User Latent Preference Model for Better Downside Management in Recommender Systems

Jian Wang
LinkedIn Corp
Mountain View, CA 94043 USA
jianwang@linkedin.com

David Hardtke
LinkedIn Corp
Mountain View, CA 94043 USA
dhardtke@linkedin.com

ABSTRACT

Downside management is an important topic in the field of recommender systems. User satisfaction increases when good items are recommended, but satisfaction drops significantly when bad recommendations are pushed to them. For example, a parent would be disappointed if violent movies are recommended to their kids and may stop using the recommendation system entirely. A vegetarian would feel steakhouse recommendations useless. A CEO in a mid-sized company would feel offended by receiving intern-level job recommendations. Under circumstances where there is penalty for a bad recommendation, a bad recommendation is worse than no recommendation at all. While most existing work focuses on upside management (recommending the best items to users), this paper emphasizes on achieving better downside management (reducing the recommendation of irrelevant or offensive items to users). The approach we propose is general and can be applied to any scenario or domain where downside management is key to the system.

To tackle the problem, we design a user latent preference model to predict the user preference in a specific dimension, say, the dietary restrictions of the user, the acceptable level of adult content in a movie, or the geographical preference of a job seeker. We propose to use multinomial regression as the core model and extend it with a hierarchical Bayesian framework to address the problem of data sparsity. After the user latent preference is predicted, we leverage it to filter out downside items. We validate the soundness of our approach by evaluating it with an anonymous job application dataset on LinkedIn. The effectiveness of the latent preference model was demonstrated in both offline experiments and online A/B testings. The user latent preference model helps to improve the VPI (views per impression) and API (applications per impression) significantly which in turn achieves a higher user satisfaction.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms

Algorithms, Design, Experimentation

Keywords

Recommender System; Downside Management; User Latent Preference Model

1. INTRODUCTION

Recommender systems are popular research topics in the information retrieval community. A host of academic and industrial incarnations of recommender systems exist in domains such as movies (Netflix), music (Pandora), and e-commerce product recommendations (eBay, Amazon). Most existing work focuses on upside management by finding the best items to recommend to a user. Better upside management can be achieved by leveraging state-of-art models such as collaborative filtering, matrix factorization, deep learning neural networks and so on. While it is essential to discover good items, it is equally important to avoid recommending irrelevant or offensive items to a user. User satisfaction is affected by both upside items and downside items. For example, a CEO in a mid-sized company would be interested in other executive positions while he would feel offended by receiving intern-level job recommendations. An item could be irrelevant or offensive for various reasons in different domains. It might be caused by wrong dining preference, wrong job location, wrong movie category and so on. In some cases, a single bad recommendation could cause a user to abandon the recommendation system completely.

One simple strategy for downside management is not to show an item when the predicted probability from recommender systems is too low: you just need to find an appropriate threshold. Such strategy is based on the assumption that an "ideal" recommender system could assign an appropriate probability for an item. In particular, the probability should indicate the user's willingness of accepting the item in different dimensions. Yet it is not the case in the real world. Most existing hybrid recommender systems (such as logistic regression, gradient boosting tree, etc.) consider all features together to predict a single probability for an item. The item might be recommended to a user according to its high probability if the content match is high even though the other contextual match is low. Users might not complain about a slight content mismatch. Yet they are less tolerant to be pushed with recommendations in the wrong location, wrong category or for other contextual reasons. Thus it would be hard to find the right threshold on a single probability from the existing recommender system if a downside item is about some specific dimension.

Downside management has been studied by some recent work [28, 29, 1, 8, 19, 25]. Researchers studied context-aware recommender systems [1, 8, 19, 25] to find items that match with the user's contexts. Contextual information could be leveraged in the pre-filtering stage, post-filtering stage, or in the contextual modeling. The concept of "context" is different from the "latent preference" in our model. The "context" is an explicit condition (for example, the current location of the user) that could be observed for each user while the "latent preference" is an implicit yet explainable factor that the model infers for each user (for example, the location of an item that the user would prefer). We study how the user's implicit preferences or interests would affect recommendations. This latent preference could be dietary restrictions of the user, the acceptable level of adult content in a movie, the geographical preference of a job seeker and so on.

In this work, we propose a general framework that predicts a user's latent preference and leverage it to filter out downside items. The core of the framework is multinomial regression that models a user's latent preference in a specific dimension. The user's preference could either be binary (for example, work at a nearby job, or take a job that requires relocation) or multi-class (for example, work as the intern level, senior level, staff level, or executive level). We assume that users in different groups tend to have different priors of preference choice. For example, job seekers around 22 years of old are more likely to relocate compared to older job seekers. To accommodate such differences in user segments, we propose to divide users in different segments and learn a multinomial regression for each segment. The segment could be based on the user's age, gender, seniority, region and so on. We further extend it with a hierarchical Bayesian framework to address the problem of data sparsity for those small segments. With this framework, different segments learn from each other by sharing the prior of the regression model. Detailed mathematical parameter inference steps are derived with the variational Bayesian algorithm. Items that would hurt the user satisfaction are filtered out in the recommendation stage.

While the framework we propose is general to any downside dimension in any domain, we apply it to the location dimension in the job domain in this paper. In other words, the model aims to avoid recommending jobs in locations where the job seeker would not want to work. In the experiments, we first evaluate the performance of the user latent preference model itself with traditional classification IR metrics. We then demonstrate the effectiveness of leveraging the user latent preference in recommender systems. The model was evaluated with both an offline anonymous job application dataset and real-world job seekers on LinkedIn. Experiments show that the user latent preference model helps to improve the VPI (views per impression) and API (applications per impression) metrics significantly. At the same time, the absolute number of applications and views remain stable in spite of fewer impressions. Such performance follows our intuition that the user latent preference model achieves better downside management which in turn results in higher user satisfaction. The major contributions of this paper include the following:

- Propose and study the problem of downside management in the field of recommender systems.
- Design the **User Latent Preference Model** to tackle the problem and leverage it in recommender systems. Propose to use the multinomial regression as the core model and extend it with a hierarchical Bayesian framework. Derive detailed parameter inference steps based on the variational Bayesian algorithm.
- Evaluate the model with both offline framework and online A/B testing on LinkedIn. Demonstrate its effect in achieving better downside management of recommender systems.

2. RELATED WORK

A major task of the recommender system is to present recommendations to the user. The task is usually conducted by first predicting a user's ratings for each item and then ranking all items in the descending order. There are two major recommendation approaches: content-based filtering and collaborative filtering. Content-based filtering [17, 22] assumes that descriptive features of an item indicate a user's preferences. Thus, a recommender system makes a decision for a user based on the descriptive features of other items the user likes or dislikes. Usually, the system recommends items that are similar to what the user liked before. Collaborative filtering [12, 24, 13, 16, 26, 21, 10] on the other hand assumes that users with similar tastes on some items may also have similar preferences on other items. Thus, the main idea is to use the behavior history from other like-minded users to provide the current user with good recommendations. Research on collaborative filtering algorithms reached a peak due to the 1 million dollar Netflix movie recommendation competition [4]. Factorization-based collaborative filtering approaches [7, 14, 27, 23], such as regularized Singular Value Decomposition, performed well on this competition, possibly better than Netflix's own well-tuned Pearson correlation coefficient algorithm. A common characteristic of these models is the introduction of user latent factors or/and item latent factors to solve the data sparsity issue.

Context-aware recommender systems [1, 8, 19, 25, 2, 11, 15] has become a hot topic in the research field recently. Most work focuses on using user contextual information or latent tastes to improve the upside management. For example, Weston et.al [30] proposed an advanced matrix factorization model to incorporate user's latent tastes. Hariri et. al [11] proposed a unified probabilistic model to integrate the user profile, item representations, and contextual information. They demonstrated the effectiveness of their model in the domain of article and music recommendation where tags were considered as contextual information. Zhang et. al [31] proposed the Explicit Factor Model (EFM) to generate explainable recommendations, meanwhile keeping a high prediction accuracy. Nguyen et.al [18] studied Gaussian Process Factorization Machines (GPFM) for context-aware recommendations using Gaussian processes. Different from this work, we leverage the user latent preference information to achieve better downside management.

Downside management has been studied in recent work of recommender systems. In [29], Wang et. al studied the influence of recommending jobs to users at the right time. They proposed to use a hierarchical proportional hazards model to push recommendations to a user only if the probability of a user applying to a job at the current time is higher than a threshold. In [27], Wang and Zhang proposed the concept of *utility* in the field of recommender system and explored to optimize the user's utility in the e-commerce domain. In [28], Wang and Zhang further studied the effect of finding the right time to push recommendations in the e-commerce domain. They proposed an opportunity model to discover the best opportunity to push recommendations to optimize for the user satisfaction. In this paper, we propose a more general framework for downside management and show the effectiveness of the model in finding the right location to recommend jobs. The framework is general which could be applied to any existing state-of-art recommender systems.

3. USER LATENT PREFERENCE MODEL

3.1 Problem Definition

In this paper, we aim to answer the following questions: 1) How do we discover the user's latent preference(s) 2) How do we leverage it for better downside management? The following notations are used in the paper.

- $u = 1, 2, ..., U$: the index of the user.
- $j = 1, 2, ..., J$: the index of the item.
- $m = 1, 2, ..., M$: the index of the segment for a user. A segment for a user could be determined by his age, gender, region, so on and so forth.
- $D = \{D_1, ..., D_m, ..., D_M\}$: The observed data of all segments from all users.
- $D_m = \{y_{m,i}, \mathbf{x_{m,i}}\}$: A set of observed data associated with segment m. Each segment m has N_m data observations from all users in that segment. Each observation $i = 1, ..., N_m$ in segment m is associated with two parts: the preference label $y_{m,i}$ and the feature vector $\mathbf{x_{m,i}}$.
- $y_{m,i}$: the preference label with the i^{th} observation in segment m. It could be a binary choice (staying at the current location or not) or a multi-class choice (work in IT industry, financing industry, education industry, etc.). The index of the candidate choice is $k = 1, 2, ..., K$.
- $\mathbf{x_{m,i}}$: the d-dimensional vector of features associated with the i^{th} observation in segment m. Features include both static demographic features and dynamic behavior features.

The goal of the model is to predict the probability that user u has the latent preference $y_{m,i} = k$, given that the user belongs to segment m and his feature vector is $\mathbf{x_{m,i}}$.

3.2 Review of Multinomial Regression

Before describing the hierarchical framework that we propose, we first briefly review the basics of multinomial regression.

Multinomial regression is widely used to solve multi-class classification problems with decent performance, e.g. predicting the probability of a user having the latent preference $y_{m,i} = k$ given the user feature vector $\mathbf{x_{m,i}}$. We consider a user's latent preference as a process to choose exactly one candidate at a time. The probability could be estimated based on all available features as follows:

$$p(y_{m,i} = k|\theta) = \frac{exp\{\theta_{\mathbf{k}}^{\mathbf{T}}\mathbf{x_{m,i}}\}}{\sum_{k'} exp\{\theta_{\mathbf{k'}}^{\mathbf{T}}\mathbf{x_{m,i}}\}}$$

where $\theta = \theta_1, ..., \theta_K$ is the model parameter to be learned from the training data. In a basic multinomial regression, all users share the same parameter θ.

Assuming that the prior distribution of each model parameter is a Gaussian centered on zero, the optimal parameters θ can be learned from the training data using the maximum a posteriori probability (MAP) estimation. The multinomial regression model reduces to logistic regression when the number of classes is 2 (positive and negative).

3.3 Model Extension with Bayesian Framework

In this paper we adopt multinomial regression as the core model for user latent preference model and use it to estimate $p(y_{m,i} = k)$. Users in different segments may have different priors for latent preferences. For example, young people tend to relocate more than old people. People in big cities tend to relocate less than those in small cities. Thus we propose to learn one regression model per user segment. Without loss of generality, we do not specify how to segment users in this paper. It could be determined by domain experts or offline experiments.

In the real world, a small number of user segments often have more users while most segments have few users. To address the problem of data sparsity, we follow a common practice and extend the multinomial regression with a hierarchical Bayesian framework as illustrated in Figure 1. This framework helps segments with few observations by borrowing information from other segments through a common prior for parameters of multinomial regression.

For each segment m, θ_m is sampled from a Gaussian distribution: $\theta_m \sim N(\mu_\theta, \mathbf{\Sigma}_\theta)$. We denote $\phi = (\mu_\theta, \mathbf{\Sigma}_\theta)$. For each i^{th} observation in segment m with its observed features $\mathbf{x_{m,i}}$, its latent preference model $y_{m,i}$ is sampled from the multinomial regression model,

$$p(y_{m,i} = k|\theta_m) = \frac{exp\{\theta_{\mathbf{m,k}}^{\mathbf{T}}\mathbf{x_{m,i}}\}}{\sum_{k'} exp\{\theta_{\mathbf{m,k'}}^{\mathbf{T}}\mathbf{x_{m,i}}\}}.$$

Consider that data D consists of a series of observations from all segments. The latent preferences in the observations are generated by using a set of hidden variables $\theta = \{\theta_1, \theta_2 ..., \theta_M\}$. Note that the core model does not need to be multinomial regression. Other suitable models such as the proportional hazards model [29] could be leveraged here as well. θ_m contains all parameters in the core model. The data likelihood can be written as a function of $\phi = (\mu_\theta, \mathbf{\Sigma}_\theta)$. We use y_m to represent $\{y_{m,1}, ..., y_{m,i}, ..., y_{m,N_m}\}$, i.e., preference observations in segment m. Assuming that the data is independent identically distributed, we can present the

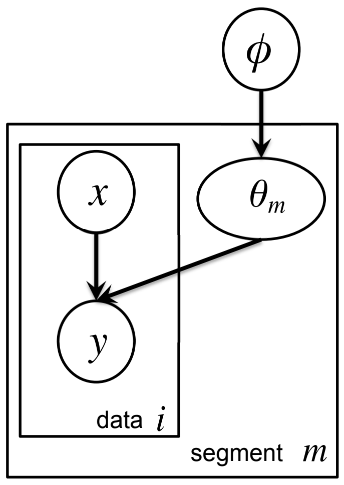


Figure 1: Illustration of dependencies of variables in the hierarchical Bayesian regression model. It shows the i^{th} observation of segment m. $y_{m,i}$ is the label which is dependent on the regression model θ_m of segment m, as well as the observed features $\mathbf{x_{m,i}}$ of this user. Each segment m has its own parameters of multinomial regression θ_m. Models of each segment share information through the prior, $\phi = (\mu_\theta, \mathbf{\Sigma}_\theta)$.

data likelihood as

$$p(D|\phi) = \prod_{m=1}^{M} p(y_m|\phi) = \prod_{m=1}^{M} \int p(\theta_m, y_m|\phi) d\theta_m. \quad (1)$$

3.4 Parameter Inference with the Variational Bayesian Method

There is no closed-form solution for the estimation of model parameters. We follow the variational Bayesian method[3] for constrained (approximate) optimization to derive an iterative process to find an approximate solution. Maximizing the likelihood in Equation 1 is equivalent to maximizing the log likelihood $L(\phi)$,

$$L(\phi) = \ln p(D|\phi) = \sum_{m=1}^{M} \ln p(y_m|\phi) = \sum_{m=1}^{M} \ln \int p(\theta_m, y_m|\phi) \mathrm{d}\theta_m.$$

We can simplify the problem by introducing an auxiliary distribution $q(\theta_m)$ for each hidden variable θ_m [3]. In the variational approach, we constrain $q(\theta_m)$ to be a particular tractable form for computational efficiency. In particular, we assume that $q(\theta_m) = N(\mu_{\theta_m}, \Sigma_{\theta_m})$.

$$\begin{aligned} L(\phi) &= \sum_{m=1}^{M} \ln p(y_m|\phi) \\ &= \sum_{m=1}^{M} \ln \int q(\theta_m) \frac{p(\theta_m, y_m|\phi)}{q(\theta_m)} \mathrm{d}\theta_m \\ &\geq \sum_{m=1}^{M} \int q(\theta_m) \ln \frac{p(\theta_m, y_m|\phi)}{q(\theta_m)} \mathrm{d}\theta_m \\ &\equiv \boldsymbol{F}(q(\theta_1), ..., q(\theta_M), \phi). \end{aligned} \quad (2)$$

Note that $\boldsymbol{F}(q(\theta_1), ..., q(\theta_M), \phi)$ is the lower bound of $L(\phi)$. The process to infer parameters is to iterate between the E-step and M-step until convergence. In particular, the E step maximizes $\boldsymbol{F}(q(\theta_1), ..., q(\theta_M), \phi)$ with respect to $q(\theta_m)$ given the current parameter setting ϕ. The M step maximizes $\boldsymbol{F}(q(\theta_1), ..., q(\theta_M), \phi)$ with respect to ϕ given $q(\theta_m)$ learnt from the E step. Here are the detailed mathematical representation:

E step:

$$q(\theta_m) \leftarrow \arg\max_{q(\theta_m)} \boldsymbol{F}(q(\theta_1), ..., q(\theta_M), \phi), \forall m \in \{1, ..., M\} \quad (3)$$

M step:

$$\phi \leftarrow \arg\max_{\phi} \boldsymbol{F}(q(\theta_1), ..., q(\theta_M), \phi) \quad (4)$$

3.4.1 E-Step

In the **E-step**, maximizing $\boldsymbol{F}(q(\theta_1), ..., q(\theta_M), \phi)$ is equivalent to minimizing the following quantity to find each distribution $q(\theta_m)$.

$$\begin{aligned} q(\theta_m) \leftarrow \arg\min_{q(\theta_m)} & \int q(\theta_m) \ln \frac{q(\theta_m)}{p(\theta_m, y_m|\phi)} \mathrm{d}\theta_m \\ &= \arg\min_{q(\theta_m)} \int q(\theta_m) \ln \frac{q(\theta_m)}{p(\theta_m|\phi)} \mathrm{d}\theta_m - \int q(\theta_m) \ln p(y_m|\theta_m) \mathrm{d}\theta_m \\ &= \arg\min_{q(\theta_m)} KL[q(\theta_m)||p(\theta_m|\phi)] - \int q(\theta_m) \ln p(y_m|\theta_m) \mathrm{d}\theta_m. \end{aligned} \quad (5)$$

The first part in Equation 5 is the KL-divergence between the posterior distribution $q(\theta_m)$ and the prior distribution $p(\theta_m|\phi)$. The KL-divergence between two Gaussian distributions is

$$\begin{aligned} KL[q(\theta_m)||p(\theta_m|\phi)] =& \frac{1}{2}[tr(\Sigma_\theta^{-1}\Sigma_{\theta_m}) + (\mu_\theta - \mu_{\theta_m})^T \Sigma_\theta^{-1} (\mu_\theta - \mu_{\theta_m}) \\ & - \ln(\frac{\det \Sigma_{\theta_m}}{\det \Sigma_\theta}) - d]. \end{aligned}$$

The second part in Equation 5 is to maximize the likelihood of $y_m = \{y_{m,1}, ..., y_{m,i}, ..., y_{m,N_m}\}$ with the current θ_m,

$$\begin{aligned} &\sum_{i=1}^{N_m} \int q(\theta_m) \ln p(y_{m,i} = k|\theta_m) \mathrm{d}\theta_m \quad (6) \\ &= \sum_{i=1}^{N_m} [E(\theta_{m,k})^T \mathbf{x_{m,i}} - E(ln \sum_{k'} exp(\theta_{m,k'}^T \mathbf{x_{m,i}}))]. \end{aligned}$$

The expectation $E(\theta_{m,k})$ in the above equation is $\mu_{\theta_{m,k}}$. There is no closed form solution for the expectation $\delta = E(ln \sum_{k'} exp(\theta_{m,k'}^T \mathbf{x_{m,i}}))$. The moment generating function for the multivariate normal distribution is

$$E[exp(\theta_{m,k'}^T \mathbf{x_{m,i}})] = exp(\mu_{\theta_{m,k'}}^T \mathbf{x_{m,i}} + \frac{1}{2} \mathbf{x_{m,i}}^T \Sigma_{\theta_{m,k'}} \mathbf{x_{m,i}}). \quad (7)$$

By using this function, we get the upper bound of δ with the Taylor expansion as in the following equation,

$$E(ln \sum_{k'} exp(\theta_{m,k'}^T \mathbf{x_{m,i}})) \tag{8}$$
$$\leqslant \gamma \sum_{k'} exp(\mu_{\theta_{m,k'}}^T \mathbf{x_{m,i}} + \frac{1}{2} \mathbf{x_{m,i}}^T \Sigma_{\theta_{m,k'}} \mathbf{x_{m,i}}) - log(\gamma) - 1,$$

for every $\gamma \in \mathbb{R}^K$. Such approximation is widely used in literature [5] for other inference problems.

We can combine the above derivations with Equation 5, then find $q(\theta_m)$ with the conjugate gradient descent method.

3.4.2 M-Step

In the **M-step**, the goal is to maximize $\boldsymbol{F}(q(\theta_1), ..., q(\theta_M), \phi)$ in Equation 2 with respect to ϕ given all $q(\theta_m)$. It is same as to maximize the following quantity:

$$\phi \leftarrow \arg\max_{\phi} \sum_{m=1}^{M} \int q(\theta_m) \ln p(\theta_m, y_m | \phi) \mathrm{d}\theta_m. \tag{9}$$

We derive the closed-form solution for μ_θ and Σ_θ:

$$\begin{aligned}(\mu_\theta, \Sigma_\theta) &\leftarrow \arg\max_{\mu_\theta, \Sigma_\theta} \sum_{m=1}^{M} \int q(\theta_m) \ln p(\theta_m | \mu_\theta, \Sigma_\theta) \mathrm{d}\theta_m \\ &= \arg\max_{\mu_\theta, \Sigma_\theta} \sum_{m=1}^{M} \ln \frac{1}{\sqrt{(2\pi)^k |\Sigma_\theta|}} \\ &\quad - \frac{1}{2} E[(\theta_m - \mu_\theta)^T \Sigma_\theta^{-1} (\theta_m - \mu_\theta)].\end{aligned} \tag{10}$$

By setting the first derivative to zero, we have the following closed form:

$$\mu_\theta = \frac{\sum_m^M \mu_{\theta_m}}{M}$$
$$\Sigma_\theta = \frac{\sum_m^M [\Sigma_{\theta_m} + (\mu_{\theta_m} - \mu_\theta)(\mu_{\theta_m} - \mu_\theta)^T]}{M}.$$

4. USAGE OF THE USER LATENT PREFERENCE MODEL

Let us denote $p(i, j)$ as the joint probability of user i accepting item j. It considers both the match from an existing recommender system and the prediction of the user's latent preference. $p(i, j)$ could be estimated by the following equation:

$$p(i, j) = p_{rec}(i, j) \sum_k [p(y_{m,i} = k) I_{j \in S(i,k)}]. \tag{11}$$

p_{rec} is the matching probability between user i and item j, which could be predicted by any state-of-art recommender systems. $p(y_{m,i} = k)$ is the probability of user i having a specific latent preference k. It is predicted by the user latent preference model, as shown in Equation 12. $S(i, k)$ contains all items that match with the user's preference. For example, if user i is predicted to seek jobs in his local area, $S(i, k = local)$ contains all local jobs. Note that the candidates of a user's preference $k = 1, 2, ..., K$ are all exclusive and item j can only belong to one set $S(i, k)$. I_* is a indicator function which equals to 1 if $*$ is true.

Table 1: The utility set of the recommender system. There are four types of utilities, depending on whether the system recommends the item to the user and whether the user accepts the item.

	show:Y	show:N
accept:Y	u_{TP}	u_{FN}
accept:N	u_{FP}	u_{TN}

$$p(y_{m,i} = k) = E[\frac{exp\{\theta_{\mathbf{k}}^{\mathbf{T}} \mathbf{x_{m,i}}\}}{\sum_{k'} exp\{\theta_{\mathbf{k'}}^{\mathbf{T}} \mathbf{x_{m,i}}\}}] \tag{12}$$
$$E[exp\{\theta_{\mathbf{k}}^{\mathbf{T}} \mathbf{x_{m,i}}\}] = exp\{\mu_{\theta_k}^T \mathbf{x_{m,i}} + \frac{1}{2} \mathbf{x_{m,i}}^T \Sigma_{\theta_k} \mathbf{x_{m,i}}\}.$$

The recommender systems aim to send recommendations to a user to increase user satisfaction. While a good recommendation (upside item) would increase the user satisfaction, a bad recommendation (downside item) would decrease the user satisfaction. Let us consider the utility for each type of recommendation as shown in Table 1. u_{TP} is the utility for a good recommendation that the user accepts. u_{FP} is the utility for a bad recommendation that the user doesn't accept. u_{FN} is the utility for missing a recommendation that the user would accept. u_{TN} is the utility of avoiding a bad recommendation. The utility for each type of recommendations (u_{TP}, u_{FP}, u_{FN}, and u_{TN}) could be set by domain experts in real-world applications. The user i's satisfaction of a recommendation set $utility_i$ is calculated by Equation 13.

$$\begin{aligned} utility_i = \sum (&u_{TP} I_{show,accept} + u_{FP} I_{show,\bar{accept}} \\ &+ u_{FN} I_{\bar{show},accept} + u_{TN} I_{\bar{show},\bar{accept}}). \end{aligned} \tag{13}$$

To achieve a better user satisfaction/utility, the system with a filtering component should send recommendations only if the expected utility of an item is higher than zero. In other words, the system should send recommendations if the joint probability of a user accepting the item $p(i, j)$ is higher than the threshold α. The filtering threshold α is automatically determined by the utility set, which is a common practice as in [28, 29, 9]:

$$\alpha = \frac{u_{FP} - u_{TN}}{u_{FP} - u_{TN} + u_{FN} - u_{TP}}. \tag{14}$$

5. EVALUATION OF THE USER LATENT PREFERENCE MODEL

As described before, the user latent preference model could be applied to a single dimension in any domain. Without loss of generality, we evaluate it in the location dimension in the job domain in the experiment. We observe that some users have a strong preference of being a local user and would complain about receiving non-local job recommendations, and vice versa. *Local jobs* are defined as jobs within 50 miles of the user's current location (determined by his postal code). We treat *local users* as positive labels and *non-local users* as negative labels.

We first list all research questions, followed by the experimental setup, data analysis and performance analysis. Our major research questions for evaluating the user latent preference model include:

- How accurate is the prediction of the user latent preference model? What's the percentage of local users that the model could cover?

- How is the performance comparison for different user segments? Does the hierarchical Bayesian regression model have a better prediction power than the basic regression model?

5.1 Evaluation Metrics

As mentioned before, we label a user as the positive data if the user intends to stay at the current location and otherwise negative. It naturally reduces to a binary classification task. Traditional classification metrics are used to evaluate the prediction power of the user latent preference model: precision, recall and F1 measure. In the following equation, TP, FP and FN correspond to *true positive*, *false positive* and *false negative* respectively.

$$precision = \frac{TP}{TP + FP}$$
$$recall = \frac{TP}{TP + FN}$$
$$F1 = \frac{2}{\frac{1}{precision} + \frac{1}{recall}}$$

5.2 Models to Compare

In our experiments, we compare the basic regression model, as well as the hierarchical regression model with different segments. As shown in Table 2, *M-one* is the basic multinomial regression model with no segments. Other models segment users based on different features.

All models use the same set of features for the core regression model. Features include both the user static information (age, gender, region, current job, current industry, etc.) and the dynamic behavior information (# of local job applications in last month, # of non-local job applications in last month, last profile modification time, etc.). Extensive feature engineering could be explored to improve the model performance, which is beyond the scope of this paper.

5.3 Dataset

As mentioned earlier in the paper, we apply our model in the context of a job recommender system and use a real-world dataset from LinkedIn to evaluate our models. In order to build the user latent preference model, we random sampled from the job application data of United States users for three months from Jan 2014 to March 2014. The sample data contains around 530k users that applied to a reasonable number of jobs in that period [1]. If all jobs that a user applied to are local jobs, we label the user as a local user, and otherwise as a non-local user. In the dataset, 57.22% of users are labeled as local users (positive data) while 42.78% users are labeled as users that are willing to relocate (negative data). 10-cross validation is performed in the experiments. A significance level of 0.05 with the paired two-tailed t-test is used to compare two models.

We further analyze the data by dividing users into different segments (age, gender, region). In Figure 2, the bar plots the percentage of users in each segments. It shows that the data sparsity issue exists (in both age and region dimension). In addition, the line plots the percentage of local users, who only applied to local jobs. It indicates that users in different segments have different priors in relocating. This justifies our motivation for building a hierarchical Bayesian regression model that can 1) let different segments learn from each other to address the problem of data sparsity. 2) let each segment learn its own regression model based on users in the segment.

Some interesting observations include 1) Users around 22 years old are more likely to relocate. It follows the intuition that college graduates would like to change location when they apply for their first jobs. 2) The probability of staying in the same local area doesn't increase for older applicants. 3) Females are slightly more likely to stay in the same local area. 4) Applicants from the big cities/regions tend to stay in the same local area. Top 5 regions that users tend to be local users include San Francisco Bay Area, Greater New York City Area, Houston, Texas Area, Greater Denver Area, and Greater Seattle Area. 5) Applicants from small towns tend to relocate. Top regions that users tend to relocate usually associate with a university or college, including University of South Carolina, University of Wisconsin Madison, Syracuse university, Michigan State University and so on.

The findings from the LinkedIn job application data are consistent with results from demographic surveys [6]. 37% of American adults have never left their hometown, and 57% of American adults have never lived outside their home state. More educated individuals are more likely to have relocated during their lifetimes (77% of college graduates have changed communities at least once). People are most likely to move when they are young, and people from the midwest are less likely to move than those from the Western United States. People who have moved before are much more likely to relocate than those who have never moved. People with higher incomes are more likely to move for job opportunities, while lower income individuals cite different reasons for moving. The demographic data motivates our basic hypothesis that some people will not move under any circumstances, and these users should not be shown non-local job recommendations.

5.4 Performance Analysis

We evaluate the classification performance of different user latent preference models by treating the task as a classification task. We assume that the model predicts a user as a local user if the probability is higher than the classification threshold. We show the performance for different classification thresholds from 0.1 to 0.9. The left plot in Figure 3 shows the performance of the basic regression model *M-one*. It has decent performance in all metrics including precision, recall and F1. High precision ensures that we only filter out non-local jobs for local users. High recall ensures that we can discover most local users and achieve better downside management for these users. F1 metric balances the trade-off between the precision and recall metrics. As demonstrated in the right plot in Figure 3, it is clear that all hierarchical Bayesian regression models help to improve the precision performance significantly. As shown in Table 2, there are different ways of segmenting users. Among all models, *M-age* has the highest lift (5.82% for classification threshold 0.3) with 7 user segments. It is essential to choose the right user segments in the model. Models with too few segments

[1]Exact number is omitted for business concern

Table 2: Different segmentation methods of the user latent preference model.

Model	User segment	# of segments	Sample segments
M-one	none	1	N/A
M-age	age	7	<20 years old, 20-25 years old, 25-30 years old, ...
M-gender	gender	2	female, male
M-region	region	282	San Francisco bay area, Great New York area, ...
M-ageGender	(age, gender)	14	(<20 years old, female), ...
M-ageRegion	(age, region)	1974	(<20 years old, San Francisco bay area), ...
M-regionGender	(region, gender)	564	(San Francisco bay area, female), ...

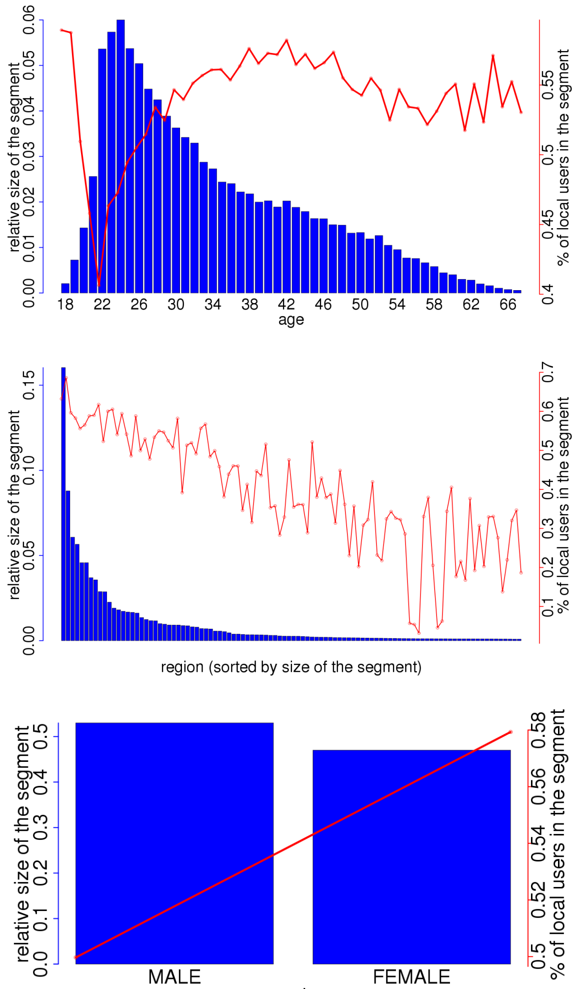


Figure 2: Data analysis of different user segments in the job domain. The bar plots the percentage of users in each segment with the left axis. The line plots the percentage of local users in each segment with the right axis. The three plots correspond to age segments, gender segments, and region segments respectively. Major observations include that 1) the data sparsity issue exists in some user segments 2) users in different segments have different priors in relocating.

(*M-gender*) or too many segments (*M-region, M-ageRegion, M-genderRegion, M-ageGender*) do not work as well as the *M-age* model.

6. EVALUATION OF THE RECOMMENDER SYSTEM

In this section, we incorporate the user latent preference model into the recommender system and evaluate its contribution in downside management. We first list all research questions that we intend to answer, followed by the experimental setup, and performance analysis. Major research questions include:

- Does it achieve higher user satisfaction by adding a filtering component to the basic recommender system?
- Could the user latent preference help to improve the downside management of recommender systems? Does it achieve even higher user satisfaction?
- How about using an extreme filter that filters out all non-local jobs for all users? Does it work as well? What's the contribution of using the prediction from the user latent preference model?

6.1 Models to Compare

In our offline and online experiments, we compare the following models. All models are trained with millions of job applications that were randomly sampled over months and tested with 17K users in the testing period. The threshold of the filtering component is determined by the utility set, as shown in Equation 14.

Baseline is the baseline recommender systems that select a set of items for the user, without considering the user latent preference. It could be any state-of-art recommendation model. We choose logistic regression here as an example. It is a common practice with decent performance and low time complexity [20, 29, 28]. The baseline recommender system does not have a filtering component. It always sends a set of top K items to each user (K is set to be 5). When we build the baseline logistic regression model, we leverage basic features including similarity-related features between the user profile (including the user's working experience, education information, etc) and the job information (including the job's title, description, etc), the user behavior features, and so on. Features include location-related features as well, such as the transition probability from the user's location to the job's location and so on.

Baseline.F applies a filter on the recommendation set from *Baseline* based on the recommendation probability $p_{rec}(i,j)$ without considering the user's latent preference.

Baseline+Latent.F applies a filter on the recommendation set based on the joint probability $p(i,j)$ which considers the user latent relocation preference. As a result, the model recommends local jobs to users that prefer to stay in the local area and non-local jobs to other users.

Baseline+AllLocal.F applies a filter on the recommendation set based on the joint probability $p(i,j)$. It assumes that all users are local users. In other words, it estimates $p(y_{m,i} = local) = 1$ for all users. As a result, the model recommends local jobs to all users.

6.2 Offline Testing Setup

As we discussed in Section 4, the user satisfaction is affected by both good recommendations and bad recommendations. The corresponding evaluation metric is the average utility which truly reflects the user satisfaction. Assume that there are U users to send recommendations in the testing period. The average utility/user satisfaction *utility* can be calculated as following: $utility = \frac{\sum_{i=1}^{U} utility_i}{U}$.

Unlike traditional metrics such as the number of applications/views, the utility metric considers both the positive effect of *good* recommendations and the negative effect of *bad* recommendations. The higher the utility, the better the model.

Table 3: Average utility of recommendations in the offline scenario. There are 17K users in the test dataset who applied to 2.12 jobs on average. *Rec coverage* is the percentage of users that are recommended at least one item. *Rec count* is the average number of unique recommendations that are pushed to each user.

utility set 1
u_{TP} = 2, u_{FN} = -1
u_{FP} = -1, u_{TN} = 0
Threshold = 0.25

Model	Baseline	Baseline.F	Baseline+Latent.F	Baseline+AllLocal.F
Utility	0.590	1.300	**1.895**	1.235
Rec coverage	100%	98.4%	95.0%	74.8%
Rec count	5	4.282	3.488	2.852

utility set 2
u_{TP} = 2, u_{FN} = -1
u_{FP} = -2, u_{TN} = 0
Threshold = 0.4

Model	Baseline	Baseline.F	Baseline+Latent.F	Baseline+AllLocal.F
Utility	-2.479	-0.206	**1.039**	0.500
Rec coverage	100%	96.6%	88.9%	74.2%
Rec count	5	3.882	2.969	2.831

utility set 3
u_{TP} = 2, u_{FN} = -2
u_{FP} = -1, u_{TN} = 0
Threshold = 0.2

Model	Baseline	Baseline.F	Baseline+Latent.F	Baseline+AllLocal.F
Utility	0.393	0.951	**1.460**	0.486
Rec coverage	100%	99.0%	96.3%	75.6%
Rec count	5	4.460	3.729	2.854

6.3 Online A/B Testing Setup

In the online A/B testing, we evaluate the performance of recommender systems with real-world users on LinkedIn. We randomly select 5% United States users [2] that visit LinkedIn for each model and present the corresponding recommendations to each user group. The difference of the performance between two user buckets are reported in the performance analysis. A significance level of 0.05 with the paired two-tailed t-test is used to compare two models. We let each model to run for one week to burn in the novelty effect (active job seekers tend to apply to any new jobs they see due to a change in the recommendation model) and start the comparison after that.

In the LinkedIn product, summaries of job vacancy postings are shown to the user. We call this an **impression** (each job vacancy summary shown to the user is considered to be impressed). Job seekers can open a new page to view the entire job description, which we call as a **view** action. In addition, users will apply to the job if they wish to be considered for the position, which we call as an **apply** action. We report metrics that measure the model's performance in both upside management and downside management: the number of impressions, the number of views and applications, API ($\frac{\#\ of\ applications}{\#\ of\ impressions}$), and VPI ($\frac{\#\ of\ views}{\#\ of\ impressions}$).

6.4 Performance Analysis

6.4.1 Offline Experiments

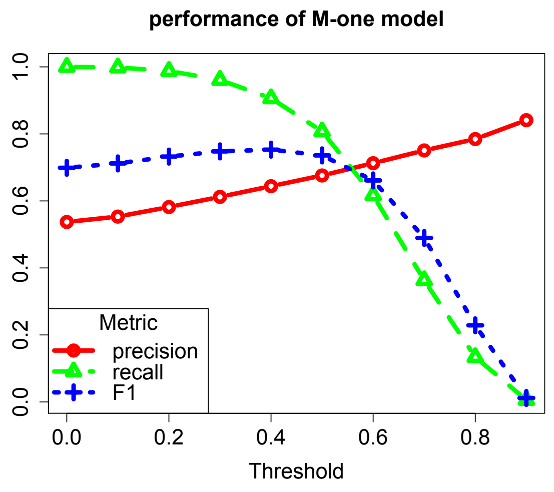
We first evaluate all models in the offline scenario. The testing data include 17K users who applied to 2.12 jobs on average. Three different sets of utilities are tested, as shown in Table 3. Each set reflects different scenarios (depending on a user's tolerance to bad recommendations) and it could be tuned by domain experts in real-world applications.

In the *first* utility set, the utility u_{TP} for a good recommendation that the user applies to is set to 2. When the system shows a job and the user does not apply to it, the utility u_{FP} is -1. The penalty u_{FN} for missing a good recommendation is set as -1. The resulting filtering threshold is 0.25. Model that applies a filter all perform better than the *Baseline* model without filtering. Among all three models that have a filtering component, both $Baseline + Latent.F$ and $Baseline + AllLocal.F$ perform better than the $Baseline.F$ model by filtering out many downside items (i.e., jobs that are not in the user's preferred area). It follows our intuition that the probability $p(i,j)$ with latent preference is a stronger signal than the pure recommendation probability $p_{rec}(i,j)$. In addition, $Baseline + Latent.F$ model still keeps potentially good jobs that a non-local user wants to apply to, which leads to a higher utility than $Baseline + AllLocal.F$ model. Note that the average number of recommendations of $Baseline + Latent.F$ is significantly higher than the $Baseline + AllLocal.F$ model.

In the *second* utility set, the utility u_{FP} is set to -2, indicating that the user is less tolerant to downside items. Thus the threshold of recommending an item is higher, being 0.4. Not surprisingly, the *Rec coverage* of $Baseline + Latent.F$ decreases than that in the first set of utility. It still performs the best among all four models.

In the *third* utility set, u_{FN} is set to -2. It is the penalty for not recommending a good job that would be applied by the user. $Baseline + Latent.F$ model still achieves the highest utility among all models. $Baseline + AllLocal.F$ has a lower utility since it misses some potentially good recommendations for non-local users.

[2]By the end of 2014, there are more than 111 million members in United States in LinkedIn.



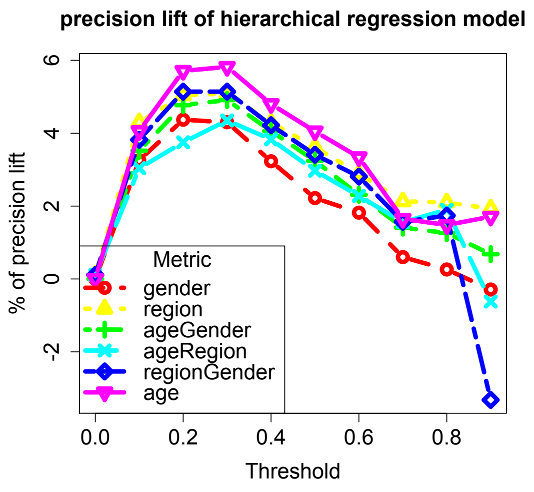


Figure 3: Performance of the user latent preference model in the classification task. The left plot shows the performance of the baseline multinomial regression in precision, recall and F1. The right plot shows the precision lift of hierarchical Bayesian regression models.

Table 4: The performance comparison of online A/B testing. The number shows the lift of the model. Numbers in bold are significantly better than the value in the corresponding baseline model. The absolute numbers are omitted for business concern.

Model	# of impressions	# of applications	# of views	API	VPI
Baseline+Latent.F vs. Baseline	**-11.1%**	+0.5%	-0.6%	**+12.9%**	**+11.7%**
Baseline+AllLocal.F vs. Baseline	**-18.4%**	**-10.4%**	**-6.1%**	**+9.7%**	**+15%**
Baseline+Latent.F vs. Baseline+AllLocal.F	**+8.9%**	**+12.1%**	**+5.8%**	+2.9%	-2.9%

6.4.2 Online Experiments

In the online A/B testing, the performance follows our intuition as shown in Table 4. *Baseline + Latent.F* model has a significantly better downside management than the *Baseline* model. In details, *Baseline + Latent.F* model shows less impressions to users. The corresponding API and VPI metrics increase significantly while the number of applications and views remain stable. It shows that downside items that are not interesting to users are removed from the recommendation set. The performance is consistent with that in offline experiments.

If we filter out non-local recommendations for all users, the performance of such model *Baseline+AllLocal.F* is shown in Table 4. As expected, it shows much fewer impressions. Although VPI and API increase significantly, the absolute number of applications and views drop. Such a model might filter out some good non-local recommendations for users who are willing to relocate. For example, recent college graduates would prefer to relocate to other cities for their first job. In this case, *Baseline+AllLocal.F* still recommends local jobs to these users. Yet the user latent preference model *Baseline + Latent.F* would treat these users as non-local users and recommend jobs in non-local areas.

For a real-world recommender system, one goal is to improve the upside management, which is reflected by the *# of applications* and *# of views* metrics. A good upside management helps users to discover items that they like, which saves users' time and effort. While good upside management lets users engage themselves with the system, it is equally important for a recommender system to have a good downside management. It aims to avoid showing those obviously terrible items to users. For example, if a user would like to stay in the San Francisco Bay Area, a good recommender system shouldn't recommend a job in the New York city even if the job is a perfect match. Good downside management is reflected in the high *API* and *VPI*. It makes recommendations more explainable and transparent to users, which in turn helps to increase the user satisfaction.

7. CONCLUSION AND FUTURE WORK

In this paper, we propose to develop the *user latent preference model* to predict the probability of a user having a specific preference. It is a general framework that could be applied to any existing recommender system in any domain (including job search, e-commerce, movie recommendations, etc.). The model uses multinomial regression to predict the user's preference among all candidates. We build a regression model for each user segment. In order to tackle the data sparsity issue, we further extend the model with the hierarchical Bayesian framework. The variational Bayesian inference algorithm helps to make the parameter estimation more computationally efficient. In the recommendation stage, we leverage the prediction of the user's latent preference to estimate the joint probability for a user accepting an item. We show the effectiveness of the user latent preference model in the context of recommending the right job at the right location. Both offline experiments and online A/B testing demonstrate that our proposed model has a better downside management, which leads to higher user satisfaction.

This is just the first step to tackle the downside management in recommender systems. In this paper, we propose to use multinomial regression as the core model in the user latent preference model. Other distributions or models could be explored for other dimensions. In addition, the current model predicts a user's latent preference in one dimension, such as time, location and so on. It would be interesting to explore how to integrate a user's latent preferences in multiple dimensions to provide better recommendations. Last but not least, the model could be evaluated in other domains to show its effectiveness in downside management.

8. REFERENCES

[1] G. Adomavicius and A. Tuzhilin. Context-aware recommender systems. In *Recommender systems handbook*, pages 217–253. Springer, 2011.

[2] L. Baltrunas and X. Amatriain. Towards time-dependant recommendation based on implicit feedback. In *Context-aware Recommender Systems Workshop at RecSys09*. ACM, 2009.

[3] M. Beal. *Variational algorithms for approximate Bayesian inference*. PhD thesis, University of London, 2003.

[4] J. Bennett and S. Lanning. The netflix prize. 2007.

[5] D. M. Blei and J. D. Lafferty. A correlated topic model of science. *The Annals of Applied Statistics*, pages 17–35, 2007.

[6] D. Cohn and R. Morin. Who Moves? Who Stays Put? Where's Home? Technical report, Dec 2008.

[7] P. Cremonesi, Y. Koren, and R. Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the fourth ACM RecSys 2010*, pages 39–46. ACM, 2010.

[8] L. Do and H. W. Lauw. Modeling contextual agreement in preferences. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14, pages 315–326. International World Wide Web Conferences Steering Committee, 2014.

[9] C. Elkan. The foundations of cost-sensitive learning. In *Proceedings of the 17th IJCAI'01*, IJCAI'01, pages 973–978. Morgan Kaufmann Publishers Inc., 2001.

[10] N. Golbandi, Y. Koren, and R. Lempel. Adaptive bootstrapping of recommender systems using decision trees. In *Proceedings of the fourth ACM WSDM*, WSDM '11, pages 595–604. ACM, 2011.

[11] N. Hariri, B. Mobasher, and R. Burke. Query-driven context aware recommendation. In *Proceedings of the 7th ACM Conference on RecSys*, RecSys '13, pages 9–16, New York, NY, USA, 2013. ACM.

[12] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd annual international ACM SIGIR*, pages 230–237. ACM, 1999.

[13] R. Jin, L. Si, C. Zhai, and J. Callan. Collaborative filtering with decoupled models for preferences and ratings. In *Proceedings of the twelfth CIKM*, pages 309–316. ACM, 2003.

[14] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '08, pages 426–434. ACM, 2008.

[15] X. Liu and K. Aberer. Soco: A social network aided context-aware recommender system. In *Proceedings of the 22nd International Conference on World Wide Web*, WWW '13, pages 781–802. International World Wide Web Conferences Steering Committee, 2013.

[16] B. Marlin and R. S. Zemel. The multiple multiplicative factor model for collaborative filtering. In *Proceedings of the 21st ICML '04*, page 73. ACM, 2004.

[17] R. J. Mooney and L. Roy. Content-based book recommending using learning for text categorization. In *DL '00*, pages 195–204. ACM, 2000.

[18] T. V. Nguyen, A. Karatzoglou, and L. Baltrunas. Gaussian process factorization machines for context-aware recommendations. In *Proceedings of the 37th International ACM SIGIR Conference*, SIGIR '14, pages 63–72. ACM, 2014.

[19] U. Panniello, A. Tuzhilin, M. Gorgoglione, C. Palmisano, and A. Pedone. Experimental comparison of pre- vs. post-filtering approaches in context-aware recommender systems. In *Proceedings of the Third ACM Conference on RecSys*, RecSys '09, pages 265–268. ACM, 2009.

[20] D. Parra, A. Karatzoglou, X. Amatriain, and I. Yavuz. Implicit feedback recommendation via implicit-to-explicit ordinal logistic regression mapping. In *Proceedings of the CARS-2011*, 2011.

[21] D. Parra-Santander and P. Brusilovsky. Improving collaborative filtering in social tagging systems for the recommendation of scientific articles. *Web Intelligence and Intelligent Agent Technology*, 1:136–142, 2010.

[22] M. Pazzani, D. Billsus, S. Michalski, and J. Wnek. Learning and revising user profiles: The identification of interesting web sites. In *Machine Learning*, pages 313–331, 1997.

[23] E. Shmueli, A. Kagian, Y. Koren, and R. Lempel. Care to comment?: recommendations for commenting on news stories. In *Proceedings of the 21st WWW*, WWW '12, pages 429–438. ACM, 2012.

[24] L. Si and R. Jin. Flexible mixture model for collaborative filtering. In *Proceedings of ICML*, pages 704–711. AAAI Press, 2003.

[25] K. Verbert, N. Manouselis, X. Ochoa, M. Wolpers, H. Drachsler, I. Bosnic, and E. Duval. Context-aware recommender systems for learning: a survey and future challenges. *Learning Technologies, IEEE Transactions on*, 5(4):318–335, 2012.

[26] J. Wang, B. Sarwar, and N. Sundaresan. Utilizing related products for post-purchase recommendation in e-commerce. In *Proceedings of the fifth ACM conference on Recommender systems*, RecSys '11, pages 329–332. ACM, 2011.

[27] J. Wang and Y. Zhang. Utilizing marginal net utility for recommendation in e-commerce. In *Proceedings of the 34th international ACM SIGIR'11*, pages 1003–1012. ACM, 2011.

[28] J. Wang and Y. Zhang. Opportunity model for e-commerce recommendation: right product, right time. In *Proceedings of the 36th international ACM SIGIR Conference*, SIGIR '13. ACM, 2013.

[29] J. Wang, Y. Zhang, C. Posse, and A. Bhasin. Is it time for a career switch. In *Proceeding of the 22nd International Conference on World Wide Web*, WWW '13. ACM, 2013.

[30] J. Weston, R. J. Weiss, and H. Yee. Nonlinear latent factorization by embedding multiple user interests. In *Proceedings of the 7th ACM Conference on RecSys*, RecSys '13, pages 65–68. ACM, 2013.

[31] Y. Zhang, G. Lai, M. Zhang, Y. Zhang, Y. Liu, and S. Ma. Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In *Proceedings of the 37th International ACM SIGIR Conference*, SIGIR '14, pages 83–92. ACM, 2014.