Day of Week, People, Dining Time, Holiday information, and we transform the information into a list of context tags in Table 1. In order to supply the evidences that the 'Active Bookings' on behalf of all bookings, we calculate the Kullback-Leibler(KL) divergence between them, the result is shown in Table 2, and we can find that all of the KL divergence are close to 0.

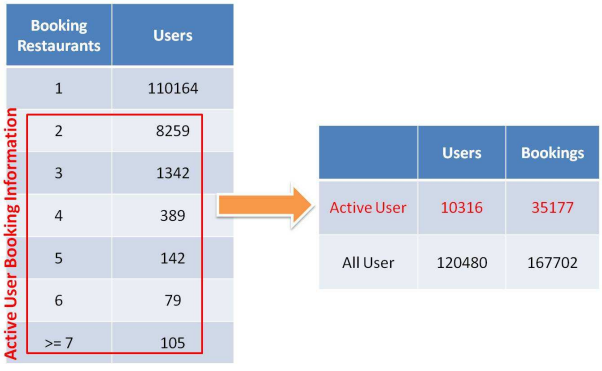

Figure 4: Booking distribution of the EZTABLE dataset

Context	*KL divergence*
Purpose	0.002552
Season	0.001397
Day of Week	0.000470
People	0.007321
Dining Time	0.000274
Holiday	0.001452

Table 2: KL divergence between the distribution of active bookings and all bookings under the given context

5.2 Experimental Setup

In this subsection, we will introduce how the training set and testing set are split first, and then present the evaluation metrics. Lastly, we will show some baseline algorithms and explain why we choose them as the comparison objects.

5.2.1 Training and Testing

As Figure 4 shows, we only use the 35177 Active Bookings which are booked by the 10316 Active Users. Because some users go to a restaurant more than once, the visiting matrix $V : U \times I$ only 24564 entries equal to 1, others are 0, and $|U| = 10316, |I| = 359$.

Due to the sparsity, we use the leave one out evaluation scheme, where we remove for 1031 (a tenth of $|U|$) users randomly one action (one entry of $V_{u,i} = 1$), and these entries be the testing set V_{test}, and the other entries whose $V_{u,i} = 1$ be the training set V_{train}.

5.2.2 Evaluation Metrics

Refer to the evaluation of BPR by Rendle [19], the models are then learned on V_{train} and their predicted personalized ranking is on the test set V_{test} by the average AUC statistic:

$$AUC = \frac{1}{|V_{test}|} \sum_{u} \frac{1}{|E(u)|} \sum_{(i,j) \in E(u)} \delta(\hat{y}_{u,i} > \hat{y}_{u,j}) \quad (12)$$

Where the evaluation pairs per testing user u are $E(u) := \{(i,j)|(u,i) \in V_{test} \wedge (u,j) \notin (V_{test} \cup V_{train})\}$, and δ is an indicator function.

Except for the AUC, we also use *Recall@k* metric because we want to find the suitable restaurant at top k of the ranked list [3].

$$Recall@k = \frac{1}{|V_{test}|} \sum_{u} |R_{u,k} \cap V_{test}| \quad (13)$$

Where $R_{u,k}$ is the set of top k restaurants in the ranked list of the testing user u.

When we use the recommender algorithm which considers the context information, the average AUC statistic and *Recall@k* should be redefined as follows:

$$AUC_c = \frac{1}{|V_{test}|} \sum_{u} \frac{1}{|E(u,c)|} \sum_{(i,j) \in E(u,c)} \delta(\hat{y}_{u,c,i} > \hat{y}_{u,c,j}) \quad (14)$$

Where the evaluation pairs per testing user u under the context value c are $E(u,c) := \{(i,j)|(u,c,i) \in V_{test} \wedge (u,c,j) \notin$

$(V_{test} \cup V_{train})\}$, and δ is an indicator function.

$$Recall@k_c = \frac{1}{|V_{test}|} \sum_u |R_{u,c,k} \cap V_{test}| \quad (15)$$

Where $R_{u,c,k}$ is the set of top k restaurants in the ranked list of the testing user u under the context value c.

5.2.3 Baseline Algorithms

In order to show the effect of PITF-BPR, we choose some algorithms as our baseline. In the following, the chosen algorithms are introduced:

- Popularity: Recommendation by the amount of booking for each restaurant, and the Top-N popular restaurants will always be recommended for each user. This is the most trivial way, and we would like to find out whether or not most consumers would go to these popular restaurants by this simple method.
- Item-to-Item CF: Linden uses this algorithm for producing recommendations on Amazon [11]. Under this method, we create restaurant-to-restaurant cosine similarity table by using the historical information. When the testing user has more than two historical dining records, we can rank by the sum or by the maximum value of the similarity score.
- Matrix Factorization(MF): We use this method by treating all missing entries as 0, and all non-missing entries as 1. Here, we use the AMAN(treat all missing as negative) approach. Without the special setting, we would like to recommend restaurants by model the interaction between user latent features and restaurant latent features. The details of MF can be found in [9].
- BPR with MF: We use the BPR optimization with MF, and update the latent features by pairwise learning. The setting is the same as [19], and we would like to use this method to see the effects of pairwise learning without context information.
- Context-Popularity: We use the overall popularity under the given context tag. As mentioned before, the context value $c = \{t_1, t_2, \ldots, t_k\}$. When a context value c of a request from user u, we rank the restaurants I according to the popularity of context tags whose value equal 1. Using this method, we recommend restaurants by only considering the context information, and don't consider the users' preferences. By using the novel way to use the context information, we can compare whether the context is effective or not.

5.3 Experimental Result

In this subsection, we will show the experimental result without context v.s. with context by AUC and Recall at first. Then, we will show the variance of recommended restaurant styles under different algorithms by a case study. Lastly, we will compare the difference between recommendations and present discussions.

5.3.1 without Contexts v.s. with Contexts

We will perform experiments in two parts. In the first stage, we try several algorithms without context shown in Table 3. The Popularity, Item-to-Item CF (Max or Sum) are the AMAU approaches, the MF (0,1) is the AMAN approach, and the BPR with MF (Bootstrap Sampling or All Pair) are methods in between these two extreme approaches listed above. In Table 3, we find that Item-to-Item CF [11] dominates the other algorithms. Furthermore, the bad performance of BPR with MF is unexpected, we think that the sparsity of dataset cause the training model can't learn well. In addition, we find that the *Recall*@30 of Popularity is close to 0.5, which means that most of the users choose the Top-30 popular restaurants.

In the second stage, we try Context-Popularity and PITF-BPR algorithms with context information, and the results with different contexts are shown in Table 4 and Table 5. The following are interesting discoveries: First, we find that the Context-Popularity shows only a bit of improvement compare to the Popularity without context. Secondly, we find that the *Recall*@30 of Context-Popularity with Purpose exceeds 0.5, which means that more than half of the users choose the Top-30 popular restaurants under the corresponding purpose. Third, we find that the performance of PITF-BPR with Season is the best among all contexts, and the difference between Context-Popularity and PITF-BPR with Season is the greatest. It's not surprising that the context Season is useful, because most consumers would choose different restaurants based on the weather at the time. For example, the frequency of going to a hot pot restaurant during the Winter is much more than during the Summer for most consumers. Lastly, we find that the PITF-BPR algorithm performs well with most of the context information except for People. We believe that using just People as context information is not enough to create a good model. For example, the relationship between two people may be a couple, good friends, brothers, sisters, or others. As Figure 5 shows, the most popular number of People is two. However, we can only find dating as a small portion of Purpose in Figure 6.

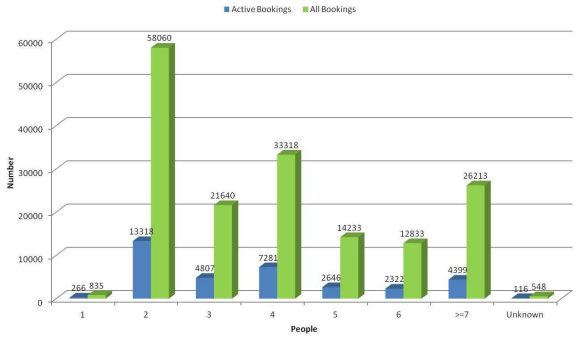

Figure 5: People distribution of the active bookings and all bookings

In order to present the influence of contexts, we show the Popularity, Item-to-Item CF(the best performance

Algorithm without Context Information	*Recall@30*	*AUC*
Popularity	0.471113	0.831427
Item-to-Item CF (Max)	0.605176	0.87601
Item-to-Item CF (Sum)	0.604692	0.875653
MF (0,1)	0.547481	0.830382
BPR with MF (Bootstrap Sampling)	0.458333	0.829978
BPR with MF (All Pair)	0.379845	0.802442

Table 3: Experiment result of *Recall*@30 and *AUC* without context

Context-Popularity	*Recall@30*	*AUC*
Purpose	0.505814	0.839348
Season	0.478682	0.836012
Day of Week	0.501931	0.833219
People	0.486434	0.83425
Dining Time	0.484496	0.833767
Holiday	0.472868	0.831564

Table 4: Experiment result with Context-Popularity

PITF-BPR	*Recall@30*	*AUC*
Purpose	0.673450	0.895341
Season	0.690891	0.895848
Day of Week	0.667636	0.893706
People	0.474806	0.817531
Dining Time	0.683140	0.896730
Holiday	0.677326	0.894694

Table 5: Experiment result with context PITF-BPR

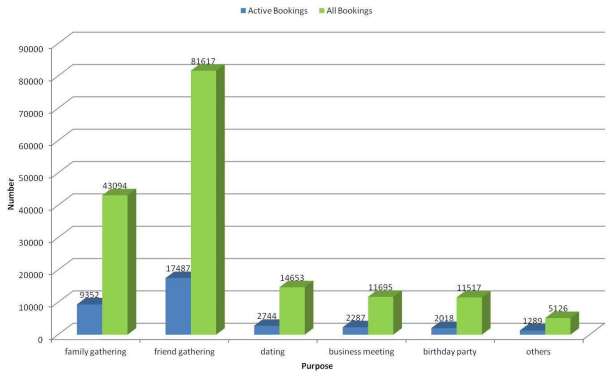

Figure 6: Purpose distribution of the active bookings and all bookings

without context), the Popularity with Purpose(the best performance of Context-Popularity), and the PITF-BPR with Season(the best performance of PITF-BPR with context) in Figure 7. Then, we find that the PITF-BPR with Season dominates other algorithms, and the result of *Recall*@30(0.690891) is higher than Item-to-Item CF. Hence, we can conclude that the context information with appropriate modeling method is useful for restaurant recommendation.

Furthermore, in Table 5, except the context Season, we find that the Purpose, Day of Week, Dining Time, and Holiday are effective context information. Purpose is useful because most consumers would make different decisions when they go out with different companionships. Day of Week and Holiday are useful because most consumers would choose fancy restaurants on Friday, weekends, and special holidays. Dining Time is useful because most consumers would make different decisions between lunch and dinner. For example, most consumers would prefer going to a restaurant with night view for dinner than lunch.

Lastly, we compare the result of BPR with PITF-BPR. Both algorithms use pairwise learning to train a model with implicit feedback. Interestingly, we find that the pairwise learning can't learn well in BPR without context information, but it learns well in PITF-BPR with context information. Hence, we can comment that the modeling method of pairwise learning with context information is more powerful than the traditional two-dimensional CF algorithms with implicit feedback.

5.3.2 Case Study

In order to show the influence of context information, we will illustrate a testing example below. A user would want to find a suitable restaurant on Valentine's Day, and the request is {Purpose = Date, Season = Spring, DayOfWeek = Wednesday, People = 2, DiningTime = 19:00, Holiday = Valentine's Day}. When using the Popularity, the type of restaurants shown in the preceding list are {famous, Thai cuisine, all-you-can-eat}. The Popularity recommends restaurants only by popularity, and doesn't consider other factors. Hence, the famous restaurants will appear in the front of the restaurant list intuitively.

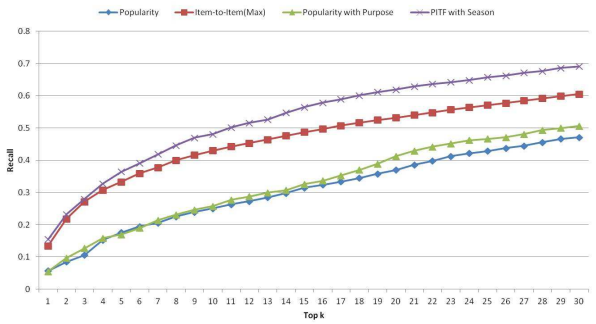

Figure 7: Comparing the experiment result with context and without context

The user has gone to the restaurants which have the properties { Thai cuisine, all-you-can-eat, barbecue } in the past. When using the Item-to-Item CF, the historical records will be considered. Hence, the type of restaurants in the preceding list by the Item-to-Item CF are {Thai cuisine, all-you-can-eat, barbecue, hot pot}. However, the Item-to-Item CF doesn't consider the context information. Since most of the historical records are group of friends gathering together, the scenario of this request(celebrate Valentine's Day with significant other) is different. Hence, the quality of the recommendation is not good using Item-to-Item CF in this case.

When using Context-Popularity with Holiday, the type of restaurant in the preceding list are {famous, haute cuisine, romantic}, which reflects most consumer's choices when celebrating Valentine's Day. When using PITF-BPR with Holiday, the type of restaurants in the preceding list are {famous, romantic, all-you-can-eat} which not only reflects the context information, but also shows the user's preference. From this example, we can find that Holiday is very useful as event context for restaurant recommendation. Although there is no dining records on Valentine's Day of the user, there are many dining records on Valentine's Day by other users. Hence, we can find a suitable restaurant recommendation list which meet the scenario of this dining together when considering the event context Holiday.

5.3.3 Comparison and Discussion

According to the experimental results above, these are the following comments: Item-to-Item CF only considers the user's preference, and doesn't use any context information. Context-Popularity only uses the context information, and doesn't consider the user's preference. However, both the user's preference and the context information are very important factors for restaurant recommendation. Hence, neither Item-to-Item CF nor Context-Popularity can perform well. By using the PITF-BPR, we model the user's preference, and context information simultaneously, which greatly improve the performance in our experiments. As a result, we believe that PITF-BPR is a good contextual modeling approach for restaurant recommendation.

6. Conclusion

This research aims to incorporate context information other than location to improve restaurant recommendation results. We proposed to use model-based algorithm PITF-BPR to explicitly model the relationship between context and users' implicit feedback (i.e. restaurant booking history). Our experimental results on EZTable booking history showed that context-based PITF-BPR outperformed item-to-item CF and context popularity approaches. The result also suggests that "season" is the most useful context among all the contexts. Future work would focus on combining multiple contexts in our model and deploy the system to real online restaurant recommendation service.

Acknowledgments

The authors would like to thank EZTable to provide their booking history for our experiments.

References

[1] Gediminas Adomavicius and Alexander Tuzhilin. Context-aware recommender systems. In Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors, *Recommender Systems Handbook*, pages 217–253. Springer US, 2011.

[2] Han-wen Chang, Yu-chin Tai, and Yung-jen Hsu. Context-aware taxi demand hotspots prediction. *International Journal of Business Intelligence and Data Mining*, 5:3–18, 2010.

[3] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22:5–53, 2004.

[4] Yifan Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *Data Mining, 2008. ICDM '08. Eighth IEEE International Conference on*, pages 263 –272, dec. 2008.

[5] Chi-Chia Huang. Impact analysis of contextual information in a mobile restaurant recommender system. Master's thesis, Department of Computer Science and Information Engineering College of Electrical Engineering and Computer Science National Taiwan University, 2009.

[6] Diane Kelly and Jaime Teevan. Implicit feedback for inferring user preference: a bibliography. *SIGIR Forum*, 37:18–28, September 2003.

[7] Yasuhiko Kitamura, Toshiki Sakamoto, and Shoji Tatsumi. A competitive information recommendation system and its behavior. In *Proceedings of the 6th International Workshop on Cooperative Information Agents VI*, CIA '02, pages 138–151, London, UK, UK, 2002. Springer-Verlag.

[8] Kazuki Kodama, Yuichi Iijima, Xi Guo, and Yoshiharu Ishikawa. Skyline queries based on user locations and preferences for making location-based recommendations. In *Proceedings of the 2009 International Workshop on Location Based Social Networks*, LBSN '09, pages 9–16, New York, NY, USA, 2009. ACM.

[9] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30 –37, August 2009.

[10] Bae-Hee Lee, Heung-Nam Kim, Jin-Guk Jung, and Geun-Sik Jo. Location-based service with context data for a restaurant recommendation. In Stphane Bressan, Josef Kng, and Roland Wagner, editors, *Database and Expert Systems Applications*, volume 4080 of *Lecture Notes in Computer Science*, pages 430–438. Springer Berlin / Heidelberg, 2006.

[11] G. Linden, B. Smith, and J. York. Amazon.com recommendations: item-to-item collaborative filtering. *Internet Computing, IEEE*, 7(1):76 – 80, jan/feb 2003.

[12] Joseph F McCarthy. Pocket restaurantfinder: A situated recommender system for groups. *Workshop on Mobile AdHoc Communication at the 2002 ACM Conference on Human Factors in Computer Systems*, pages 1–10, 2002.

[13] K. Oku, S. Nakajima, J. Miyazaki, and S. Uemura. Context-aware svm for context-dependent information recommendation. In *Mobile Data Management, 2006. MDM 2006. 7th International Conference on*, page 109, may 2006.

[14] Kenta Oku, Shinsuke Nakajima, Jun Miyazaki, Shunsuke Uemura, and Hirokazu Kato. A ranking method based on users' contexts for information recommendation. In *Proceedings of the 2nd international conference on Ubiquitous information management and communication*, ICUIMC '08, pages 289–295, New York, NY, USA, 2008. ACM.

[15] Rong Pan, Yunhong Zhou, Bin Cao, N.N. Liu, R. Lukose, M. Scholz, and Qiang Yang. One-class collaborative filtering. In *Data Mining, 2008. ICDM '08. Eighth IEEE International Conference on*, pages 502 –511, dec. 2008.

[16] Umberto Panniello, Alexander Tuzhilin, Michele Gorgoglione, Cosimo Palmisano, and Anto Pedone. Experimental comparison of pre- vs. post-filtering approaches in context-aware recommender systems. In *Proceedings of the third ACM conference on Recommender systems*, RecSys '09, pages 265–268, New York, NY, USA, 2009. ACM.

[17] Moon-Hee Park, Jin-Hyuk Hong, and Sung-Bae Cho. Location-based recommendation system using bayesian users preference model in mobile devices. In Jadwiga Indulska, Jianhua Ma, Laurence Yang, Theo Ungerer, and Jiannong Cao, editors, *Ubiquitous Intelligence and Computing*, volume 4611 of *Lecture Notes in Computer Science*, pages 1130–1139. Springer Berlin / Heidelberg, 2007.

[18] Jeffrey Parsons, Paul Ralph, and Katherine Gallager. Using viewing time to infer user preference in recommender systems. 2004.

[19] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Schmidt-Thie Lars. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, UAI '09, pages 452–461, Arlington, Virginia, United States, 2009. AUAI Press.

[20] Steffen Rendle and Lars Schmidt-Thieme. Pairwise interaction tensor factorization for personalized tag recommendation. In *Proceedings of the third ACM international conference on Web search and data mining*, WSDM '10, pages 81–90, New York, NY, USA, 2010. ACM.

[21] Xiaoyuan Su and Taghi M. Khoshgoftaar. A survey of collaborative filtering techniques. *Adv. in Artif. Intell.*, 2009:4:2–4:2, January 2009.

[22] Hung-Wen Tung and Von-Wun Soo. A personalized restaurant recommender agent for mobile e-service. In *Proceedings of the 2004 IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE'04)*, EEE '04, pages 259–262, Washington, DC, USA, 2004. IEEE Computer Society.

[23] Zhiwen Yu, Xingshe Zhou, Daqing Zhang, Chung-Yau Chin, Xiaohang Wang, and Ji men. Supporting context-aware media recommendations for smart phones. *Pervasive Computing, IEEE*, 5(3):68 –75, july-sept. 2006.