Learning to Rank Features for Recommendation over Multiple Categories

Xu Chen[1] Zheng Qin[2] Yongfeng Zhang[3] Tao Xu[4]

[124] School of Software,Tsinghua National Laboratory for Information Science and Technology
Tsinghua University, Beijing,10084,China
{xu-ch14,xut14,qinzh}@mails.tsinghua.edu.cn
[3] Department of Computer Science & Technology,Tsinghua University, Beijing,10084,China
yongfeng14@mails.tsinghua.edu.cn

ABSTRACT

Incorporating phrase-level sentiment analysis on users' textual reviews for recommendation has became a popular method due to its explainable property for latent features and high prediction accuracy. However, the inherent limitations of the existing model make it difficult to (1) effectively distinguish the features that are most interesting to users, (2) maintain the recommendation performance especially when the set of items is scaled up to multiple categories, and (3) model users' implicit feedbacks on the product features. In this paper, motivated by these shortcomings, we first introduce a tensor matrix factorization algorithm to **L**earn to **R**ank user **P**references based on **P**hrase-level sentiment analysis across **M**ultiple categories (LRPPM for short), and then by combining this technique with Collaborative Filtering (CF) method, we propose a novel model called LRPPM-CF to boost the performance of recommendation. Thorough experiments on two real-world datasets demonstrate that our proposed model is able to improve the performance in the tasks of capturing users' interested features and item recommendation by about 17%-24% and 7%-13%, respectively, as compared with several state-of-the-art methods.

Keywords

Recommender Systems; Sentiment Analysis; Collaborative Filtering; Tensor Factorization

1. INTRODUCTION

Emerging popularity of e-commerce has highlighted the importance of recommendation systems, with many models have been studied. Among these models, latent factor models [2, 5, 11, 20] have gained much attention from the research community and industry due to their good prediction accuracy on some benchmark datasets. However, recommendation based on these methods could hardly give intuitional explanations for the latent features which weakens the ability to persuade users and help users make better decisions in practical systems.

Figure 1: An example of feature ranking with ratings predicted by rating-based optimization functions.

In recent years, there has been an upsurge of interest in exploiting the users' textual review information to enhance the interpretability of latent factor models. A well-known example is the HFT model [6]. By linking the latent factors leveraged in modeling user rating behavior with the topics presented in the user comments, this model directly explain the latent factors by the user mentioned topics. However, as a topic may contain products' different features and a user could express different opinions for various features in the same topic, simple topic-level analysis without more detailed natural language processing makes such approaches biased and limited.

Several attempts to construct a more explainable recommendation model have been made very recently in [17, 18]. In their models, phrase-level sentiment analysis are used to excavate products' explicit features and users' corresponding opinions.For instance, if a user purchased a phone, and made a 5 star rating with the text review "screen is perfect, but earphone is bad!" The methods proposed in [17, 18] could capture the feature-sentiment pairs like (screen, perfect) and (earphone, bad), which would provide finer-grained analysis of the textual reviews as compared with HFT, and could help to make more accurate recommendations.

Despite that encouraging improvements have been brought by these models, they still suffer from three issues. First, selecting the most interesting features is an important component in their models for user profiling, however the existing rating-based optimization target is not suitable for such an inherently ranking task. Second, they both consider a user's extent of interest towards each feature as static over all products.This setting is not practical in real-world

SIGIR '16, July 17-21, 2016, Pisa, Italy

DOI: http://dx.doi.org/10.1145/2911451.2911549

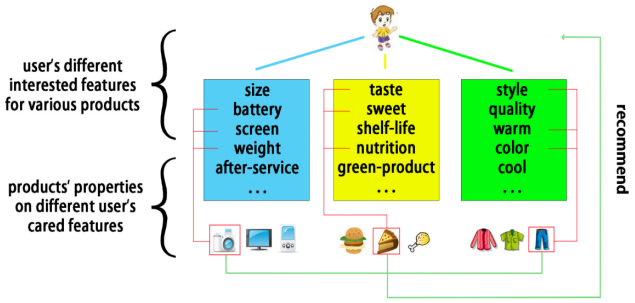

Figure 2: User's different features for various products are first extracted utilizing phrase-level sentiment analysis, and then the items which perform well on the user's cared features would be recommended.

scenarios, especially when the products are from different categories. Finally, instead of modeling users' implicit feedback(mentioned a feature in the review or not)directly, these models convert the implicit feedbacks into rating scores, which would bias it in terms of preference estimation [14].

In this paper we describe and analyze a general method to learn and infer user preferences from ratings along with textual reviews. The main building block of our proposed method is an effective tensor-matrix factorization algorithm to **L**earn to **R**ank user **P**references based on **P**hrase-level sentiment analysis across **M**ultiple categories (LRPPM). By learning to rank, we wish to make our model more suitable for ranking tasks in selecting the most interesting features for each user. By tensor-matrix factorization, we wish to capture users' different interests to various products directly from their implicit feedback, and to keep the recommendation performance even when the set of products scales up to multiple categories. Furthermore, by bridging this method with traditional Collaborative Filtering (CF) approaches, we build a novel LRPPM-CF algorithm to boost the performance of personalized recommendation.

Compared with the existing models, the key advantages of our proposed method are: (1) It introduces ranking-based optimization objective to replace rating-based target for better understanding user preferences on feature-level. In Figure 1 for instance, where our goal is to select the top-three favourite features for a user from the feature pool $\{A, B, C, D, E\}$. Suppose that the real scores the user would make on these features are $\{5, 4, 2, 3, 1\}$, and we have two strategies that predict the feature scores by $\{4, 5, 3, 2, 1\}$ and $\{3, 2, 1, 2, 1\}$, respectively. From the perspective of rating-based optimization, the former strategy is better because it gains higher prediction accuracy (lower root mean square error(RMSE)), and thus the features A, B, C would be selected. However, the latter strategy is actually more preferred because it selects A, B, D, which are the top favourite features according to the real ratings, although this strategy gains lower prediction accuracy according to RMSE. (2) It extracts the different favoured features for each user more precisely, which can further make our model more accurate when making recommendations. In Figure 2 for example, our model would value more on the screen pixel when a user is selecting digital cameras, while the style might be considered as a more important feature when he/she is looking for clothes. (3) It is able to model the user implicit feedbacks directly to capture the user preferences.

In the rest of the paper, we first review the related work in section 2, and give detailed explanation of our methods in section 3. Another rating-based tensor factorization method is explored in section 4 for model comparison. And then in section 5, we describe our experiments and analyze the effectiveness of our methods. The conclusions and outlook of this work are presented in section 6.

2. RELATED WORK

With the ever growing amount of user generated textual reviews, the problem of how to leverage such information-rich resources to construct more explainable recommendation models has received increasing attention. Until recently, a lot of recommendation algorithms [1, 3, 10, 12, 18] based on sentiment analysis of textual reviews have been proposed. In general, these algorithms are conducted on three different levels: (1) review-level, (2) sentence-level, and (3) phrase-level.

Review- or sentence-level methods. These methods take a review or a sentence as a whole, and analyze its sentiment directly. The method proposed in [1] presents three approaches to identify movie aspects which are used as features for collaborative filtering. [3] considers different sentimental orientations (SO) of similar words in different scenarios, and proposes a framework to determine the sentimental orientations. [10] focuses on the situation of user textual review without explicit rating labels, and introduces a sentiment-aware nearest neighbor model to boost the performance of existing recommendation methods. Finally, [12] constructs an opinion matrix by extracting user interests from the reviews, and combines this matrix with traditional model-based collaborative filtering to provide rating predictions.

The models mentioned above have made great contributions to the modeling of user review information in recommendation tasks. However, such review- or sentence-level approaches could not explicitly identify the product features in a review/sentence, and therefore fail to capture user preferences in a finer-grained manner towards specific features, which can be very important to generate personalized recommendations and explanations.

Phrase-level methods. A recent approach proposed in [18] utilizes phrase-level sentiment analysis to extract the explicit product features and finer-grained per feature sentiment information from user reviews. Specifically, it constructs a sentiment lexicon from a corpus of reviews, based on which to further generate an explicit user-feature attention matrix and item-feature quality matrix, where the former reflects user interests towards different features, and the latter shows the quality of each product on each feature. It then adopts matrix factorization techniques to complete these matrices by optimizing the root mean square error (RMSE). At last, this model makes recommendations by matching the most favourite features of a user with the intrinsic properties of items.

Though this model could provide more detailed user preferences and more accurate predictions, the simple matrix factorization approach to optimize RMSE as a rating-based task makes it limited in that: (1) the estimation of user-feature attentions is inherently a ranking-based task to select the favourite features, (2) it fails to distinguish users' different interests of features on different products, and (3) it converts users' implicit feedbacks on product features into

explicit rating scores, which could be a step that introduces inaccuracy. In contrast, in this paper, we propose a ranking-based tensor-matrix factorization algorithm to resolve these problems, which makes more practical, explainable, and accurate recommendations.

3. THE FRAMEWORK

In this section, we describe the major components of our model. Firstly,we give a brief introduction to the extraction of feature-sentiment pairs from user textual reviews. And then we analyze the Yelp[1] dataset to verify our assumption that users usually care about different features for different product categories. Next, we elaborate the main components of our LRPPM model. At last, by combing this technique with collaborative filtering, we propose a novel framework for purchase prediction and recommendation. Specifically, we implement our framework on both product-level and category-level, respectively, so as to make it easy to compare and understand the performance of our framework.

3.1 Extracting Feature-Sentiment Pairs

In the first stage, we construct the set of feature-sentiment pairs from a corpus of textual reviews based on the state-of-art optimization approach described in [4, 18, 19] due to its high accuracy. Specifically, we first extract feature word set F from all the user textual reviews. Then for a given piece of review, we generate a set of feature-sentiment pairs (F, S) to represent this review, where S is assigned as 1 or -1 according to the sentiment polarity that the user expressed on this feature. For example, for the piece of review 'the taste is perfect, but the appearance is ugly!', the extracted feature-sentiment pairs can be (taste, +1) and (appearance, -1). Since the feature-sentiment pair extraction is not the key contribution of this paper, we refer the readers to the related literature such as [4, 18, 19], and focus our attention on the next stages of user preference prediction and recommendation.

3.2 Data Analysis and Case Studies

We adopt the Yelp dataset for analysis because it covers items from multiple categories, which matches with our research tasks. We focus our analysis on the categories of $beauty, entertainment, food, health, clothing$ and $bars$ due to their high co-occurrence frequencies in user transactions. To verify the hypothesis that users usually care about different features for various products, we select the most popular features in every category according to the frequencies they are mentioned in the textual reviews. In this dataset, we select top-10 most cared features for empirical analysis, as shown in Table 1. Based on simple observations we can find that:

(1) On pairwise level, only three pairs of categories contain three common features, and the others have at most two common features or no intersection at all.

(2) Among all these 47 mentioned features, only three appear in more than three categories.

These observations and case studies imply that the user interests may vary with the categories, which is an important motivation for us to model the user preferences and recommendation with the discrimination of different categories. In the next subsections, we introduce our LRPPM framework

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

Table 1: The top-10 most cared features in user reviews of different categories.

category	beauty	entertainment	food
features	'hair'	'show'	'food'
	'staff'	'room'	'service'
	'job'	'rooms'	'pizza'
	'massage'	'seats'	'taste'
	'service'	'staff'	'cheese'
	'room'	'hotel'	'location'
	'salon'	'theater'	'staff'
	'nails'	'location'	'flavor'
	'spa'	'bar'	'fries'
	'location'	'tickets'	'prices'

category	health	clothing	bars
features	'staff'	'store'	'food'
	'care'	'selection'	'bar'
	'appointment'	'prices'	'service'
	'patient'	'staff'	'def'
	'doctor'	'deals'	'wait'
	'location'	'shoes'	'drinks'
	'office'	'price'	'hour'
	'paperwork'	'quality'	'music'
	'dentist'	'clothes'	'staff'
	'pain'	'items'	'table'

so as to Learn to Rank user Preferences based on Phrase-level sentiment analysis across Multiple categories, and further integrate this framework with Collaborative Filtering on both product- and category-levels.

3.3 The LRPPM model

The most direct yet naive implementation to model user interests in different categories is to conduct user/item profiling on each category independently, for example, by analyzing the reviews from each category with EFM [18] in isolation. However, with the large and ever growing number of categories, this method is practically infeasible. To alleviate the problem of both effectiveness and efficiency, we propose a unified framework based on tensor factorization in this section, and the major components of this framework would be introduced in detail in the following.

Factorizing User-Item-Feature Cube. To capture users' different favored features for various items, we should model interactions among users, items and features simultaneously, this inspires us to introduce tensor matrix factorization method, and for clear exposition, let $U = \{u_1, u_2, ...u_{|U|}\}$ be the set of users, $I = \{i_1, i_2, ...i_{|I|}\}$ be the set of items and $F = \{f_1, f_2, ...f_{|F|}\}$ be the set of features extracted from the textual reviews. Then we represent users' reviewing behavior as a set of user-item-feature triplets as follows:

$$O := \{(u, i, f) | u \in U, i \in I, f \in F, User\ u\ mentioned\ feature\ f\ in\ his/her\ review\ on\ item\ i.\} \quad (1)$$

We consider the user-item-feature relationship as an interaction cube T,where the element reflects the extent that a user is interested in a feature when he/she reviewed on an item. For more accurate modeling, we model the implicit feedback of users directly by labeling the triplets from O as observed elements in the corresponding position of cube T. Note that, instead of being converted to explicit rat-

ings(e.g.'1'), these observed elements would be employed to construct preference pairs in the next section.

A number of techniques exist for tensor factorization [15, 16, 13], and we adopt a method similar to [15] due to its efficiency and relatively lower learning complexity. In our method, the pairwise interactions among users, items and features are modeled directly, and the scoring function which reflects $u's$ interest for $i's$ feature f is shown as follows:

$$\hat{T}_{uif} = \sum_{k=0}^{K-1} R^U_{uk} \cdot R^{UF}_{fk} + \sum_{k=0}^{K-1} R^I_{ik} \cdot R^{IF}_{fk} + \sum_{k=0}^{K-1} R^U_{uk} \cdot R^I_{ik} \tag{2}$$

where $R^U \in \mathbb{R}_+^{|U|\times K}$, $R^I \in \mathbb{R}_+^{|I|\times K}$, $\{R^{UF} \in \mathbb{R}_+^{|F|\times K}$, $R^{IF} \in \mathbb{R}_+^{|F|\times K}\}$ are the latent matrices of users, items and features respectively, and K represents the number of factors in these matrices.

Ranking-based optimization target on implicit feedback. As selecting the favourite features for each user is an inherently ranking-oriented task, we should care more about users' relative preference on different features rather than the explicit rating predictions on them, and that a ranking-based approach can be more suitable than the rating-based criterion used in [18].

In our model, we use the ranking-based criterion of Bayesian Personalized Ranking (BPR) [14] as our optimization goal. Intuitively, a user would generally review on his/her cared features, while the features not mentioned in his/her comments, in turn, are not attractive to him/her, so to conduct BPR method, we build preference pairs between the observed elements in T which correspond to interested features with the non-observed ones, we define:

$$\hat{T}_{uif_Af_B} = \hat{T}_{uif_A} - \hat{T}_{uif_B} \tag{3}$$

which reveals user $u's$ interests for feature f_A over f_B when he reviewed item i. Then we use a similar method in [14] to estimate our model parameters by maximizing a log posterior (MAP):

$$\hat{\Theta} = \arg\max_{\Theta} \sum_{u\in U}\sum_{i\in I}\sum_{f_A\in F^+_{ui}}\sum_{f_B\in F^-_{ui}} \log\sigma(\hat{T}_{uif_Af_B}) - \lambda_\Theta \|\Theta\|_F^2 \tag{4}$$

where $\sigma()$ is a logistic sigmoid function, F^+_{ui} is the set of features that were mentioned by user u for item i, F^-_{ui} is the set of features that were not mentioned, namely, $F^+_{ui} = \{f \mid (u,i,f) \in O\}$ and $F^-_{ui} = \{f|(u,i,f) \notin O\}$, Θ is the model parameter and λ_Θ is the regularization constant.

After estimating the parameters,we could rank the features according to equation (2) and select the features with top-N scores to model the user preferences on a specific item.

3.4 The Integrated LRPPM-CF Model

In this section, we incorporate both rating and reviewing information together to build a hybrid model. Specifically, we first share the latent factors used for modeling user rating behavior with the factors embedded in user reviewing behavior, and then propose a unified LRPPM-CF framework by combining our model LRPPM with model-based collaborative filtering method, which attempts to optimize the following objective:

$$\min_{\Theta} \sum_{u\in U}\sum_{i\in I}(A_{ui} - (R^U_{u\cdot})^T \cdot R^I_{i\cdot})^2 - \lambda \sum_{u\in U}\sum_{i\in I}\sum_{f_A\in F^+_{ui}}\sum_{f_B\in F^-_{ui}} ln\sigma(\hat{T}_{uif_Af_B}) + \lambda_\Theta \|\Theta\|_F^2 \tag{5}$$

where $R^U_{u\cdot}$ is the u-th row of R^U, $(R^U_{u\cdot})^T$ is the transpose of $R^U_{u\cdot}$, $R^I_{i\cdot}$ is the i-th row of R^I, Θ is the model parameter, A_{ui} is the rating that user u gives to item i, λ is the tuning parameter that balances the weight between the two types of behaviors, and λ_Θ is the regularization constant.In this expression, we model the user ratings with latent tensor factors by the first term, and further model the user preferences on features embedded in reviews through learning to rank by the second term. The parameters are regularized by a unified way in the last term.

3.4.1 Model Learning for LPRRM-CF

There is no closed-form solution for equation (8), and we introduce a stochastic gradient descent algorithm to find the optimal solution for the parameters $\Theta = \{R^U, R^I, R^{UF}, R^{IF}\}$. For convenience, we define the training set as $D_S = \{(u,i,f_A, f_B)|(u,i,f_A) \in O,\ (u,i,f_B) \notin O\}$ and the decaying parameter $l_{ui} = \frac{1}{|F^+_{ui}|*|F^-_{ui}|}$in the optimization algorithm shown in Algorithm 1. In this algorithm, we first initialize the parameters, and then update these parameters repeatedly until convergency or reaching the maximum number of iterations.

3.4.2 Making Recommendations

Given the optimal solutions of $\{R^U, R^I, R^{UF}, R^{IF}\}$, we estimate the user-item-feature cube as $\hat{T}$ and user-item rating matrix $\hat{A}$. In order to make recommendations, we consider two aspects:(1)the direct rating that a user would score on an item and (2) the compatibility between a user's interested features and an item's high-quality features.

A user $u's$ estimated rating for item i could be readily derived as $R^{rating}_{ui} = \hat{A}_{ui}$. To evaluate the consistency between a user's favorite features and an item's intrinsic properties. We first estimate items' quality on different features. Suppose item i's quality on various features is defined as $\{s_{i1}, s_{i2}, ...s_{i|F|}\}$. Item $i's$ feature j is mentioned with sentiments $\{m_{ij1}, m_{ij2}, ...m_{ijn_j}\}$, where n_j is the number of times feature j is mentioned in all the reviews of item i, $m_{ijk} \in \{-1, 1\}$ for $k = 1, 2, ...n_j$. Then we evaluate s_{ij} by $\frac{1}{1+e^{-\sum_{l=1}^{n_j} m_{ijl}}}$ if item i is reviewed on feature j and assign it as 0 otherwise.

We assume that a user's decision about whether or not to make a purchase is based on several important product features to him or her, rather than considering all hundreds of possible features. For a given user-item pair (u,i), we could select $u's$ favorite features according to equation (2), let the indices of the n_f largest feature scores in the cube $\hat{T}$ be $INDF_{ui} = \{indf_{ui1}, indf_{ui2}, ...indf_{uin_f}\}$. Then a user's interests for an item could be derived as follows:

$$R^{feature}_{ui} = \frac{\sum_{f\in INDF_{ui}} \hat{T}_{uif} \cdot s_{if}}{\pi n_f} \tag{6}$$

where π is a rescaling parameter.Note that when implementing, we normalize every feature score to be a value in (0,1).

Algorithm 1 LPRRM-CF

Input: $A, m, n, p, K, \lambda, \lambda_\Theta, O$
Output: R^U, R^I, R^{UF}, R^{IF}

initialize
$R^U \leftarrow \mathbb{R}_+^{m\times K}$, $R^I \leftarrow \mathbb{R}_+^{n\times K}$, $R^{UF} \leftarrow \mathbb{R}_+^{p\times K}$, $R^{IF} \leftarrow \mathbb{R}_+^{p\times K}$;
iter=0
repeat
$iter \leftarrow iter + 1$;
draw (u, i, f_A, f_B) from D_S
$\hat{T}_{uif_Af_B} \leftarrow \hat{T}_{uif_A} - \hat{T}_{uif_B}$
$\delta_{ranking} \leftarrow 1 - \sigma(\hat{T}_{uif_Af_B})$
$\delta_{rating} \leftarrow 2l_{ui}(A_{ui} - \sum_{k=0}^{K-1} \hat{R}^U_{uk}\hat{R}^I_{ik})$
for $k \in 0, 1, ...K-1$ **do**

$R^U_{uk} \leftarrow R^U_{uk} + learning\_rate * (\delta_{rating} \cdot R^I_{ik} + \lambda \cdot \delta_{ranking} \cdot (R^{UF}_{f_Ak} - R^{UF}_{f_Bk}) - \lambda_{R^U} \cdot R^U_{uk})$

$R^I_{ik} \leftarrow R^I_{ik} + learning\_rate * (\delta_{rating} \cdot R^U_{uk} + \lambda \cdot \delta_{ranking} \cdot (R^{IF}_{f_Ak} - R^{IF}_{f_Bk}) - \lambda_{R^I} \cdot R^I_{ik})$

$R^{UF}_{f_Ak} \leftarrow R^{UF}_{f_Ak} + learning\_rate * (\lambda \cdot \delta_{ranking} \cdot R^U_{uk} - \lambda_{R^{UF}} \cdot R^{UF}_{f_Ak})$

$R^{UF}_{f_Bk} \leftarrow R^{UF}_{f_Bk} + learning\_rate * (\lambda \cdot \delta_{ranking} \cdot (-R^U_{uk}) - \lambda_{R^{UF}} \cdot R^{UF}_{f_Bk})$

$R^{IF}_{f_Ak} \leftarrow R^{IF}_{f_Ak} + learning\_rate * (\lambda \cdot \delta_{ranking} \cdot R^I_{ik} - \lambda_{R^{IF}} \cdot R^{IF}_{f_Ak})$

$R^{IF}_{f_Bk} \leftarrow R^{IF}_{f_Bk} + learning\_rate * (\lambda \cdot \delta_{ranking} \cdot (-R^I_{ik}) - \lambda_{R^{IF}} \cdot R^{IF}_{f_Bk})$

end for
until Converge or iter > max_iter
return R^U, R^I, R^{UF}, R^{IF}

At last, we set the final ranking score (RS) of user u for item i as follows:

$$RS_{ui} = \alpha \cdot R^{feature}_{ui} + (1-\alpha) R^{rating}_{ui} \tag{7}$$

In our datasets, as the rating range is [1,5] and thus the maximum value of $\hat{A}_{ui}$ is 5, we set $\pi = 0.2$ to map the first part of (7) that falls in the range of (0,1) into a comparable value with $\hat{A}_{ui}$. $0 < \alpha < 1$ is a scale parameter that controls the trade off between feature-based score and direct user-item ratings. The recommendation list for user u can be constructed by ranking the items in descending order of RS_{ui}.

3.4.3 Transfer to Category-level LPRRM-CF

The LRPPM-CF model above is conducted on product-level, which is able to capture the various interested features of users for different products especially when the dataset is sufficiently dense. However, in many practical scenarios, user interactions with a single product is very sparse (usually only a single piece of review), and to make our model more robust and adaptable in such application scenarios, we further extend our LRPPM-CF approach to category-level modeling. We define the set of categories as $C = \{c_1, c_2, ...c_{NC}\}$, where NC is the number of categories and the set of user-category-feature triplets as follows:

$$O^* := \{(u, c, f) | u \in U, c \in C, f \in F, User\ u\ mentioned\ feature\ f\ when\ he(she)\ reviewed\ products\ in\ category\ c.\} \tag{8}$$

Similar to product-level LRPPM-CF, the scoring function $\hat{T}^*_{ucf}$ is computed as follows when factorizing this cube:

$$\hat{T}^*_{ucf} = \sum_{k=0}^{K-1} R^U_{uk} \cdot R^{UF}_{fk} + \sum_{k=0}^{K-1} R^C_{ck} \cdot R^{CF}_{fk} + \sum_{k=0}^{K-1} R^U_{uk} \cdot R^C_{ck} \tag{9}$$

where $R^U \in \mathbb{R}_+^{|U|\times K}$, $R^C \in \mathbb{R}_+^{|C|\times K}$, $\{R^{UF} \in \mathbb{R}_+^{|F|\times K}, R^{CF} \in \mathbb{R}_+^{|F|\times K}\}$ are the representation matrices of users, categories and features, respectively, and K is the dimension of the representations.

Suppose $R^I \in \mathbb{R}_+^{|I|\times K}$ is the representation of items. Let the products in category c be $CI_c = \{i_{c1}, i_{c2}, ...i_{cn_c}\}$. We set $R^C_{c\cdot} = \frac{1}{n_c}\sum_{i\in CI_c} R^I_{i\cdot}$ to capture the relationship between products and their categories. The integrated optimization task thus is:

$$\min_\Theta \sum_{u\in U}\sum_{i\in I}(A_{ui} - (R^U_{u\cdot})^T \cdot R^I_{i\cdot})^2 - \lambda \sum_{u\in U}\sum_{c\in C}\sum_{f_A\in F^+_{uc}}\sum_{f_B\in F^-_{uc}} ln\sigma(\hat{T}^*_{ucf_A} - \hat{T}^*_{ucf_B}) + \lambda_\Theta \|\Theta\|^2_F \tag{10}$$

where λ is a tuning parameter, $\Theta = \{R^U, R^I, R^{UF}, R^{IF}\}$, λ_Θ is the regularization constant. F^+_{uc} is the set of features that are mentioned by user u for category c's items, and F^-_{uc} is the set of features that are not mentioned, namely, $F^+_{uc} = \{f \mid (u, c, f) \in O^*\}$ and $F^-_{uc} = \{f | (u, c, f) \notin O^*\}$.We adopt $u's$ favourite features for $i's$ category c as the final feature list for item i. The recommendation list could be generated according equation (7), where we replace $\hat{T}_{uif}$ by $\hat{T}^*_{ucf}$.

The product-level and category-level LRPPM-CF approaches are suitable for different scenarios regarding the characteristic of the tasks (e.g., data sparsity), and they play a complementary role to each other.

3.4.4 Relations between Product-level LPRRM-CF, Category-level LPRRM-CF, and EFM

LRPPM-CF vs EFM The relationship between these two models lies in that they both attempt to model the pairwise interactions among users, items, and features. The difference is that when generating users' favourite features, a rating-based 2D-matrix factorization technique is used in EFM model to capture the static produce-irrelevant feature preferences of users. However, in our LRPPM-CF model, we designed a ranking-based tensor-matrix factorization approach to discriminate users' different interests on feature over different products.

Product- vs Category-level LRPPM-CF Obviously, if every category contains only one product, Category-level LRPPM-CF would reduce to Product-level LRPPM-CF. When the categories contain more than one products, we can bring equation (9) into (10), and rewrite the Category-level LRPPM-

CF optimization objective function as:

$$\min_{\Theta} \sum_{u\in U}\sum_{i\in I}(A_{ui} - (R^U_{u\cdot})^T \cdot R^I_{i\cdot})^2 - \lambda \sum_{u\in U}\sum_{c\in C}\sum_{f_A\in F^+_{uc}}\sum_{f_B\in F^-_{uc}} ln\sigma(\sum_{k=0}^{K-1} R^U_{uk}\cdot R^{UF}_{f_Ak} + \sum_{k=0}^{K-1} R^C_{ck}\cdot R^{CF}_{f_Ak} - \sum_{k=0}^{K-1} R^U_{uk}\cdot R^{UF}_{f_Bk} - \sum_{k=0}^{K-1} R^C_{ck}\cdot R^{CF}_{f_Bk}) + \lambda_\Theta \|\Theta\|^2_F \tag{11}$$

$$Let \;\; H = \sum_{c\in C} ln\sigma(\sum_{k=0}^{K-1} R^U_{uk}\cdot R^{UF}_{f_Ak} + \sum_{k=0}^{K-1} R^C_{ck}\cdot R^{CF}_{f_Ak} - \sum_{k=0}^{K-1} R^U_{uk}\cdot R^{UF}_{f_Bk} - \sum_{k=0}^{K-1} R^C_{ck}\cdot R^{CF}_{f_Bk}) \tag{12}$$

and bring $R^C_{c\cdot} = \frac{1}{n_c}\sum_{i\in CI_c} R^I_{i\cdot}$ into (12) we have:

$$\begin{aligned} H &= \sum_{c\in C} ln\sigma(\sum_{k=0}^{K-1} R^U_{uk}\cdot R^{UF}_{f_Ak} + \sum_{k=0}^{K-1}\frac{1}{n_c}\sum_{i\in CI_c} R^I_{ik}\cdot R^{CF}_{f_Ak} - \sum_{k=0}^{K-1} R^U_{uk}\cdot R^{UF}_{f_Bk} - \sum_{k=0}^{K-1}\frac{1}{n_c}\sum_{i\in CI_c} R^I_{ik}\cdot R^{CF}_{f_Bk}) \\ &= \sum_{c\in C} ln\sigma(\frac{1}{n_c}\sum_{i\in CI_c}\sum_{k=0}^{K-1}(R^U_{uk}\cdot R^{UF}_{f_Ak} + R^I_{ik}\cdot R^{CF}_{f_Ak}) - \frac{1}{n_c}\sum_{i\in CI_c}\sum_{k=0}^{K-1}(R^U_{uk}\cdot R^{UF}_{f_Bk} + R^I_{ik}\cdot R^{CF}_{f_Bk})) \end{aligned} \tag{13}$$

we set $R^{CF} = R^{IF}$, and according to (3)(4), we have:

$$\begin{aligned} H &= \sum_{c\in C} ln\sigma(\frac{1}{n_c}\sum_{i\in CI_c}\sum_{k=0}^{K-1}(R^U_{uk}\cdot R^{UF}_{f_Ak} + R^I_{ik}\cdot R^{IF}_{f_Ak} + R^U_{uk}\cdot R^I_{ik}) \\ &- \frac{1}{n_c}\sum_{i\in CI_c}\sum_{k=0}^{K-1}(R^U_{uk}\cdot R^{UF}_{f_Bk} + R^I_{ik}\cdot R^{IF}_{f_Bk} + R^U_{uk}\cdot R^I_{ik})) \\ &= \sum_{c\in C} ln\sigma(\frac{1}{n_c}\sum_{i\in CI_c}(\hat{T}_{uif_A}) - \frac{1}{n_c}\sum_{i\in CI_c}(\hat{T}_{uif_B})) \\ &= \sum_{c\in C} ln\sigma(\frac{1}{n_c}\sum_{i\in CI_c}(\hat{T}_{uif_Af_B})) \end{aligned} \tag{14}$$

then the Category-level LRPPM-CF optimization objective function can be rewriten as:

$$\min_{\Theta}\sum_{u\in U}\sum_{i\in I}(A_{ui} - (R^U_{u\cdot})^T\cdot R^I_{i\cdot})^2 - \lambda\sum_{u\in U}\sum_{f_A\in F^+_{uc}}\sum_{f_B\in F^-_{uc}}\sum_{c\in C} ln\sigma(\frac{1}{n_c}\sum_{i\in CI_c}(\hat{T}_{uif_Af_B})) + \lambda_\Theta\|\Theta\|^2_F \tag{15}$$

Comparing with Product-level LRPPM-CF optimization goal shown in equation (5), we see that Category-level LRPPM-CF uses all the features extracted from a category as the positive observations for its products.

4. RATING-BASED TENSOR FACTORIZATION

In our LRPPM-CF framework described above, we model the user preferences towards different features with a ranking-based approach by modeling the implicit feedbacks directly, because we believe the estimation of user preferences on features is inherently a learning to rank task, which attempts to capture the preference of a feature over another one on products- or category-levels. To verify our assumption, we also study a rating-based tensor factorization model that treats the implicit feedbacks as explicit ratings.

We define the rating user u gave to item i is defined as a_{ui}, the features extracted from user u's textual review for item i are $F_{ui} = \{f_{ui1}, f_{ui1}, ...f_{uiK_{ui}}\}$, where K_{ui} is the number of extracted features.

Similar to LRPPM-CF, we consider user-item-feature relationship as an interaction cube T^s. However, instead of modeling user's implicit behavior, the element T^s_{uik} in T^s is assigned as a score that reflects the degree of user u's interest towards item i on feature f. In this method, we assign T^s_{uik} as $\frac{a_{ui}}{K_{ui}}$ if $f_k \in F_{ui}$ with a positive sentiment polarity, $-\frac{a_{ui}}{K_{ui}}$ if $f_k \in F_{ui}$ with a negative sentiment polarity, and 0 when $f_k \notin F_{ui}$. To make fair comparison, the fitting function adopted here is the same as the method used in LRPPM-CF(equation (2)).

We adopt the least square minimization method to fit these ratings, suppose O^s is defined as:

$$O^s := \{(u,i,f)|u\in U, i\in I, f\in F, User\ u\ mentioned\ feature\ f\ in\ his/her\ review\ on\ item\ i.\} \tag{16}$$

then the optimization objective function is:

$$\min_{\Theta}\sum_{(u,i,f)\in O^s}(T^s_{uif} - \hat{T}^s_{uif})^2 + \lambda_\Theta\|\Theta\|^2_F \tag{17}$$

where $\Theta = \{R^U, R^I, R^{UF}, R^{IF}\}$ are the model parameters.

We adapt stochastic gradient descent algorithm to learn the parameters. When making recommendation for user u, we first estimate the score u would give to the items, and then select the items with top-N highest scores to generate the recommendation list, where the score $\hat{a}_{ui}$ is computed by:

$$\hat{a}_{ui} = \sum_{k=1}^{|F|}\hat{T}^s_{uik} \tag{18}$$

Although this method tends to capture various user interests in a more compact way, it does not perform better than the LRPPM-CF approach in our experiments, which will be introduced in the following sections. The underlying reason can be that: (1) It may introduce too much noise when directly assigning T^s_{uik} as $\frac{a_{ui}}{K_{ui}}$; (2) Different features may account for different weights when deriving the rating estimations $\hat{a}_{ui}$; (3) Converting the user implicit feedback into explicit ratings may be bias and limited, which could be a shortcoming of the EFM approach, while our LRPPM-CF approach can model such implicit feedbacks directly in a learning to rank framework.

5. EXPERIMENTS

In this section, we conduct extensive experiments to evaluate our LRPPM-CF framework. We focus on the following research questions:

(1) What is the performance of our LRPPM-CF model in

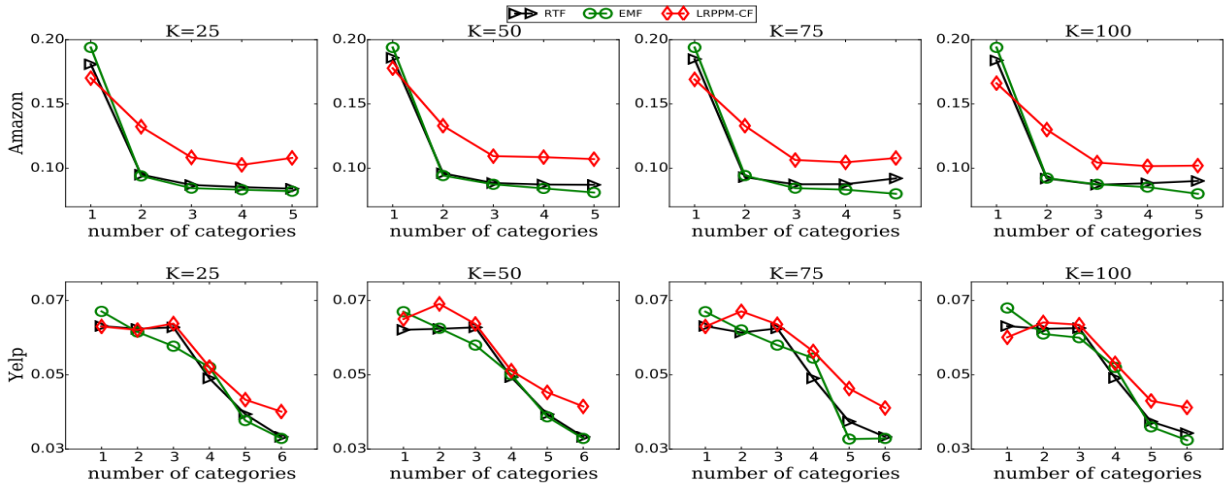

Figure 3: Performance comparison on F_1-measure of LRPPM-CF, EFM and RTF over two datasets. The number of total factors increase from 25 to 100.

Table 2: Statistics of the Amazon and Yelp datasets

	#Users	#Items	#Reviews	$\frac{\#Rviews}{\#Users}$
Amazon	965	13005	38430	39.82
Yelp	859	8750	51965	60.54

capturing various features for different products/categories. (2) What is the performance of our LRPPM-CF model in the task of item recommendation.

We begin by introducing the experimental setup, and then report and analyze the experimental results to attempt to answer the research questions.

5.1 Experimental Setup

We choose the Amazon[2][7, 8] and Yelp[3] datasets for experiments. The former dataset contains user transaction records and textual reviews from Amazon spanning May 1996 - July 2014. The latter dataset consists of user reviews on various businesses. For evaluating the properties of our models, in both of these datasets, we select the users who purchased items in at least 3 categories, and choose the items with 5 or more reviews. We randomly holdout 30% reviewed items from each user to construct the testing set, and the others are used for training. The statistics of our datasets are shown in Table 2.

5.2 Performance in Capturing Various Features

In this subsection, we investigate the performance of LRPPM-CF in capturing different features for various products/categories. We compare the predicted features with the truely mentioned features for every user-item pair in the test dataset. Two methods are selected as our baselines, which are Rating-based Tensor Factorization (RTF) proposed in section 4 and Explicit Factor Model (EFM) [18], which is the state-of-the-art approach for user preference prediction and recommendation based on explicit features from reviews. When implementing RTF, we select 5 features according to their scores in the fitted cube. When implementing EFM,

[2]http://jmcauley.ucsd.edu/data/amazon/
[3]http://www.yelp.com/dataset_challenge

Table 3: Statistics of the datasets containing various number of categories

	#Users	#Items	#Reviews	#Categories
Amazon_C1	749	2697	3568	1
Amazon_C2	949	5584	13004	2
Amazon_C3	965	7641	22412	3
Amazon_C4	965	8867	25242	4
Amazon_C5	965	13005	38430	5
Yelp_C1	856	1902	14274	1
Yelp_C2	858	2204	15514	2
Yelp_C3	858	2245	15607	3
Yelp_C4	859	2832	17252	4
Yelp_C5	859	8265	46059	5
Yelp_C6	859	8750	51965	6

in order to achieve the best performance, we tune the ratio between explicit and latent factors by fixing the total number of factors as 25, 50, 75, 100, respectively, and the weighting scalar is set as 0.85 as reported in [18]. For every user, we select 5 features for all the products according to their scores in the fitted user-feature attention matrix. When implementing LRPPM-CF, we set the dimension K as 25, 50, 75, 100, respectively, so as to ensure equal model complexity, and also select the top-5 features utilizing the method proposed in section 3.2 as the predicted results.

The hyper-parameters of these methods are selected by conducting grid search and 5-fold cross-validation. We compare these models on the metric of F_1-score. In order to validate the performance under different number of categories, we construct 5 datasets for Amazon and 6 datasets for Yelp, where the n-th dataset contains products from n categories, respectively. The statistics of the datasets are shown in Table 3.

5.2.1 *Comparison between LRPPM-CF and Baselines*

The overall experimental results of comparing product-level LRPPM-CF with RTF and EFM on the task of predicting the features for a given user-item pair are shown in Figure 3. We see from the results that:

(1) When the datasets contain only one category, the EFM

method performs better than RTF and LRPPM-CF across all the choices on the number of factors. The reason may be that people tend to care about a similar and limited set of features when purchasing products in the same category, thus the RTF and LRPPM-CF approach with 25 or more factors to capture the features could face severe overfitting problems.

(2) When the datasets contain more than one categories, LRPPM-CF performs better than RTF and EFM(enhance the performance by about 17% and 24% on the Amazon dataset containing 5 categories and Yelp dataset containing 6 categories respectively), and the improvements are significantly at 0.01 level. This result is in expectation because the RTF method forcefully converts user's implicit feedbacks to ratings, which could bias the results, and that for a given user, EFM does not consider his/her different interests on features for different products. Instead, our LRPPM-CF approach models the implicit interaction between users, items, and features simultaneously and directly.

(3) With the increasing on the number of categories, LRPPM-CF maintains a satisfactory prediction accuracy compared with EFM, whose performance decrease significantly with the increase on categories.

5.2.2 Product- and Category-level LRPPM-CF

In this section, we further investigate the performance of LRPPM-CF by applying this model on two levels of granularities, i.e., product-level and category-level. Similar to the previous experiment, we generate 5 features for each user on each category when implementing category-level LRPPM-CF. The parameters of category-level LRPPM-CF are also selected by conducting grid search in their corresponding ranges. The models are evaluated on Amazon datasets containing 5 categories(i.e., the dataset Amazon_C5) and Yelp datasets containing 6 categories (i.e., the dataset Yelp_C6), where the statistics of the datasets can be seen in the 5th and the last line of Table 3.

Table 4: Performance comparison of Product-level and Category-level LRPPM-CF. The dimensionality is increase from 25 to 100.

dataset	dimension	25	50	75	100
Yelp	Category	0.043	0.0431	0.044	0.0432
	Product	0.040	0.0415	0.0411	0.0413
Amazon	Category	0.11	0.112	0.111	0.104
	Product	0.107	0.106	0.107	0.100

From the results shown in Table 4, we notice that category-level LRPPM-CF performs better than product-level LRPPM-CF in all dimensions, also at a significant level of 0.01. The underlying reason can be that: (1) As the implicit feedback in the user-item-feature cube on product-level is very sparse, this model fails to capture sufficiently many positive features to generate user preference pairs, which could lead to lower prediction accuracy. (2) Because all the products in a category are considered as a whole when conducting category-level LRPPM-CF, much more positive features are collected that help to alleviate the problem of data sparsity.

5.3 Performance in Item Recommendation

In this section, we investigate the LRPPM-CF model in the task of item recommendation. We analyze the model to find what and how the performance is affected by some specific parameters, and which parameters are of key importance to the users in the task.

We use the Amazon containing 5 categories and Yelp containing 6 categories for our experiments (Datasets Amazon_C5 in the 5th and Yelp_C6 in the last line of Table 3). Our product-level LRPPM-CF model is compared with the following baseline methods:

MostPopular: A static method that recommends different users with the same products according to their popularity.

PMF: The Probabilistic Matrix Factorization method proposed in [9], which is a frequently used stat-of-the-art approach for rating-based optimization and prediction. The number of latent factors is set as 50 for Yelp dataset and 20 for Amazon dataset based on cross-validation.

EFM: The state-of-art algorithm [18] in terms of making recommendations based on phrase-level sentiment analysis on textual reviews. In order to achieve the best performance, we conduct grid search on the number of each user's most cared features k in the range of [5,100], as well as the weighing scalar α in the range of (0,1]. When determining the number of explicit and latent factors, we fix the total number as 50 and tune the ratio between them to find an optimal ratio, and then we increase the total number from 10 to 100 by fixing this ratio. The parameters are finally set as $k = 5, \alpha = 0.8$, total number of factors = 20, ratio of explicit factors = 60% in the Amazon dataset, and $k = 15, \alpha = 0.85$, total number of factors = 50, ratio of explicit factors = 40% in the Yelp dataset.

RTF: The compact Rating-based Tensor Factorization method proposed in section 4. We still set the number of factors as 50 for Yelp dataset and 20 for Amazon dataset when implementing this method.

We conduct top-5 recommendation on both of the Amazon and Yelp datasets. The dimension K in our LRPPM-CF model is set as 50 for Yelp datasets and 20 for Amazon datasets to ensure equal model complexity. Grid search and five-fold cross-validation are used for parameter tuning and performance evaluation. We adopt F_1-score and Normalized Discounted Cumulative Gain (NDCG) as the measures to evaluate the performance of different models.

The experimental results are shown in Table 5. From the results we can see that: (1) Among the baseline methods PMF performs better than MostPopular because PMF takes the global information into consideration and thus provides personalized recommendations. By utilizing user review information and sentiment analysis, EFM and RTF performs better than other baseline methods, while little difference is observed between EFM and RTF. (2) Our product-level LRPPM-CF approach gains superior performance against all the baseline methods, and this is actually as expected because it captures user interests over features more precisely on a per-product level, and thus makes more accurate item recommendations.

5.3.1 Influence of the number of most cared features n_f

We first fix the number of most cared features $n_f = 5$ and find that the optimal value for λ is 0.3. We then fix $\lambda = 0.3$ throughout the following experiments to focus on the effect of the key parameter n_f.

Our goal is to investigate the performance of LRPPM-CF when n_f increases from 5 to the maximum possible value (100 for Amazon, and 80 for Yelp), larger n_f would lead to significant bias and the results on Yelp are shown in Figure 4. Based on performance of F_1-score we see that our LRPPM-CF model outperforms all the baseline methods when n_f

Table 5: Comparison between LRPPM-CF and baseline methods on the Amazon and Yelp dataset. The starred values indicate significant improvements against the best baseline on 0.01 and 0.005 level, respectively.

dataset	metric	MP	PMF	EFM	RFT	LRPPM-CF
Yelp	F1-score@5	0.0183	0.025	0.029	0.028	**0.033***
	NDCG@5	0.0111	0.012	0.0131	0.0144	**0.0186**
Amazon	F1-score@5	0.0135	0.0140	0.0142	0.0139	**0.0152***
	NDCG@5	0.0119	0.0120	0.0124	0.0132	**0.0138**

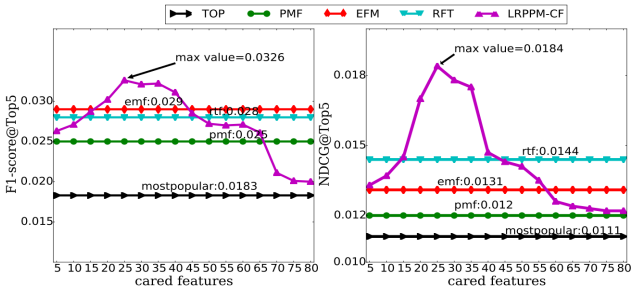

Figure 4: Influence of n_f on the Yelp dataset

ranges from 20 to 40. The best performance of LRPPM-CF with $n_f = 25$ is 11% better than EFM, which performs best among the baseline methods. From the perspective of LRPPM-CF itself, the performance continues to rise as n_f increases from 5 to 25. However, when n_f falls in the range of [25,40], the performance tends to be stable, and when n_f exceeds 40, the performance begins to drop rapidly. From the performance of NDCG@5 we see that LRPPM-CF performs better than all the baseline methods when n_f is in [15,40], and achieves its best performance when n_f arrives at 25. Similar results are observed on Amazon dataset, which are shown in Figure 5.

These observations confirm our hypothesis in section 3.3.3 that when making decisions, users usually care about several key features, and taking too many features into consideration could introduce noise into the models, which is consistent with the observations in EFM [18].

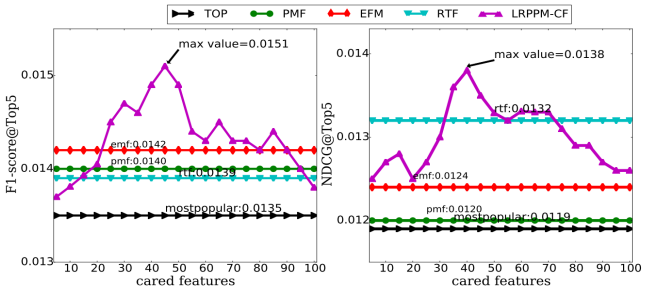

Figure 5: Influence of n_f on the Amazon dataset

5.3.2 *Influence of Weighting Parameter λ*

In the following set of experiments, we set $n_f = 25$ which achieves the best performance as reported above.We study how the performance (F_1-score) changes as λ increases from 0.05 to 1.2(larger λ would lead to significant bias), and the results on Yelp dataset is shown in Figure 6. We can see that:

(1) LRPPM-CF outperforms the other models when λ is in the range of [0.2, 0.5]. It confirms that our integrated model does enhance the recommendation quality as compared with the simple model of leveraging only the ratings, i.e., when $\lambda = 0$.

(2) The performance of LRPPM-CF continues to rise until λ reaches around 0.3, then after hovering approximately stable in the range of [0.3, 0.4], it begins to drop rapidly with the increase of λ. This observation indicates that although the user reviewing behavior is important in boosting the performance of recommendation, user rating behavior still helps to make accurate predictions. Besides, similar results can be seen on the Amazon dataset, which are shown in Figure 6.

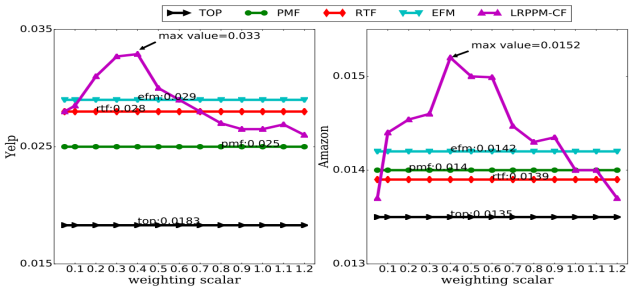

Figure 6: Influence of λ on Amazon and Yelp datasets

5.3.3 *Further Analysis of Category-level LRPPM-CF*

Although the results in section 5.2 show that category-level LRPPM-CF performs well in the task of capturing various features, we still want to explore whether LRPPM-CF is adequately qualified in the task of item recommendation.

In this set of experiments, we compare the performance between Category- and Product-level LRPPM-CF on the Yelp dataset. The key parameters in both of these methods are n_f and λ. For fair comparison, we tune one of these parameters by fixing the other to observe the differences between the models. For implementing, we first fix λ as 0.4, and observe the change in performance with different n_f, and then investigate the performance with various λ when $n_f = 25$. The dimensions of category- and product-level LRPPM-CF are set as 50. We still adopt F_1-score@5 for performance evaluation.

The experimental results are shown in Figure 7. It shows that category-level LRPPM-CF performs better than product-level in most cases, which indicates that a higher level LRPPM-CF may be more practically effective in personalized recommendation. The underlying reason could be that ranking and summarizing the features for users on per-category level help to alleviate the problem of data sparsity.

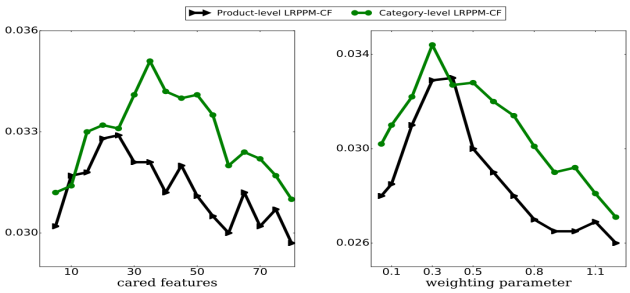

Figure 7: Performance comparison on F_1-score between Category- and Product-level LRPPM-CF in the task of item recommendation.

6. CONCLUSIONS

In this paper, we propose a learning to rank framework to model user preference on explicit features extracted from textual reviews, which is able to capture the implicit feedbacks in a direct way that promotes the accuracy. We thus propose a tensor-matrix factorization technique to learn user interests over features on both product- and category-levels. Furthermore, we integrate this technique with traditional collaborative filtering methods, and propose a unified hybrid framework for both more accurate product/category-level feature ranking and better performance on personalized recommendations, which are verified thorough extensive experiments on both Amazon and Yelp datasets.

This is a first step towards our goal in explainable recommendation on high-level feature spaces and heterogeneous cross-domain recommendation, and there is much room for further improvements. In the future, we will focus on the following research directions. Because our LRPPM approach is a framework rather than simply an algorithm, we are able to integrate more personalization models into this framework according to specific application scenarios, which may bring us more inspiring insights on the characters and performance of traditional methods from the angle of explainable recommendation. We can also adapt other machine learning methods beyond tensor factorization to capture the user-item-feature interactions, such as probabilistic graphic models and topical modeling.

7. REFERENCES

[1] N. Jakob, S. H. Weber, M. C. Müller, and I. Gurevych. Beyond the stars: exploiting free-text user reviews to improve the accuracy of movie recommendations. In *Proceedings of the 1st international CIKM workshop on Topic-sentiment analysis for mass opinion*, pages 57–64. ACM, 2009.

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

[3] C. W. Leung, S. C. Chan, and F.-l. Chung. Integrating collaborative filtering and sentiment analysis: A rating inference approach. In *Proceedings of the ECAI 2006 workshop on recommender systems*, pages 62–66. Citeseer, 2006.

[4] Y. Lu, M. Castellanos, U. Dayal, and C. Zhai. Automatic construction of a context-aware sentiment lexicon: an optimization approach. In *WWW*, pages 347–356. ACM, 2011.

[5] B. M. Marlin. Modeling user rating profiles for collaborative filtering. In *Advances in neural information processing systems*, page None, 2003.

[6] J. McAuley and J. Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 165–172. ACM, 2013.

[7] J. McAuley, R. Pandey, and J. Leskovec. Inferring networks of substitutable and complementary products. In *SIGKDD*, pages 785–794. ACM, 2015.

[8] J. McAuley, C. Targett, Q. Shi, and A. van den Hengel. Image-based recommendations on styles and substitutes. In *SIGIR*, pages 43–52. ACM, 2015.

[9] A. Mnih and R. Salakhutdinov. Probabilistic matrix factorization. In *Advances in neural information processing systems*, pages 1257–1264, 2007.

[10] N. Pappas and A. Popescu-Belis. Sentiment analysis of user comments for one-class collaborative filtering over ted talks. In *SIGIR*, pages 773–776. ACM, 2013.

[11] D. Y. Pavlov and D. M. Pennock. A maximum entropy approach to collaborative filtering in dynamic, sparse, high-dimensional domains. In *Advances in neural information processing systems*, pages 1441–1448, 2002.

[12] Š. Pero and T. Horváth. Opinion-driven matrix factorization for rating prediction. In *User Modeling, Adaptation, and Personalization*, pages 1–13. Springer, 2013.

[13] S. Rendle, L. Balby Marinho, A. Nanopoulos, and L. Schmidt-Thieme. Learning optimal ranking with tensor factorization for tag recommendation. In *SIGKDD*, pages 727–736. ACM, 2009.

[14] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 452–461. AUAI Press, 2009.

[15] S. Rendle and L. Schmidt-Thieme. Pairwise interaction tensor factorization for personalized tag recommendation. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 81–90. ACM, 2010.

[16] P. Symeonidis, A. Nanopoulos, and Y. Manolopoulos. Tag recommendations based on tensor dimensionality reduction. In *Recsys*, pages 43–50. ACM, 2008.

[17] Y. Wu and M. Ester. Flame: A probabilistic model combining aspect based opinion mining and collaborative filtering. In *WSDM*, pages 199–208. ACM, 2015.

[18] 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 *SIGIR*, pages 83–92. ACM, 2014.

[19] Y. Zhang, H. Zhang, M. Zhang, Y. Liu, and S. Ma. Do users rate or review?: boost phrase-level sentiment labeling with review-level sentiment classification. In *SIGIR*, pages 1027–1030. ACM, 2014.

[20] Y. Zhang, M. Zhang, Y. Liu, S. Ma, and S. Feng. Localized matrix factorization for recommendation based on matrix block diagonal forms. In *WWW*, pages 1511–1520. International World Wide Web Conferences Steering Committee, 2013.