Explicit Factor Models for Explainable Recommendation based on Phrase-level Sentiment Analysis

Yongfeng Zhang[†],Guokun Lai[†],Min Zhang[†],Yi Zhang[‡],Yiqun Liu[†],Shaoping Ma[†]
[†]State Key Laboratory of Intelligent Technology and Systems
[†]Department of Computer Science & Technology, Tsinghua University, Beijing, 100084, China
[‡]School of Engineering, University of California, Santa Cruz, CA 95060, USA
{zhangyf07,laiguokun}@gmail.com,{z-m,yiqunliu,msp}@tsinghua.edu.cn,yiz@soe.ucsc.edu

ABSTRACT

Collaborative Filtering(CF)-based recommendation algorithms, such as Latent Factor Models (LFM), work well in terms of prediction accuracy. However, the latent features make it difficulty to explain the recommendation results to the users.

Fortunately, with the continuous growth of online user reviews, the information available for training a recommender system is no longer limited to just numerical star ratings or user/item features. By extracting explicit user opinions about various aspects of a product from the reviews, it is possible to learn more details about what aspects a user cares, which further sheds light on the possibility to make explainable recommendations.

In this work, we propose the Explicit Factor Model (EFM) to generate explainable recommendations, meanwhile keep a high prediction accuracy. We first extract explicit product features (i.e. aspects) and user opinions by phrase-level sentiment analysis on user reviews, then generate both recommendations and disrecommendations according to the specific product features to the user's interests and the hidden features learned. Besides, intuitional feature-level explanations about why an item is or is not recommended are generated from the model. Offline experimental results on several real-world datasets demonstrate the advantages of our framework over competitive baseline algorithms on both rating prediction and top-K recommendation tasks. Online experiments show that the detailed explanations make the recommendations and disrecommendations more influential on user's purchasing behavior.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Filtering; I.2.7 [**Artificial Intelligence**]: Natural Language Processing

Keywords

Recommender Systems; Sentiment Analysis; Collaborative Filtering; Recommendation Explanation

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
SIGIR'14, July 6–11, 2014, Gold Coast, Queensland, Australia.
Copyright 2014 ACM 978-1-4503-2257-7/14/07 ...$15.00.
http://dx.doi.org/10.1145/2600428.2609579.

1. INTRODUCTION

In the last few years, researchers have found or argued that explanations in recommendation systems could be very beneficial. By explaining how the system works and/or why a product is recommended, the system becomes more transparent and has the potential to allow users to tell when the system is wrong (scrutability), increase users' confidence or trust in the system, help users make better (effectiveness) and faster (efficiency) decisions, convince users to try or buy (persuasiveness), or increase the ease of the user enjoyment (satisfaction). A variety of techniques have been proposed to generate explanations, mainly for content based recommendation algorithms, neighbor based algorithms, or simple statistics analysis based algorithms.

Meanwhile, Latent Factor Models (LFM) such as Matrix Factorization (MF) [14] techniques have gained much attention from the research community and industry due to their good prediction accuracy on some benchmark datasets. However, recommender systems based on these algorithms encounter some important problems in practical applications. First, it is difficult to know how users compose their judgement of the various attributes of an item into a single rating, which makes it difficult to make recommendations according to the specific needs of the users. Second, it is usually difficult to give intuitional explanations of why an item is recommended, and even more difficult to explain why an item is *not* recommended given other alternatives. Lack of explainability weakens the ability to persuade users and help users make better decisions in practical systems [39].

A dilemma practitioners often face is whether to choose an understandable/explainable simple algorithm while sacrificing prediction accuracy, or choose an accurate latent factorization modeling approach while sacrificing explainability. A major research question is: can we have a solution that is both highly accurate and easily explainable?

Fortunately, the advance detailed sentiment analysis and the ever increasing popularity of online user textual reviews shed some light on this question. Most e-commerce and review service websites like Amazon and Yelp allow users to write free-text reviews along with a numerical star rating. The text reviews contain rich information about user sentiments, attitudes and preferences towards product features [19, 8, 9], which sheds light on new approaches for explainable recommendation. For example, from the review text "The *service* rendered from the seller is *excellent*, but the *battery life* is *short*", the entries (*service, excellent*, +1) and (*battery life, short*, −1) of the form (F, O, S) could be extracted by phrase-level sentiment analysis [19], where F is

for Feature word or phrase that reveals some product aspect, O is for Opinion word or phrase that the user chose to express the attitude towards the feature, and S is the Sentiment of the opinion word when commenting on the feature word, which could be positive or negative.

Different users might care about different product aspects. We found that users tend to comment on different features in textual reviews, *e.g.*, one would mostly care about the screen size of a mobile phone, while another might focus on its battery life, although they might even make the same star rating on the product. Extracting the explicit product features and the corresponding user opinions from reviews not only helps to understand the different preferences of users and make better recommendations, but also helps to know why and how a particular item is or is not recommended, thus to present intuitional explanations. In this way, we could not only recommend to users about which to buy, but also presenting disrecommendations by telling the users why they would better not buy.

Based on our preliminary analysis, we propose a new Explicit Factor Model (EFM) to achieve both high accuracy and explainability. Figure 1 illustrates the overview of the proposed solution with an example. First, phrase-level sentiment analysis over textual review corpus generates a sentiment lexicon, where each entry is an (F, O, S) triplet, and the feature words together serve as the explicit feature space. Then, user attentions and item qualities on these features are integrated into a unified factorization model (i.e. EFM), which are later used to generate personalized recommendations and explanations. In this example, the system identified that a user might care about *memory*, *earphone* and *service*, thus a product that performs especially well on these features would be a promising recommendation for him.

In the rest of this paper, we first review some related work (Section 2) and provide detailed expositions of our approach, including the algorithm for model learning (Section 3). Then we describe the offline experimental settings and results for verifying the performance of the proposed approach in terms of rating prediction and top-K recommendation (Section 4), as well as online experiments for testing the effect of intuitional explanations (Section 5). Finally, we conclude the work, discuss the limitations of the work and point out some of the future research directions in Section 6.

2. RELATED WORK

With the ability to take advantage of the wisdom of crowds, Collaborative Filtering (CF) [33] techniques have achieved great success in personalized recommender systems, especially in rating prediction tasks. Recently, Latent Factor Models (LFM) based on Matrix Factorization (MF) [14] techniques have gained great popularity as they usually outperform traditional methods and have achieved state-of-the-art performance in some benchmark datasets [33]. Various MF algorithms have been proposed in different problem settings, such as Singular Value Decomposition (SVD) [14, 32], Non-negative Matrix Factorization (NMF) [15], Max-Margin Matrix Factorization (MMMF) [29], Probabilistic Matrix Factorization (PMF) [30], and Localized Matrix Factorization (LMF) [45, 44]. They aim at learning latent factors from user-item rating matrices to make rating predictions, based on which to generate personalized recommendations. However, their *latent* characteristic makes it difficult to make recommendations in situations where we know a user cares

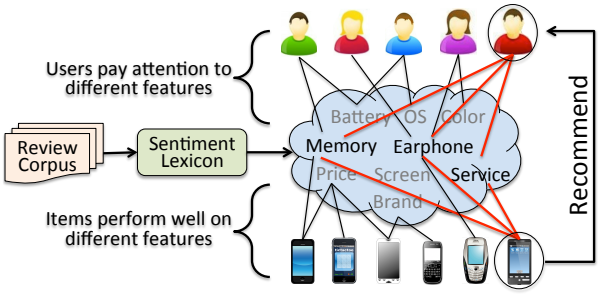


Figure 1: The product feature word and user opinion word pairs are extracted from user review corpus to construct the sentiment lexicon, and the feature word set further serves as the explicit feature space. An item would be recommended if it performs well on the features that a user cares.

about certain particular product features. Further more, it is also difficult to generate intuitional explanations for the recommendation results. Besides, the frequently used metrics such as RMSE and MAE do not necessarily have direct relationship with the performance in practical top-K recommendation scenarios [4].

It is important to notice that the increasingly growing amount of user generated textual reviews contain rich information about product features and user preferences, which can be extracted, summarized and structured by Sentiment Analysis (SA) [18, 24]. Sentiment analysis can be conducted on three different levels: review/document-level, sentence-level and phrase-level. Review- [25, 42] and sentence-level [40, 22] analysis attempt to classify the sentiment of a whole review or sentence to one of the predefined sentiment polarities, including *positive*, *negative* and sometimes *neutral*, while phrase-level analysis [41, 19, 6] attempts to extract explicit features of the products, and further analyze the sentiment orientations that users express on these features based on opinion words that the users use to express the attitude towards the features [11]. The core task in phrase-level sentiment analysis is the construction of Sentiment Lexicon [34, 17, 6, 19], where each entry is a (Feature,Opinion,Sentiment) triplet. This is generated by extracting feature-opinion word pairs and determining their sentiments from text reviews. The lexicon could be used in various sentiment related tasks. It is necessary to note that the sentiment lexicon could be contextual [41]. For example, the opinion word *high* has a positive sentiment when modifying the feature word *quality*, yet has a negative sentiment when accompanied by *noise*.

With the continuous growth of such information-rich sources, how to use textual reviews in recommender systems has received increasing attention over the last few years [1, 20, 21, 26, 36, 12, 13, 7, 27]. The most early approach constructs manually defined ontologies to structure the free-text reviews [1]. However, constructing ontologies is domain-dependent and time consuming, and it is also not nicely integrated with the wisdom of crowds as in CF. More recent work leverages review- or sentence-level sentiment analysis to boost the performance of rating prediction [12, 13, 7, 27, 26]. However, rating prediction accuracy is not necessarily related to the performance in practical applications [4]. Besides, the sentence- or review-level analysis does not offer rich finer-grained sentiment information, which can be better utilized in recommendation and explanation.

In an attempt to address the problems, some recent work conducted topic discovery from the reviews to estimate user preferences for personalized recommendations [20, 21, 36].

However, as a user could in fact be criticizing rather than appreciating the product on a mentioned topic, simple topic level analysis without more detailed natural language processing makes such approaches biased and limited. In contrast, we leverage phrase-level sentiment analysis to model user preferences and item performances on an explicit feature/aspect for more detailed user preference modeling, more accurate predicting, and more intuitional explanations.

Researchers have shown that providing appropriate explanations could improve user acceptance of the recommended items [10, 37], as well as benefit user experience in various other aspects, including system transparency, user trust, effectiveness, efficiency, satisfaction and scrutability [38, 3, 2]. However, the underlying recommendation algorithm may influence the types of explanations that can be constructed. In general, the computationally complex algorithms within various latent factor models make the explanations difficult to be generated automatically [38].

Many meticulously designed strategies have been investigated to tackle the problem, ranging from the simple 'people also viewed' explanations in e-commerce websites [38] to the more recent social friends or social tags based explanations [31, 39]. However, such explanations are either over simplification of the true reasons, or difficult to generate in non-social scenarios. Nevertheless, an important advantage of utilizing explicit features for recommendation is its ability to provide intuitional and reasonable explanations for both recommended and disrecommended items. Besides, the proposed model can also integrate social or statistical features, and thus generalizes the existing explanation solutions.

3. THE FRAMEWORK

This section describes the major components of the proposed framework (Figure 1). The major notations used in the rest of the paper are summarized in Table 1. $\mathcal{L}$ is generated by phrase-level sentiment analysis on user reviews. A is the numerical user-item rating matrix. The matrices X and Y are constructed by mapping the reviews of each user and item onto the lexicon $\mathcal{L}$. The following sub-sections describe more details about the framework.

3.1 Sentiment Lexicon Construction

In the first stage, the contextual sentiment lexicon $\mathcal{L}$ is constructed from textual user reviews based on the state-of-the-art optimization approach described in [19, 43]. This process mainly consists of three steps. First, the feature word set $\mathcal{F}$ are extracted from the text review corpus using grammatical and morphological analysis tools. Then, the opinion word set $\mathcal{O}$ is extracted and further paired up with the feature words where possible. This leads to the feature-opinion pairs (F, O). Finally, sentiment polarity labelling of these pairs are conducted based on an optimization framework, and each pair is assigned a sentiment value S, and this leads to the final entries (F, O, S) in $\mathcal{L}$.

Table 1: Table of notations in the framework.

$\mathcal{L}$	The contextual sentiment lexicon
(F,O,S)	A lexicon entry of $\mathcal{L}$, where $S \in [-1, 1]$
$\mathcal{F}$	The feature word set of $\mathcal{L}$, where $\lvert\mathcal{F}\rvert = p$
$\mathcal{O}$	The opinion word set of $\mathcal{L}$, where $\lvert\mathcal{O}\rvert = q$
$A \in \mathbb{R}_+^{m \times n}$	The user-item numerical rating matrix
m, n	The number of users and items, respectively
$X \in \mathbb{R}_+^{m \times p}$	The user-feature attention matrix
$Y \in \mathbb{R}_+^{n \times p}$	The item-feature quality matrix
N	The non-zeros in A, X, Y are in the range of $[1, N]$

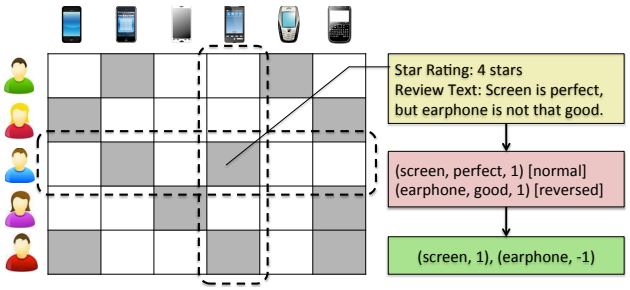


Figure 2: An example of user-item review matrix, where each shaded block is a review made by a user towards an item; the entries included in the review are extracted, and further transformed to feature scores while considering the negation words.

Since the lexicon construction procedure is not the focus of this paper, we refer the readers to the related literature such as [19, 43] for more details. The quality of the constructed lexicon will be reported later in the experimental section.

3.2 Feature-Opinion Pair Mapping

Given the sentiment lexicon $\mathcal{L}$ and a piece of text review, we generate a set of feature-sentiment (F, S') pairs to represent the review, where S' is the reviewer's sentiment on the particular product feature as expressed in the review text.

To illustrate the idea, we consider a toy example shown in Figure 2. Each shaded block is a user's review about an item, which includes a numerical rating and a piece of review text. Based on that, we need to identify which lexicon entries are mentioned by the user in the review text, and whether the sentiment polarity of the entry is reversed by negation words like 'not' or 'hardly'. Then we can generate a set of feature-sentiment pairs to represent a piece of review text. In this example, the set includes (*screen*, +1) and (*earphone*, -1).

Without lose of generality, we adopt the approaches in [19] and [35]. The algorithm analyzes the review text on clause level by parsing sentences into syntactic tree structures, with which a rule-based finite state matching machine approach is conducted for feature-opinion pair mapping, and the entries such as (*screen, perfect*, +1) and (*earphone, good*, +1) in $\mathcal{L}$ are matched in the review. Then, negation words detection is conducted to check whether the sentiment of each matched entry is reversed. For example, the sentiment of entry (*earphone, good*, +1) is reversed because of the presence of the negation word 'not'. Then we have:

$$S' = \begin{cases} S, \ if \ O \ is \ not \ reversed \ by \ negation \ words \\ -S, \ if \ O \ is \ reversed \ by \ negation \ words \end{cases} \quad (1)$$

3.3 User-Feature Attention Matrix

We assume that different users might care about different features, and they tend to comment more frequently on those features that he or she particularly cares. Thus we construct a user-feature attention matrix X, where each element measures to what an extent a user cares about the corresponding product feature/aspect.

Let $\mathcal{F} = \{F_1, F_2, \cdots, F_p\}$ be the set of explicit product features/aspects, and let $\mathcal{U} = \{u_1, u_2, \cdots, u_m\}$ denote the m users. To generate the matrix, we consider all the text reviews written by a user u_i, then extract all (F, S') entries in the collection. Suppose feature F_j is mentioned by user

u_i for t_{ij} times, we define each element in the user-feature attention matrix X as follows:

$$X_{ij} = \begin{cases} 0, \ \textit{if user } u_i \textit{ did not mention feature } F_j \\ 1 + (N-1)\Big(\frac{2}{1+e^{-t_{ij}}} - 1\Big), \ \textit{else} \end{cases} \tag{2}$$

The major goal of (2) is to rescale t_{ij} into the same range $[1, N]$ as the rating matrix A by reformulating the sigmoid function. The choice of N is 5 in many real-world five stars based reviewing systems, such as Amazon and Yelp.

3.4 Item-Feature Quality Matrix

We also construct item-feature quality matrix Y, where each element measures the quality of an item for the corresponding product feature/aspect.

Let $\mathcal{P} = \{p_1, p_2, \cdots, p_n\}$ denote the n items/products. For each of the items p_i, we use all of its corresponding reviews, and extract the corresponding (F, S') pairs. Suppose feature F_j is mentioned for k times on item p_i, and the average of sentiment of feature F_j in those k mentions are s_{ij}. We define the item-feature measure Y_{ij} as:

$$Y_{ij} = \begin{cases} 0, \ \textit{if item } p_i \textit{ is not reviewed on feature } F_j \\ 1 + \dfrac{N-1}{1+e^{-k \cdot s_{ij}}}, \ \textit{else} \end{cases} \tag{3}$$

This measure captures both the sentiment orientation (through s_{ij}) and the popularity (through k) of feature F_j for product p_i. It is also rescaled into the range of $[1, N]$.

3.5 Integrating Explicit and Implicit Features

The non-zeros in matrices X and Y indicate the observed relations between users, items and explicit features. Now we exposit how to integrate these into a factorization model for both accurate prediction and explainable recommendations.

Similar to factorization models over user-item rating matrix A, we can build a factorization model over user-feature attention matrix X and item-feature quality matrix Y, which means estimating hidden representations of users, features, and items based on the observed user-feature and item-feature relations. This can be done as follows:

$$\begin{gathered} \underset{U_1, U_2, V}{\text{minimize}} \Big\{ \lambda_x \|U_1 V^T - X\|_F^2 + \lambda_y \|U_2 V^T - Y\|_F^2 \Big\} \\ \text{s.t. } U_1 \in \mathbb{R}_+^{m \times r}, U_2 \in \mathbb{R}_+^{n \times r}, V \in \mathbb{R}_+^{p \times r} \end{gathered} \tag{4}$$

where λ_x and λ_y are regularization coefficients, and the number of *explicit factors* r represents the number of factors.

We assume a user's overall star ratings about an item (i.e. an element in matrix A) is based on her underlying opinions over various product aspects. Based on this assumption, we estimate the rating matrix using the latent representations U_1 and U_2, which captures the user attentions and item qualities on the explicit product features. However, we also acknowledge that the explicit features might not be able to fully explain a rating and a user might consider some other hidden factors when making a decision (i.e. rating) about a product. As a result, we also introduce r' *latent factors* $H_1 \in \mathbb{R}_+^{m \times r'}$ and $H_2 \in \mathbb{R}_+^{n \times r'}$, and use $P = [U_1 \ H_1]$ and $Q = [U_2 \ H_2]$ to model the overall rating matrix A as follows:

$$\underset{P,Q}{\text{minimize}} \Big\{ \|PQ^T - A\|_F^2 \Big\} \tag{5}$$

To put Eq.(4) and Eq.(5) together, we have a factorization model that integrates explicit and implicit features.

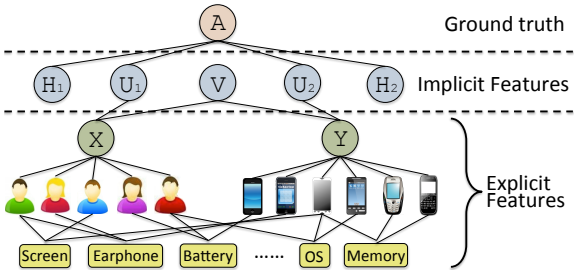


Figure 3: The relationships of product features, partially observed matrices, and hidden factors in the EFM framework.

The underlying relationships of product features, partially observed matrices, and hidden factors are illustrated in Figure 3. The hidden factors can be estimated through the following optimization task:

$$\begin{aligned} \underset{U_1, U_2, V, H_1, H_2}{\text{minimize}} \Big\{ & \|PQ^T - A\|_F^2 + \lambda_x \|U_1 V^T - X\|_F^2 + \lambda_y \|U_2 V^T - Y\|_F^2 \\ & + \lambda_u (\|U_1\|_F^2 + \|U_2\|_F^2) + \lambda_h (\|H_1\|_F^2 + \|H_2\|_F^2) + \lambda_v \|V\|_F^2 \Big\} \\ & \text{s.t. } U_1 \in \mathbb{R}_+^{m \times r}, U_2 \in \mathbb{R}_+^{n \times r}, V \in \mathbb{R}_+^{p \times r}, H_1 \in \mathbb{R}_+^{m \times r'}, \\ & H_2 \in \mathbb{R}_+^{n \times r'} \text{and } P = [U_1 \ H_1], \ Q = [U_2 \ H_2] \end{aligned} \tag{6}$$

When $r = 0$, this model reduces to a traditional latent factorization model on user-item rating matrix A, which means that the explicit features are not used for recommendations. The optimal solution of Eq.(6) can be further used for making recommendations and explanations.

3.6 Model Learning for EFM

There is no closed-form solution for Eq.(6). Motivated by [5], we introduce an alternative minimization algorithm (Algorithm 1) to find the optimal solutions for the five parameters U_1, U_2, V, H_1, H_2. The key idea is to optimize the

Algorithm 1: EXPLICIT FACTOR MODEL

Input: $A, X, Y, m, n, p, r, r', \lambda_x, \lambda_y, \lambda_u, \lambda_h, \lambda_v, T$
Output: U_1, U_2, V, H_1, H_2

$U_1 \leftarrow \mathbb{R}_+^{m \times r}$, $U_2 \leftarrow \mathbb{R}_+^{n \times r}$, $V \leftarrow \mathbb{R}_+^{p \times r}$;
$H_1 \leftarrow \mathbb{R}_+^{m \times r'}$, $H_2 \leftarrow \mathbb{R}_+^{n \times r'}$; //initialize randomly
$t \leftarrow 0$;
repeat
  $t \leftarrow t + 1$;
  Update: $V_{ij} \leftarrow V_{ij} \sqrt{\frac{[\lambda_x X^T U_1 + \lambda_y Y^T U_2]_{ij}}{[V(\lambda_x U_1^T U_1 + \lambda_y U_2^T U_2 + \lambda_v I)]_{ij}}}$
  Update:
  $U_{1ij} \leftarrow U_{1ij} \sqrt{\frac{[AU_2 + \lambda_x XV]_{ij}}{[(U_1 U_2^T + H_1 H_2^T) U_2 + U_1(\lambda_x V^T V + \lambda_u I)]_{ij}}}$
  Update:
  $U_{2ij} \leftarrow U_{2ij} \sqrt{\frac{[A^T U_1 + \lambda_y YV]_{ij}}{[(U_2 U_1^T + H_2 H_1^T) U_1 + U_2(\lambda_y V^T V + \lambda_u I)]_{ij}}}$
  Update: $H_{1ij} \leftarrow H_{1ij} \sqrt{\frac{[AH_2]_{ij}}{[(U_1 U_2^T + H_1 H_2^T) H_2 + \lambda_h H_1]_{ij}}}$
  Update: $H_{2ij} \leftarrow H_{2ij} \sqrt{\frac{[A^T H_1]_{ij}}{[(U_2 U_1^T + H_2 H_1^T) H_1 + \lambda_h H_2]_{ij}}}$
until *Convergence* **or** $t > T$;
return U_1, U_2, V, H_1, H_2;

objective function with respect to one parameter, while fixing the other four. The algorithm keeps updating the parameters repeatedly until convergence or reaching the maximum number of iterations. Due to the limited space, we sketch the derivations of the updating rules in the Appendix. Similar to [15], the correctness and convergence of Algorithm 1 can be proved with the standard auxiliary function approach.

3.7 Personalized Recommendation

Given the optimal solution of the factorization model, we can estimate any missing element in the user-feature attention matrix X, item-feature quality matrix Y and user-item rating matrix A, as $\tilde{X} = U_1 V^T$, $\tilde{Y} = U_2 V^T$, and $\tilde{A} = U_1 U_2^T + H_1 H_2^T$. Based on that, we can generate personalized top-K recommendations and provide feature-level explanations, as described in the following subsections.

3.7.1 Top-K Recommendation

We assume that a user's decision about whether to make a purchase is based on several important product features to him or her, rather than considering all hundreds of possible features. Thus we use the most cared k features of a user when generating the recommendation list for him/her. For user $u_i (1 \leq i \leq m)$, let the column indices of the k largest values in row vector $\tilde{X}_{i\cdot}$ be $\mathcal{C}_i = \{c_{i1}, c_{i2}, \cdots, c_{ik}\}$. We set the ranking score of item $p_j (1 \leq j \leq n)$ for user u_i as follows:

$$R_{ij} = \alpha \cdot \frac{\sum_{c \in \mathcal{C}_i} \tilde{X}_{ic} \cdot \tilde{Y}_{jc}}{kN} + (1 - \alpha)\tilde{A}_{ij} \tag{7}$$

where $N = \max(A_{ij})$ is used to rescale the first part and $N = 5$ in most rating systems. The first part is a user-item similarity score based on the k most important product features that user u_i cares. $0 \leq \alpha \leq 1$ is a scale that controls the trade off between feature based score and direct user-item ratings. The top-K recommendation list for user u_i can be constructed by ranking the items based on R_{ij}.

3.7.2 Personalized Feature-Level Explanation

Traditional factorization models are usually difficulty to understand and it is hard to automatically generate explanations about why or why not an item is recommended through the hidden factors. An important advantage of our EFM framework is its ability to analyze which of the various features play a key role in pushing an item to the top-K. Besides recommending products, we also study an uncommon usage scenario of our framework: disrecommending an item that the system considers "does not worth buying" when a user is viewing it. We believe that this is an important usage of a filtering/recommendation system, and that explaining why the system "thinks" the user should not buy a product might gain user trust and help user make a more informed purchasing decision.

There could be many different ways to construct effective explanations with the explicit features. However, due to the limited experimental resources, we designed a straightforward template based explanation and an intuitional word cloud based explanation for the recommended and disrecommended items. The templates are shown as follows:

`You might be interested in [feature], on which this product performs well.`
`You might be interested in [feature], on which this product performs poorly.`

For each user u_i and a recommended item p_j, the feature used for explanation construction is F_c, where:

$$c = \text{argmax}_{c \in \mathcal{C}_i} \tilde{Y}_{jc} \tag{8}$$

While for each disrecommended item p_j, the feature F_c is:

$$c = \text{argmin}_{c \in \mathcal{C}_i} \tilde{Y}_{jc} \tag{9}$$

We focus on the persuasiveness of the explanation. Among the user's most cared features, we select the one that has the best or worst performance on the corresponding product.

The word cloud based explanation further attempts to validate the textual explanations by listing the feature-opinion pairs that are matched on the reviews of a (dis)recommended item. The recommendation interface, word cloud as well as the practical effects regarding user acceptance of the recommendations will be discussed in details later.

4. OFFLINE EXPERIMENTS

In this section, we conduct extensive experiments to evaluate the EFM framework in offline settings. We mainly focus on the following research questions: 1) How do users care about the various explicit product features extracted from reviews, and 2) What is the performance of the EFM framework in both the task of rating prediction and the more practical task of top-K recommendation.

We begin by introducing the experimental setup, and report the quality evaluation of the constructed sentiment lexicon, then we report and analyze the experimental results to attempt to answer the research questions.

4.1 Experimental Setup

We choose the Yelp[1] and Dianping[2] datasets (Table 2) for experimentations. The Yelp dataset consists of user reviews on the businesses located in the Phoenix City of the US, and the Dianping dataset consists of the reviews on the restaurants located in several main cities of China, where each user made 20 or more reviews.

Table 2: Statistics of Yelp and Dianping datasets.

Dataset	#users	#items	#reviews	$\frac{\#reviews}{\#users}$
Yelp	45,981	11,537	229,907	5.00
Dianping	11,857	22,365	510,551	43.06
Yelp10	4,393	10,801	138,301	31.48

These datasets are of two different languages, which are English on Yelp, and Chinese on Dianping. As a kind of natural language processing techniques, phrase-level sentiment analysis could be language dependent. By these completely different language settings we would like to verify whether our framework works in different language environments.

It is necessary to note that the Yelp dataset includes those users that made very few reviews. In fact, 49% of the users made only one review, which makes it difficult to evaluate the performance of top-K recommendation. As a result, we selected the users with 10 or more reviews for the experimentation of top-K recommendation, which constitutes the dataset 'Yelp10', as shown at the bottom line of Table 2.

The maximum number of iterations T in Algorithm 1 is set to 100 to ensure convergence. We conducted grid search for the hyper-parameters one-by-one in the range of $(0, 1]$ with a step size of 0.05, and 5-fold cross-validation was conducted in performance evaluation for all methods.

[1] `http://www.yelp.com/dataset_challenge`
[2] `http://www.dianping.com`

4.2 Sentiment Lexicon Constructed

We shall note that, the task of phrase-level sentiment lexicon construction is inherently difficult. One always need to trade off between precision and recall. As a primary step towards using sentiment lexicon for EFM, we focus on the precision as we will only use the top features in our framework, primarily to avoid the negative effects of wrong features as much as possible. We expect as the research in sentiment analysis advances, the performance of our framework will further improve as well.

Similar to [19], manual labellings from three human annotators are used to evaluate the lexicon quality. A feature word or opinion word is considered to be proper if it is approved by at least two annotators. We found that the average agreement among annotators is 85.63%. To evaluate the sentiments, we transform each polarity value $S \in [-1, 1]$ into a binary value (*positive* or *negative*). There is no *neutral* as $S = 0$ does not exist. The annotators were asked to label the sentiment correctness for each entry, and the average agreement is 82.26%. The statistics and evaluation results of the lexicons are shown in Table 3.

Table 3: Some statistics and evaluation results of the sentiment lexicons, where 'F,O,S' are for feature word, opinion word and sentiment respectively, and 'Prec' stands for 'Precision'.

Dataset	#F	#O	#entries	Prec(F)	Prec(O)	Prec(S)
Yelp	96	155	845	92.71%	91.61%	94.91%
Dianping	113	284	1,129	89.38%	89.79%	91.41%

The lexicon construction process gives us around 100 high quality product features in both datasets. Some sampled entries on Yelp are presented in Table 4, showing how the sentiment lexicon looks in different *contextual* text reviews.

Table 4: Sampled entries from the Yelp dataset.

Feature Dependent	Opinion Dependent
(decor, cool, +)	(price, fire, +)
(service, cool, -)	(price, high, -)
(price, high, -)	(parking space, plenty, +)
(service quality, high, +)	(parking space, limit, -)

To verify the assumption that users might care about different features, we construct the user-feature attention matrix X on 'Yelp' dataset, and conduct a simple fuzzy k-means clustering to map the users into 5 fuzzy clusters within the Euclidean distance space. The reason to choose fuzzy clustering is due to the fact that a users might be interested in two or more product aspects. For the k-th ($k = 1 \cdots 5$) cluster, we rank the features according to the weighted sum frequency $\text{Freq}(F_j) = \sum_{i=1}^{m} w_{ik} t_{ij}$, where w_{ik} is the degree that user u_i belongs to the k-th cluster, and t_{ij} is the frequency that user u_i mentioned feature F_j. We then select the top-5 frequent features from each cluster and rank them in descending order of term frequency, as listed in Table 5.

Table 5: Top-5 frequent features of the 5 clusters.

cluster 1	cluster 2	cluster 3	cluster 4	cluster 5
food	service	place	price	beer
lunch	menu	area	order	drink
service	food	location	food	bar
meal	order	restaurant	service	order
experience	staff	bar	menu	wine

We see that the users in each cluster care about a different subset of features, and each subset mainly reveals a different product aspect. The modularity (measures the quality of a division of a network into communities, which is commonly used in community detection [23]) among the clusters is $Q = 0.605$, exhibiting a high inner-cluster relationship against inter-cluster relationship of the users. The results indicate that users do comment on different features, which matches our assumption that users care about different aspects. In the following experiments, we will continue to investigate the effect of explicit features in various conditions.

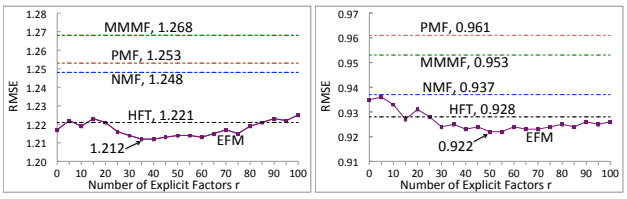
We use the original sentiment lexicon and do not use any human annotation information in the following experiments, so as to avoid any manual effort in our framework.

4.3 Rating Prediction

We investigate the performance in approximating the rating matrix A. Three popular and state-of-the-art latent factor models are chosen for comparison, which are Nonnegative Matrix Factorization (NMF) [15], Probabilistic Matrix Factorization (PMF) [30] and Max-Margin Matrix Factorization (MMMF) [29]. The hyper-parameters are selected by grid search in 5-fold cross-validation. We also compared with the Hidden Factors as Topics (HFT) model in [20], which achieved the state-of-the-art performance in terms of leveraging text reviews by extracting hidden topics. We set the number of topics as 5 in HFT as reported in [20], and assigning more topics does not further improve the performance. We evaluate by Root Mean Square Error (RMSE).

4.3.1 Ratio of Explicit and Latent Factors

The number of explicit factor r and latent factors r' are important because they capture two different types of features in the factorization process. In this subsection, we investigate the effect of explicit and latent factors by fixing their total number $r + r' = 100$, and tuning their ratio. We also use 100 latent factors for NMF, PMF and MMMF to ensure equal model complexity[3]. The experimental results of RMSE vs the number of explicit factors r are shown in Figure 4. The standard deviations in 5-fold cross-validation of each baseline algorithm and on each experimental point of our EFM framework are ≤ 0.002.



(a) Yelp dataset (b) Dianping dataset

Figure 4: RMSE vs Number of Explicit Factors r.

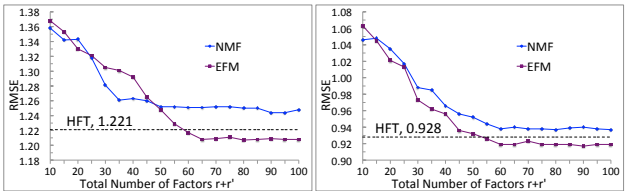
We see that when using a small number of explicit factors and keeping the majority to be latent, the performance of EFM is comparable to that of HFT or NMF. This is as expected, because the EFM algorithm simplifies into a kind of nonnegative matrix factorization when $r = 0$. However, when an appropriate percentage of explicit factors (around 30% $\sim$ 80%) are used, EFM is statistically significantly better than the baselines. The best prediction accuracy RMSE=1.212 is achieved on Yelp when $r = 35$, and the best RMSE=0.922 when $r = 50$ on Dianping. We also found that too many explicit factors hurts the performance. This suggests that although incorporating explicit factors improves

[3]Besides, increasing the number of factors beyond 100 would not affect the performance much.

the prediction accuracy, keeping a moderate amount of latent factors is necessary to ensure model flexibility, as the explicit factors might not capture all user criteria completely.

4.3.2 Total Number of Factors

We further investigate the total number of factors $r+r'$ to verify the performance under different model complexities. We fix the percentage of explicit factors to 40%, which gives near optimal performances on both datasets in the previous experiment. For comparison, we experiment with HFT, which achieved the best performance among baseline algorithms, and NMF, which is the best among three of the LFM models. The results are shown in Figure 5.



(a) Yelp dataset (b) Dianping dataset

Figure 5: RMSE vs total number of factors $r+r'$.

For both the NMF algorithm and EFM framework, RMSE tends to drop as $r+r'$ increases. However, the performance of EFM would not surpass NMF or HFT until $r+r'$ is big enough. This suggests that we need enough latent factors in order to make EFM framework work effectively. Nevertheless, as long as the number of latent factors are big enough to capture the underlying user rating patterns, the existence of explicit factors further improves rating prediction.

4.4 Top-K Recommendation

In this set of experiment, we study the EFM framework in the top-K recommendation task. We did further analysis of the framework to find whether and how the performance is affected by some specific features, and which features are of key importance to the users in the task.

We still set the percentage of explicit factors to be 40%. We set the total number of factors $r+r'$ as 65, a number that is big enough on both datasets, because assigning more factors would not affect the performance much. We compared EFM with the following baseline algorithms:

MostPopular: Non-personalized recommendation where the items are ranked by the number of ratings given by users.

SlopeOne: A neighborhood-based CF algorithm with easy implementation and comparable or better performance than user- or item-based approaches [16].

NMF: Nonnegative Matrix Factorization algorithm that is among the best latent factor models in the previous experiment.

BPRMF: Bayesian Personalized Ranking (BPR) optimization for MF [28], which is the state-of-the-art algorithm for top-K recommendation with numerical ratings.

HFT: The state-of-the-art algorithm in terms of making rating prediction with textual reviews [20].

We conduct top-5 (K=5) and top-10 (K=10) recommendation on Yelp10 and Dianping correspondingly, as the minimum number of reviews per user is 10 on Yelp10 and 20 on Dianping. We holdout the latest K reviewed items from each user for testing, assuming they are the only relevant items. The other reviews are used for training. Five-fold cross-validation is used for parameter tuning and performance evaluation. Following [28], we used Normalized Discounted Cumulative Gain (NDCG) and Area Under the ROC Curve (AUC) to help evaluate the performance of different models.

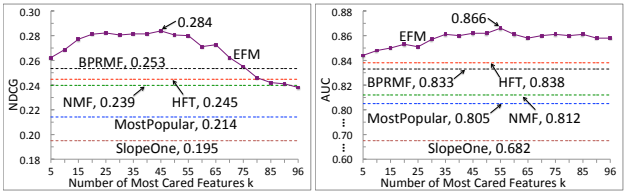
Table 6: Top-K recommendation results on Dianping dataset, where the result listed for EFM is the best performance with the corresponding k.

Method	MP	SO	NMF	BPRMF	HFT	EFM	
NDCG	0.244	0.212	0.216	0.238	0.261	$k=50$	**0.284**
AUC	0.837	0.785	0.832	0.856	0.873	$k=50$	**0.884**

4.4.1 Number of Most Cared Features

We first fix the number of most cared features $k=10$ and found that the optimal value for the weighing scalar α is 0.85 in terms of NDCG. Then we fix $\alpha=0.85$ throughout the following experiments to focus on the key parameter k.

We study how the performance (NDCG and AUC) changes as k increases from 5 to the maximum value possible (96 for Yelp10 and 113 for Dianping), and the results on Yelp10 are shown in Figure 6. It shows that the performance of EFM continues to rise with the increase of k until around 15, and tends to be stable before it begins to drop when $k=45$. It outperforms all other algorithms when $k \leq 70$, and in the best case where $k=45$, EFM is 12.3% better than the best baseline algorithm BPRMF. However, the NDCG tends to drop when k is too high. The results on Dianping are similar, as shown in Table 6. EFM is statistically significantly better than other algorithms on Yelp10 and Dianping datasets. The standard deviations of both NDCG and AUC in five-fold cross-validation for each baseline algorithm and on each experimental point of EFM are ≤ 0.006.



(a) NDCG vs k (b) AUC vs k

Figure 6: NDCG and AUC vs the number of most cared features k on the Yelp10 dataset.

This observation confirms the hypothesis that incorporating too many features would introduce noise into the recommendation procedure, which is consistent with the observations in a recent work HFT [20]. However, the results on AUC show that the EFM framework is better than the comparative algorithms consistently. As AUC evaluates only the pairwise rankings rather than the positions, this observation suggests that incorporating irrelevant features affects the position rankings of relevant items more than their pairwise rankings.

4.4.2 Further Analysis of Explicit Features

Although the results show that the performance on NDCG begins to keep stable when $k \geq 15$, it is still surprising for us, as we do not expect that users would consider tens of features when making decisions. To further analyze the user-feature relationships and learn the impact of explicit features, we calculate the averaged coverage of the k most cared features in term frequency, against all the features in

the reviews of a user, as shown in Eq.(10), where t_{ij} is the term frequency of feature F_j in the reviews of user u_i,

$$\text{Coverage@}k = \frac{1}{|U|}\sum_{i=1}^{|U|}\frac{\sum_{j\in\mathcal{C}_i} t_{ij}}{\sum_{j=1}^{p} t_{ij}} \tag{10}$$

and the relationship of Coverage versus the number of most cared features k is shown in Figure 7 below.

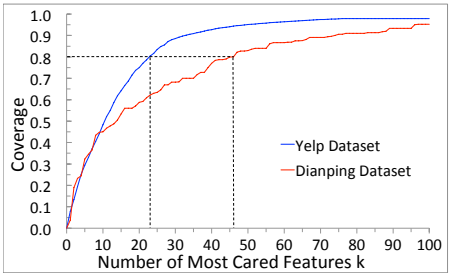


Figure 7: Coverage of term frequency vs the number of most cared features k on both of the datasets.

We notice that a small number of the most cared features dominate the coverage of term frequency in user reviews. For example, about 24 out of 96 features cover up to 80% of the feature occurrences on Yelp. This implies that users usually put most of the attentions on several most cared features, which verifies our motivation to adopt the top k features for recommendation, because incorporating more features could bring about negative effects as they might be irrelevant to users' preferences.

However, it needs 46 features to achieve the same 80% coverage on Dianping dataset, which is nearly twice the number of Yelp10. To understand the reason, we group the feature words into synonym clusters using WordNet[4] on Yelp10, and HowNet[5] on Dianping. We find that different feature words are used to describe the same aspect of the items. For example, the features *price* and *cost* are grouped into a single cluster. Some key statistics are shown in Table 7.

Table 7: Key statistics of synonym clusters.

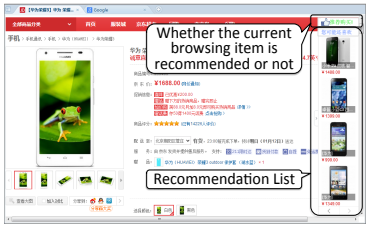
Dataset	#Feature	#Cluster	#F/#C
Yelp10	96	31	3.10
Dianping	113	26	4.35

The results show that on average there are more synonyms in the Chinese language, and the existence of synonyms dilutes the explicit feature space. For example, while there are two features *price* and *cost* grouped into one cluster on Yelp10, the semantically corresponding cluster on Dianping contains four feature words. This suggests that the optimal number of features might be different for different languages. The top 15 features on Yelp10 are grouped into 7 clusters, as shown in Table 8, where the top 15 features are bolded and the clusters are ranked by total term frequency.

Table 8: Word clusters of the top 15 features.

1	**place**, **restaurant**, **location**, area, way	2	**food**, **menu**, **lunch**, **pizza**, dinner
3	**service**, **time**, **staff**, order	4	**experience**, quality
5	**room**, **atmosphere**, decor	6	**price**, cost
7	**beer**, wine, drink, water, coffee		

[4]http://wordnet.princeton.edu
[5]http://www.keenage.com



(a) In browser recommendation

(b) Word cloud

Figure 8: Top-4 recommended items are presented by the browser at right hand side when the user is browsing an online product, and the feature-opinion word pair cloud is displayed to assist explanations when the user hovers on a recommended item.

The observations tell us that while the NDCG continues to rise when $k \leq 15$, the underlying features are still limited to about 7 key aspects. Besides, we also find that incorporating the top $15 \sim 55$ features are in fact appending features into the previously existed clusters. This might explain why NDCG keeps stable in that range before getting to drop.

The experimental results confirmed our intuition of focusing on the top product features in recommender systems. More importantly, incorporating explicit features makes it possible to leverage the many promising natural language processing techniques in recommender systems, which can be used to analyze the specific interests of different users, and thus to develop effective marketing strategies in e-commerce.

5. ONLINE EXPERIMENTS

In this section, we conduct online experiments with real-world e-commerce users to investigate the effect of automatically generated intuitional feature-level explanations, focusing on how the explains affect users' acceptance of the recommendations (i.e. persuasiveness).

5.1 Experimental Setup

We conduct A/B-tests based on a popular commercial web browser which has more than 100 million users, with 26% monthly active users. Our experiments attempt to recommend relevant phones when a user is browsing mobile phones in a very popular online shopping website JingDong[6].

Figure 8(a) shows our recommendation interface, where the recommendations for the current browsing item are displayed in the right side popup panel. At the top of the panel is an 'Indicator', which shows whether the current browsing item is recommended or disrecommended, followed by a 'List' of top-4 recommendations for the current user. Figure 8(b) shows the word cloud where the positive feature-opinion pairs are green and negative pairs are blue, and the size is proportional to the term frequency in the reviews of the recommended item. For example, the largest pair in Figure 8(b) means "PictureClarity-High".

We select those users who made ten or more reviews as target users according to the browser log, and collect their reviews to employ the EFM framework to generate (dis)recommendations and explanations for these selected users. The textual explanation and word cloud for the 'Indicator' or a recommended item in the 'List' will be displayed in a popup panel when the mouse is hovered on an item, so that we can detect whether a user has examined a recommendation and its explanation.

[6]http://www.jd.com

5.2 Recommendation List Explanation

To study the explanations in the 'List', we randomly assigned each subject to one of the following three groups. The A (experimental group) users receive our feature-level explanations, the B (comparison group) users receive the famous 'People also viewed' explanations, and the C (control group) users receive no explanation. In our result analysis, we only consider those users who hovered the mouse on the List, so we know that they have examined the recommendation explanations. Besides, we only consider the browsing logs corresponding to those items common to A, B and C, resulting in 44,681 records related to 944 common items. The results of Click Through Rate (CTR) are shown in Table 9.

Table 9: The number of browsing records, clicks and click through rate for the three user types.

User Set	A		B		C	
Records	#Record	#Click	#Record	#Click	#Record	#Click
	15,933	691	11,483	370	17,265	552
CTR	4.34%		3.22%		3.20%	

Based on ten-fold t-test, we found that the CTR of the experimental group A is significantly higher than that of the comparison group B and control group C, at $p = 0.033$ and 0.041, respectively, demonstrating that our feature-level explanations are more effective in persuading users to examine the recommended items in terms of click through rate in e-commerce recommendation scenarios.

5.3 (Dis)Recommendation Explanation

To study how (dis)recommendation explanations affect a user, we present the Indicator for both user group A (experimental group) and B (control group). The only difference is that the A users could see the popup explanations when hovering the mouse over the Indicator, while nothing shows for the B users. We did not assign a comparison group with other explanations, because to the best of our knowledge, there is no previous work on presenting disrecommnedation explanations. Due to some security reasons, we were unable to track whether a user paid for an item, so we recorded whether a user added the current browsing item into his/her cart as an approximation of user purchasing behavior.

This online experiment resulted in 53,372 browsing records of 1,328 common items, including 20,735 records from 582 A users and 32,637 records from 733 B users. The overall confusion matrix is shown in Table 10, where 'Recommend' means that the current item is recommended, and 'Add' means that the item is added to the cart. Besides, AddToCart%=$\frac{x_{11}+x_{21}}{x_{11}+x_{12}+x_{21}+x_{22}}$ is the total percentage of adding to cart; Agreement%=$\frac{x_{11}+x_{22}}{x_{11}+x_{12}+x_{21}+x_{22}}$ is the total percentage of user agreement with the (dis)recommendation; RecAgree%=$\frac{x_{11}}{x_{11}+x_{12}}$ and DisRecAgree%=$\frac{x_{22}}{x_{21}+x_{22}}$ are the percentages of user agreement when an item is recommended or disrecommended, respectively.

Both AddToCart% and Agreement% of the experimental group A are significantly higher than the control group B, with the two-tailed p-values being 0.0374 and 0.0068, respectively. This demonstrates that presenting the explanations improves recommendation persuasiveness, and at the same time improves the conversion (i.e. adding-to-cart) rate in real-world online shopping scenarios.

More interestingly, we find that presenting disrecommnedation explanations helps to prevent users from adding an item to cart when the system "thinks" that the item is not good enough for him/her. This observation reveals a strong persuasiveness and flexibility of the feature-level explanations, and it might bring new and promising marketing and advertising strategies for e-commerce users.

Table 10: The overall confusion matrix corresponding to the 1328 common items of A (with explanations) and B (without explanations) users.

Confusion Matrix	A				B			
	Add		¬Add		Add		¬Add	
Recommend	x_{11}	1,261	x_{12}	16,572	x_{11}	1,129	x_{12}	28,218
¬Recommned	x_{21}	72	x_{22}	2,830	x_{21}	541	x_{22}	2,749
AddToCart%	6.43%				5.12%			
Agreement%	19.73%				11.89%			
RecAgree%	7.07%				3.85%			
DisRecAgree%	97.5%				83.56%			

6. CONCLUSIONS AND FUTURE WORK

In this paper, we propose to leverage phrase-level sentiment analysis of user reviews for personalized recommendation. We extract explicit product features and user opinions from reviews, then incorporate both user-feature and item-feature relations as well as user-item ratings into a new unified hybrid matrix factorization framework. Besides generating personalized recommendations, we designed explicit feature-level explanations for both recommended and disrecommended items. Our analysis shows that different users could focus on different product aspects, and our experiments suggest that the size of the underlying feature space that users care about varies for different users, domains and countries. Both online and offline experiments show that our framework compares favourably with baseline methods in three tasks: rating prediction, top-K recommendation, and explanation based user persuasion.

This is a first step towards integrating detailed sentiment analysis for aspect/feature based explainable hybrid factorization models for recommendations, and there are much room for improvements. Instead of one measure per aspect, we can introduce more measures (sentiment, popularity, etc.) or capture more complex interaction between features in the future. Except for EFM, we can adapt or develop other hybrid factorization models such as tensor factorization or deep learning to integrate phrase-level sentiment analysis. Besides, we focused on the persuasiveness in explanation generation and experimental design, while it is worth studying other utilities (transparency, user trust, effectiveness, efficiency, satisfaction, and scrutability, etc.) of explanations and generating explanations automatically to optimize one or a combination of the utilities in the future.

7. APPENDIX

The objective function in Eq.(6) with respect to U_1 is:

$$\min_{U_1 \geq 0} \left\{ \|U_1 U_2^T + H_1 H_2^T - A\|_F^2 + \lambda_x \|U_1 V^T - X\|_F^2 + \lambda_u \|U_1\|_F^2 \right\} \quad (11)$$

Let Λ be the Lagrange multiplier for the constraint $U_1 \geq 0$, then the Lagrange function is:

$$L(U_1) = \|U_1 U_2^T + H_1 H_2^T - A\|_F^2 + \lambda_x \|U_1 V^T - X\|_F^2 + \lambda_u \|U_1\|_F^2 - \mathrm{tr}(\Lambda U_1) \quad (12)$$

and the corresponding gradient is:

$$\nabla_{U_1} = 2(U_1 U_2^T + H_1 H_2^T - A)U_2 + 2\lambda_x (U_1 V^T - X)V + 2\lambda_u U_1 - \Lambda \quad (13)$$

By setting $\nabla_{U_1} = 0$ we get:

$$\Lambda = 2(U_1 U_2^T U_2 + H_1 H_2^T U_2 + \lambda_x U_1 V^T V + \lambda_u U_1) - 2(AU_2 + \lambda_x XV) \quad (14)$$

With the KKT complementary condition for the constraint $U_1 \geq 0$, we have $\Lambda_{ij} \cdot {U_1}_{ij} = 0$, giving us the following:

$$[-(AU_2+\lambda_x XV)+(U_1 U_2^T U_2+H_1 H_2^T U_2+\lambda_x U_1 V^T V+\lambda_u U_1)]_{ij} \cdot {U_1}_{ij} = 0 \quad (15)$$

which further leads to the updating rule of U_1:

$${U_1}_{ij} \leftarrow {U_1}_{ij} \sqrt{\frac{[AU_2 + \lambda_x XV]_{ij}}{[(U_1 U_2^T + H_1 H_2^T)U_2 + U_1(\lambda_x V^T V + \lambda_u I)]_{ij}}} \quad (16)$$

The updating rules for U_2, V, H_1, H_2 can be derived in a similar way.

Acknowledgement

The authors thank the reviewers for the constructive suggestions. Part of this work was supported by Chinese Natural Science Foundation (60903107, 61073071) and National High Technology Research and Development Program (2011AA01 A205), and the fourth author is sponsored by the National Science Foundation (IIS-0713111). The opinions, findings or conclusions expressed in this paper are the authors', and do not necessarily reflect those of the sponsors.

8. REFERENCES

[1] S. Aciar, D. Zhang, S. Simoff, and J. Debenham. Informed Recommender: Basing Recommendations on Consumer Product Reviews. *Intelligent Systems*, 22(3):39–47, 2007.

[2] M. Bilgic and R. J. Mooney. Explaining Recommendations: Satisfaction vs. Promotion. *IUI*, 2005.

[3] H. Cramer, V. Evers, S. Ramlal, M. van Someren, et al. The Effects of Transparency on Trust in and Acceptance of a Content-Based Art Recommender. *User Modeling and User-Adapted Interaction*, 18(5):455–496, 2008.

[4] P. Cremonesi, Y. Koren, and R. Turrin. Performance of Recommender Algorithms on Top-N Recommendation Tasks. *RecSys*, pages 39–46, 2010.

[5] C. Ding, T. Li, W. Peng, and H. Park. Orthogonal Nonnegative Matrix Tri-Factorizations for Clustering. *KDD*, pages 126–135, 2006.

[6] X. Ding, B. Liu, and P. S. Yu. A Holistic Lexicon Based Approach to Opinion Mining. *WSDM*, 2008.

[7] G. Ganu, N. Elhadad, and A. Marian. Beyond the Stars: Improving Rating Predictions using Review Text Content. *WebDB*, 2009.

[8] X. He, M. Gao, M. Kan, Y. Liu, and K. Sugiyama. Predicting the Popularity of Web 2.0 Items based on User Comments. *SIGIR*, 2014.

[9] X. He, M. Kan, P. Xie, and X. Chen. Comment-based Multi-View Clustering of Web 2.0 Items. *WWW*, 2014.

[10] J. Herlocker, J. Konstan, and J. Riedl. Explaining collaborative filtering recommendations. *CSCW*, 2000.

[11] M. Hu and B. Liu. Mining and Summarizing Customer Reviews. *KDD*, pages 168–177, 2004.

[12] N. Jakob, S. H. Weber, M. C. Müller, et al. Beyond the Stars: Exploiting Free-Text User Reviews to Improve the Accuracy of Movie Recommendations. *TSA*, 2009.

[13] C. W. ki Leung, S. C. fai Chan, and F. lai Chung. Integrating Collaborative Filtering and Sentiment Analysis: A Rating Inference Approach. *ECAI*, 2006.

[14] Y. Koren, R. Bell, and C. Volinsky. Matrix Factorization Techniques for Recommender Systems. *Computer*, 2009.

[15] D. D. Lee and H. S. Seung. Algorithms for Non-negative Matrix Factorization. *Proc. NIPS*, 2001.

[16] D. Lemire and A. Maclachlan. Slope One Predictors for Online Rating-Based Collaborative Filtering. *SDM*, 2005.

[17] B. Liu, M. Hu, and J. Cheng. Opinion Observer: Analyzing and Comparing Opinions on the Web. *WWW*, 2005.

[18] B. Liu and L. Zhang. A Survey of Opinion Mining and Sentiment Analysis. *Jour. Mining Text Data*, 2012.

[19] Y. Lu, M. Castellanos, U. Dayal, and C. Zhai. Automatic construction of a context-aware sentiment lexicon: An optimization approach. *WWW*, 2011.

[20] J. McAuley and J. Leskovec. Hidden Factors and Hidden Topics: Understanding Rating Dimensions with Review Text. *RecSys*, pages 165–172, 2013.

[21] C. Musat, Y. Liang, and B. Faltings. Recommendation Using Textual Opinions. *IJCAI*, 2013.

[22] T. Nakagawa, K. Inui, and S. Kurohashi. Dependency Tree-based Sentiment Classification using CRFs with Hidden Variables. *NAACL*, 2010.

[23] M. Newman and M. Girvan. Finding and evaluating community structure in networks. *Phys. Review*, 2004.

[24] B. Pang and L. Lee. Opinion Mining and Sentiment Analysis. *Found. & Trends in Info. Retr.*, 2(1-2), 2008.

[25] B. Pang, L. Lee, et al. Thumbs up? sentiment classification using machine learning techniques. *EMNLP*, 2002.

[26] N. Pappas and A. P. Belis. Sentiment Analysis of User Comments for One-Class Collaborative Filtering over TED Talks. *SIGIR*, pages 773–776, 2013.

[27] S. Pero and T. Horvath. Opinion-Driven Matrix Factorization for Rating Prediction. *UMAP*, 2013.

[28] S. Rendle, C. Freudenthaler, et al. BPR: Bayesian Personalized Ranking from Implicit Feedback. *UAI*, 2009.

[29] J. Rennie and N. Srebro. Fast Maximum Margin Matrix Factorization for Collaborative Prediction. *ICML*, 2005.

[30] R. Salakhutdinov and A. Mnih. Bayesian Probabilistic Matrix Factorization using Markov Chain Monte Carlo. *Proc. ICML*, 2008.

[31] A. Sharma and D. Cosley. Do Social Explanations Work? Studying and Modeling the Effects of Social Explanations in Recommender Systems. *WWW*, 2013.

[32] N. Srebro and T. Jaakkola. Weighted Low-rank Approximations. *Proc. ICML*, pages 720–727, 2003.

[33] X. Su and T. M. Khoshgoftaar. A survey of collaborative filtering techniques. *Advanc. in AI*, 2009.

[34] M. Taboada, J. Brooke, M. Tofiloski, K. Voll, and M. Stede. Lexicon-Based Methods for Sentiment Analysis. *Computational Linguastics*, 37(2), 2011.

[35] Y. Tan, Y. Zhang, M. Zhang, Y. Liu, and S. Ma. A Unified Framework for Emotional Elements Extraction based on Finite State Matching Machine. *NLPCC*, 400:60–71, 2013.

[36] M. Terzi, M. A. Ferrario, and J. Whittle. Free text in user reviews: Their role in recommender systems. *RecSys*, 2011.

[37] N. Tintarev and J. Masthoff. A Survey of Explanations in Recommender Systems. *ICDE*, 2007.

[38] N. Tintarev and J. Masthoff. Designing and Evaluating Explanations for Recommender Systems. *Recommender Systems Handbook*, pages 479–510, 2011.

[39] J. Vig, S. Sen, and J. Riedl. Tagsplanations: Explaining Recommendations Using Tags. *IUI*, 2009.

[40] J. Wiebe, T. Wilson, and C. Cardie. Annotating expressions of opinions and emotions in language. *LREC*, 2005.

[41] T. Wilson, J. Wiebe, et al. Recognizing contextual polarity in phrase-level sentiment analysis. *EMNLP*, 2005.

[42] A. Yessenalina, Y. Yue, et al. Multi-level structured models for document-level sentiment classification. *EMNLP*, 2010.

[43] Y. Zhang, H. Zhang, M. Zhang, Y. Liu, et al. Do Users Rate or Review? Boost Phrase-level Sentiment Labeling with Review-level Sentiment Classification. *SIGIR*, 2014.

[44] Y. Zhang, M. Zhang, Y. Liu, and S. Ma. Improve Collaborative Filtering Through Bordered Block Diagonal Form Matrices. *SIGIR*, 2013.

[45] Y. Zhang, M. Zhang, Y. Liu, S. Ma, and S. Feng. Localized Matrix Factorization for Recommendation based on Matrix Block Diagonal Forms. *WWW*, 2013.