Using Adjective Features from User Reviews to Generate Higher Quality and Explainable Recommendations

Xiaoying Xu, Anindya Datta, Kaushik Dutta

Using Adjective Features from User Reviews to Generate Higher Quality and Explainable Recommendations

Xiaoying Xu, Anindya Datta, Kaushik Dutta

Department of Information Systems, School of Computing, National University of Singapore
{xu1987, datta, dutta}@comp.nus.edu.sg

Abstract. Recommender systems have played a significant role in alleviating the "information overload" problem. Existing Collaborative Filtering approaches face the data sparsity problem and transparency problem, and the content-based approaches suffer the problem of insufficient attributes. In this paper, we show that abundant adjective features embedded in user reviews can be used to characterize movies as well as users' taste. We extend the standard TF-IDF term weighting scheme by introducing cluster frequency (CLF) to automatically extract high quality adjective features from user reviews for recommendation. We also develop a movie recommendation framework incorporating adjective features to generated highly accurate rating prediction and high quality recommendation explanation. The results of experiments performed on a real world dataset show that our proposed method outperforms the state-of-the-art techniques.

Keywords: Recommender systems, User reviews, Adjective Features, Sparsity, Transparency

1 Introduction

Recommender systems (*RS*) have been widely applied to alleviate the well-known "information overload" problem observed on e-commerce portals, and have experienced wide deployment at major technology companies like Amazon[1], TiVo[2] and Netflix[3]. Two broad flavors RS (and hybrids thereof) are employed in practice.

1. *Collaborative Filtering* (*CF*) *RS*, whose basic idea is to find a group of similar users who share the same tastes with the target user, and then recommend the items they like to the target user [1]. This is the most common form of RS used in practice.
2. *Content-based RS*, which tries to exploit descriptive attributes of items (such as directors of movies or authors of books) to perform recommendations [2].

[1] http://www.amazon.com
[2] http://www.tivo.com
[3] http://www.netflix.com

While CF has been used successfully, there are well known shortcomings. One of the major drawbacks of CF techniques is the problem that they heavily depend on user ratings. Unfortunately, in most domains studied (movies, books, restaurants etc.) a majority of items turn out to be unrated, resulting in *sparse* ratings matrices, which adversely impact the quality of recommendations [3, 4].

A second well-known, and equally significant, drawback of virtually all CF techniques is the lack of *transparency* [5]. These methods work as black boxes without offering the users much insight into the system logic or justification for the recommendation, which typically lower the users' trust on the recommendation produced [5, 6]. As has been demonstrated conclusively, trust is the most important attribute, which impacts users' willingness to act upon recommendations [7].

In a promising recent development, researchers are attempting to accommodate additional meta-content generated by users into the recommendation process. Driven by the popularity of social media, a bulk of this work factors in *user generated tags*, which are, typically, arbitrary words and short phrases provided by the users to label items in the system [8]. Compared to descriptive attributes typically used in content-based CF, tags cover more features of the items and are more comprehended by users. However, since tags are voluntarily and freely provided by the users, problems such as unwillingness to tag and diverging vocabulary can easily arise [2]. Recall that the sparsity of ratings is a challenge of rating-based recommendation, it turns out that the problem of sparsity is even worse in the tag space. Based on the extensive experiments we conducted, we can cover only 3.45% of items (movies, in our case) if we only use the tags to infer users' preference. Using ratings, coverage increases significantly. Clearly, while user generated meta-content shows promise, there exists substantial opportunity for further improvements.

The work reported in this paper is part of a project to create more effective recommender systems using user generated meta-content. The objective of this project was to design recommendation techniques that are (a) *substantially better in produced quality* and (b) *explainable*. We started off by exploring possible content to incorporate and immediately noticed an interesting phenomenon: a readily available source of "opinion" information for many consumer products (movies, books, hotels, electronic products, mobile apps) are user reviews and such reviews have been widely used for a variety of tasks in many application areas of data mining and information retrieval. Yet, the use of textual reviews in designing recommender systems has received scant attention from scientists. There do exist a few papers reporting the incorporation of free-text user reviews to perform recommendations, almost all of which employ opinion mining and summarization techniques to factorize user reviews and then to infer user preferences [9-11]. There is no reported work, in the recommendation area, that attempts to extract meta-information in addition to user ratings from reviews.

We note however, that a wealth of information is available from reviews that could possibly be used to enhance the recommendation process. In this paper we focus on one specific kind of information from user reviews, namely *adjective features*. While the intent is to incorporate various other types of data from reviews in future work, adjectives represent a particularly attractive feature to use in recommendations. At its core, recommendation engines suggest items that the user should *like* (and, clearly,

not *dislike*). When asked to reveal why people like or dislike something, they often use adjectives to explain their preference. For instance, when asked why she likes the movie *Titanic*, a user's answer often use words such as "romantic", "moving", "astounding", "beautiful" and "sad" – all adjectives. These features can be found in abundance in user reviews and remain unexplored in recommendation research.

Therefore, in this paper, we design a recommendation framework that incorporates adjective features extracted from external user reviews in addition to ratings to generate more accurate and more explainable movie recommendation, without the requirement for users to provide any tags. To automatically extract adjective features from user reviews, we employ well understood POS tagging methods. However, we quickly discover that a lot of adjectives are not helpful in discriminating tastes, i.e., they are too general to be adequately representative of users' tastes, e.g., "good", while some adjectives are too specific to capture the users' general taste aspects, e.g., "unsinkable" in the reviews of *Titanic*. To handle these problems, we extend traditional TF-IDF term weighting to TF-IDF-CLF by introducing another term weighting measure, called *cluster frequency* (CLF), to balance the *representativeness* and *generalizability* of the extracted features. Moreover, we adapt the concept of Regularized Singular Value Decomposition (RSVD) to construct item feature vectors and user feature vectors to generate more accurate rating prediction and more explainable recommendation by listing additional personalized item features.

The result of our work is a recommendation technique that makes substantial advances over extant techniques, like CF approaches and tag based systems. In particular, CF-based approaches achieve high coverage (almost 100%) at the cost of high prediction error (MAE is 0.69 - 0.75), whereas tag-based approaches achieve low prediction error (MAE is 0.67), they have extremely low coverage (9%). Our proposed method not only reduces the prediction error of the state-of-art CF algorithm and the state-of-art tag-based method by 11.27% and 8.77% respectively, but also achieves almost 100% coverage similar to CF-based approaches. Moreover, our approach provides high quality personalized recommendation explanation.

We believe this work to be important for four reasons.

1. Firstly, to the best of our knowledge, this is the first work to incorporate adjective features extracted from textual user reviews in recommendation.
2. Secondly, we extract better adjective features for the purpose of recommendation by introducing the notion of *Cluster frequency* (CLF), balancing the *representativeness* and *generalizability* of the extracted features, which has not been addressed by existing research, and contributes to higher prediction accuracy.
3. Thirdly, by using the abundant adjective features extracted from external user reviews, we relieve the tag sparsity problem and diverging vocabulary problem which have not been well solved in existing tag-based approaches.
4. Finally, we are one of the first recommendation techniques that provide full *transparency.* By decomposing the users' ratings into multi-dimensional vectors characterized by adjective features, we not only address the data sparsity problem, but also highlight the transparency of recommendation that is ignored by existing CF approaches.

The rest of this paper is organized as follows. Firstly we present our proposed recommendation framework including the detail description of each component. The remainder of the paper then presents the experiment and results. We review some related work before the conclusion and finally, we conclude by summarizing the paper, including the contribution, limitation and implication for future research.

2 Recommendation Framework

We now describe our proposed approach, starting off with the intuition behind it, and then proceeding to describe its details. While our approach is general and can be used to recommend any consumer item, we find that it is easier to explain ideas if we choose a specific example domain. Given that the most studied consumer domain in the recommendation context is movies, and the most results are available from the movie domain (to use in assessing the effectiveness of our approach), we will, henceforth, use the movie domain to present our technique. In other words, we will present our method, from this point forward, as a method to recommend movies to users.

2.1 Intuition and Overview

We are interested in predicting ratings for the movies that are new to the users, and recommending those movies with highest predicted ratings to them, together with reasonable and personalized explanations to improve the transparency of the recommendation logic. Noting that the limited number of descriptive attributes which are commonly used in content-based movie recommendation (e.g., actor, director) are not sufficient, we automatically extract adjective features from external user reviews (available in abundance in systems like IMDb[4], and Rotten Tomatoes[5]) to define distinguishing aspects of items and of the users' tastes, which are able to truly reflect the users' perception towards the movies in a higher and more abstract level. For example, the adjective features extracted from the user reviews of *Titanic* can be "romantic", "sad", "astounding". We predict the rating of *Titanic* for a user by estimating to what extent *Titanic* is romantic, is sad and is astounding, and how much this user likes romantic movies, sad movies and astounding movies.

2.2 Movie Recommendation Framework

The overview our movie recommendation framework is shown in Figure 1, where the shaded rectangles represent the components we have designed and implemented to realize our recommendation engine. There are five such components: review crawler, POS tagger, feature extractor, vector generator and recommender. More details of each component will be introduced in the ensuing sections of the paper.

---

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

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

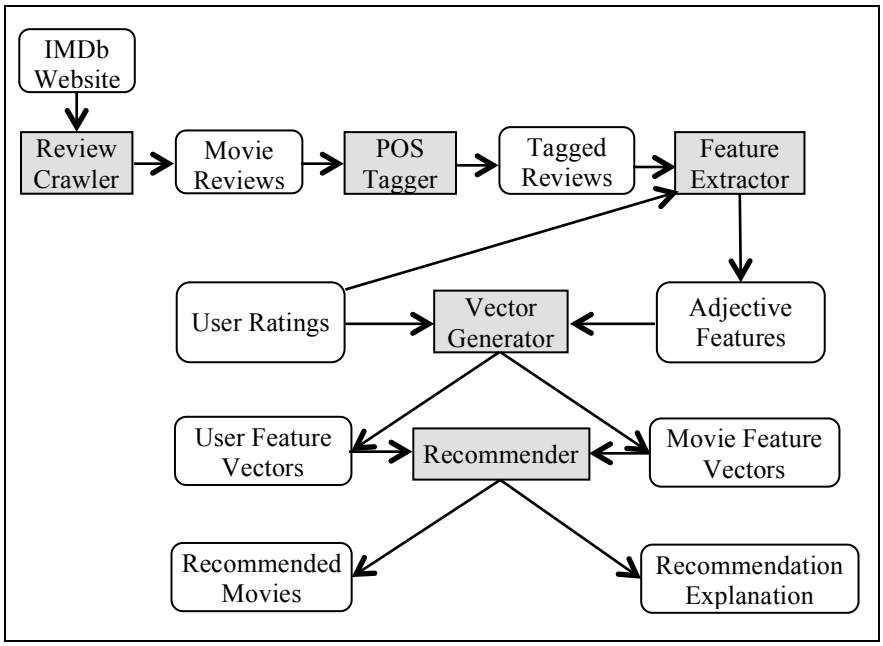

Fig. 1. Movie Recommendation Framework

Review Crawler. We obtain user reviews of the movies from a reputed external source, i.e. IMDb (the ***Internet Movie Database***). IMDb is one of the most popular online databases of movie information, with over 100 million unique users each month. IMDb also offers a platform for users to review movies, and allows other users to indicate whether they find certain review is useful. Figure 2 shows one user review of *Titanic* on IMDb website.

To obtain the reviews contents, we use a web crawler to collect the first 4 pages of user reviews (10 reviews per page) for each movie from the IMDb website, ranked by their usefulness, and then we extract the review contents from the webpages.

POS Tagger. After getting the user reviews for each movie, we employ the Stanford POS tagger [12] to assign parts of speech to each word in the reviews, such as noun, verb, adjective, etc. Since we intend to extract adjective features, we only keep the adjectives in the reviews. Taking the first paragraph of review in Figure 2 as an example, after the POS tagging, only the following words remain:

different good great boring cliché beautiful sad

Fig. 2. IMDb User Review Page

Feature Extractor. This component extracts adjective features from the tagged user reviews. Firstly, we need to assign a weight for each adjective term in the reviews. In the domain of information retrieval, term weighting is a measure of how important a word is in a document. The TF-IDF term weighting, which is a very commonly used term weighting scheme, for term t in document d is given by:

$$tf\text{-}idf_{t,d} = tf_{t,d} \times idf_t \quad . \tag{1}$$

where $tf_{t,d}$ is the frequency of term t in document d, and idf_t is the inverse document frequency of a term t, which indicates the term's discrimination power.

We regard the collection of all reviews of a movie as a document. The TF-IDF weighting for every word in the reviews can be easily obtained. While features extracted by TF-IDF weighting are able to represent movie characteristics, they are often tainted by two issues: (a) they may be too specific and might not serve as a generalizable, or common characteristic across similar movies, e.g. the word "unsinkable", which has very high TF-IDF scores in the reviews of *Titanic*, is too specific since we are unlikely to find other movies related to "unsinkable", thus it is not highly suitable to be used for representing the users' taste; and (b) they may not be able to extract some general features which are good for exposing users' taste aspects, e.g. when we are extracting features from the reviews of *Titanic*, the word "sad" may have high TF scores but low IDF scores therefore resulting in relatively low TF-IDF scores, however "sad" is a good feature since it accurately reflects a key perception of users towards this movie. In addition to *generalizability*, discussed above, the *representativeness* of the extracted features is also important, e.g. the word "good" is too general so that we cannot use it to represent the users' preference. In order to balance the *representative-*

ness and *generalizability*, we introduce another term weighting measure into TF-IDF, i.e. cluster frequency (CLF), to measure how common a word is among a cluster of documents which are similar to a particular document. We get a cluster of similar movies for a given movie, and accordingly, all reviews of the movies in the cluster will be used for calculating the CLF.

If we find a cluster of similar movies for *Titanic*, they may share some common characteristics of tragedy and the word "sad" would have high frequency among the reviews of this cluster. By introducing CLF, the term weighing of the word "sad" is higher and it is more likely to be extracted. Since we also give weight to TF-IDF weighting, those words that are too general (e.g. "good") will be filtered out.

Clearly, to realize the CLF idea, we need to group movies by "similarity". The similarity between movies can be computed by either of the following two approaches. First, we can adopt the idea of Amazon's item-based CF and use cosine similarity to compute the distance between two movies based on the users' co-rating patterns:

$$\cos(\vec{r_i}, \vec{r_j}) = \frac{\vec{r_i} \times \vec{r_j}}{\left\|\vec{r_i}\right\| \left\|\vec{r_j}\right\|} = \frac{\sum_{u \in U_{i,j}} r_{u,i} r_{u,j}}{\sqrt{\sum_{u \in U_i} r_{u,i}^2 \sum_{v \in U_j} r_{v,j}^2}} \quad . \tag{2}$$

where $U_{i,j}$ denotes the set of users rating both movie i and movie j, U_i denotes the set of users rating movie i and U_j denotes the set of users rating movie j. For each movie i, we select top M movies having highest cosine similarity scores as a group of similar movies.

Second, noticing that the item-based CF approach heavily depends on the user ratings that may not have good performance if the ratings are sparse, we also employ the Topic Modeling approach [13] which is purely based on the review contents and eliminates the dependency on user ratings. Latent Dirichlet Allocation (LDA) is a generative probabilistic model for collections of discrete data such as text corpora. Since we regard the collection of all reviews of a movie as a document, by applying LDA, each document corresponding to each movie can be represented as a multinomial distribution over latent topics, where each topic is characterized by a distribution over words. We apply Kullback–Leibler (KL) divergence, which is a non-symmetric measure of the difference between two probability distributions, to calculate the divergence from movie i's topic distribution, i.e. P_i, to movie j's topic distribution, i.e. P_j:

$$D_{KL}(P_i \parallel P_j) = \sum_{l \in LatentTopics} P_i(l) \ln \frac{P_i(l)}{P_j(l)} \quad . \tag{3}$$

where $P_i(l)$ denotes the probability that movie i belongs to the latent topic l. For each movie i, we select top M movies having smallest KL divergence as a cluster of similar movies.

After getting the cluster of similar movies for movie i, CLF weighting of term t in the reviews of i can be computed by counting how many movies in the cluster whose reviews contain term t. Finally, the integrated TF-IDF-CLF term weighting scheme is given by:

$$tf\text{-}idf\text{-}clf_{t,i} = tf_{t,i} \times idf_t \times clf_{t,i}^{\lambda_1} \quad . \tag{4}$$

where λ_1 is a parameter indicating how much weight is put in the CLF weighting. For each movie, the adjective features are extracted from its reviews by selecting the top *K* adjectives having the highest TF-IDF-CLF weightings, and then are passed to the vector generator.

Vector Generator. After getting the extracted features of each movie, we represent each movie as well as each user in the form of feature vector. Specifically, each movie *i* is represented as a vector Q_i, in which each element is associated with one of its features. The values of the elements measure the extent to which movie *i* possesses those features. Similarly, each user *u* is represented as a vector P_u and the elements are associated with the features of all movies. The values of the elements measure the extent to which user *u* likes those features. For example, let's assume that we have only two movies in the system, i.e. *Titanic* and *Spider-man*, and for each movie we extract 3 features from the user reviews, the movie feature vectors and the user feature vectors for two system users are as following:

Table 1. Movie Featrue Vector of *Titanic*

Feature	romantic	sad	astounding
Value	0.5	0.4	0.1

Table 2. Movie Feature Vector of *Spider-man*

Feature	romantic	spectacular	scary
Value	0.2	0.3	0.5

Table 3. User Feature Vectors

	romantic	sad	astounding	spectacular	scary
User A	0.4	0.5	-0.2	0.3	-0.1
User B	-0.1	0.1	0.5	0.3	0.4

The same as the latent factor model, we include the baseline predictors to estimate the non-interaction effects from users and movies respectively (i.e. $udev_u$ and $idev_i$). A predicted rating of movie *i* for user *u* is given by:

$$\hat{r}_{u,i} = \mu + udev_u + idev_i + \sum_{f \in F(i)} e_{u,f} e_{i,f} \quad . \tag{5}$$

μ denotes the overall average rating, $udev_u$ and $idev_i$ indicate the observed deviations of user *u* and item *i* respectively from the average. *F(i)* denotes the set of features belonging to movie *i*. $e_{u,f}$ is the value of feature *f* in user *u*'s feature vector P_u, and $e_{i,f}$ is the value of feature *f* in movie *i*'s feature vector Q_i.

We employ a stochastic gradient descent optimization adapted from RSVD, which was proposed by [14] and has been successfully applied by many others, to estimate

the values of the elements for both movie feature vectors and user feature vectors, as well as the baseline predictors. We iterate through the training data set until the prediction errors in the validation dataset are minimum.

Recommender. With the movie feature vectors and user feature vectors, we can easily predict a rating for a particular user given a movie that is new to him using formula (5). In order to recommend movies to a user, we can predict the ratings of all movies that are new to him, then rank these movies according to the predicted ratings, and recommend him the top *N* movies with highest predicted ratings.

One of the key features of our method is that in addition to providing recommendations, we provide explanation. We do this by explicitly listing features that caused an item to occur in the list of recommendations. For each movie *i* in user *u*'s recommended list, the product of two feature values $e_{u,f} \cdot e_{i,f}$ is the partial interaction effect regarding feature *f*, and measures the extent to which feature *f* contributes to cause movie *i* being recommended to user *u*. Therefore, we rank all the features in $F(i)$ according to the partial interaction effect, and provide the top *K* features having highest products in addition to the recommended movie *i,* as an explanation of recommendation. Using the above-mentioned example, if the movie *Spider-man* is recommended to user *B*, then we can get the partial interaction effect regarding each feature of *Spider-man*, as following:

Table 4. Partial Interaction Effect

	romantic	spectacular	scary
$e_{B,f}$	-0.1	0.3	0.4
$e_{Spider-man,f}$	0.2	0.3	0.5
$e_{B,f} \times e_{Spider-man,f}$	-0.02	0.09	0.2

If we only provide the top one feature as the explanation to user *B* for recommending this movie, the feature "scary" having the highest partial interaction effect is selected.

Since the values of features are different in different users' feature vectors depending on their preference on these features, even we recommend the same movie to two different users, the explanation should be different. Thus*, our explanation of recommendation is personalized, truly reflecting the users' tastes*.

3 Experiment and Results

All recommender systems largely rely on their *prediction engines*, which are able to predict users' preference over items. There is a basic assumption that users prefer recommender system that provides "accurate predictions"[15]. Therefore, most research in recommender systems, seek to develop algorithms that provide better prediction, and evaluate recommender systems by measuring their prediction accuracies. In line with this, we use the *Mean Absolute Error* (MAE) metric that is commonly

used in recommendation research to evaluate the accuracy of rating prediction. MAE is defined as:

$$MAE = \frac{\sum_{r_{u,i} \in TestingSet} \left| r_{u,i} - \hat{r}_{u,i} \right|}{\left| TestingSet \right|} . \quad (6)$$

where $r_{u,i}$ is the rating given by user u to item i in the testing dataset, $\hat{r}_{u,i}$ is the predicted rating and $|TestingSet|$ is the size of testing dataset.

While accurate prediction is crucial, it still does not address one key goal of good recommender systems, to be able to cover a wide range of items. In other words, an algorithm may have high prediction power, but if it is only able to work over a small portion of items it is not a desirable situation. Accordingly, researchers also measure the *coverage* of recommender systems, which refers to the proportion of items that the system can recommend, indicating the ability of exploring diverse items. We use the percentage of items in the testing set of which users' preference can be inferred, in terms of either probabilities or predicted ratings:

$$Coverage = \frac{\#Predicatble\ Items\ in\ TestingSet}{\left| TestingSet \right|} \times 100\% . \quad (7)$$

In our experiment, we will evaluate our proposed method in terms of both rating prediction accuracy and item space coverage. We will also provide the qualitative results of recommendation explanation.

3.1 Experiment Setup

We use two kinds of data in our experiment. Firstly, we use the publicly available Movielens 10M version dataset which consists of 10 million ratings and 95580 tags applied to 10681 movies by 71567 users of the online movie recommender service MovieLens. The user ratings are on a scale of 1 to 5, with 1 being bad and 5 being excellent. In the original dataset, only 4009 (5.60%) users provide tags to movies, and only 7601 (71.16%) movies receive tags from users. We preprocess the data to generate a tag-dense subset exactly the same as the one used by Wei, et al. [8] for the purpose of comparing results. In the subset, every user gives at least one tag to movies, and every movie receives at least one tag from users. Let <user,movie,tag,rating> denote a tuple. The subset has 1112 tuples with 201 users, 501 movies and 404 tags. Secondly, we crawl the first 40 user reviews from IMDb website for each movie in the dataset, ranked by the number of users who indicate that the review is useful. 99.61% of the movies in the dataset have user reviews in IMDb.

To evaluate the performance of our proposed method, we will compare both the rating prediction accuracy and the item space coverage among the following methods:

1. **User-based CF (UCF)** [1]. This method uses the traditional user-based collaborative filtering.

2. **Item-based CF (ICF)** [16]. This method is the Amazon's item-based collaborative filtering.
3. **Tag Bayesian Network (TBN)**. This method generates recommendation based on the ternary relationship among users-items-tags using Bayesian Network.
4. **Probabilistic Matrix Factorization (PMF)** [17]. This is a state-of-art rating-based collaborative filtering algorithm that utilizes the relationship among users, items and ratings.
5. **Quaternary Sematic Analysis (QSA)** [8]. This method represents the state-of-art tag-based approaches. It models the quaternary relationship of users-items-tags-ratings as 4-order tensor and performs the multi-way latent semantic analysis.
6. **Adjective Feature Vector (AFV).** This is our proposed method using adjective features extracted from external user reviews. Since we use two approaches to get the cluster of similar movies, our method has two variants: *AFV using rating-based similarity* (AFV-R) and *AFV using review-topic-based similarity* (AFV-T).

3.2 Results of Item Rating Prediction

We first evaluate the item rating prediction performance of our proposed AFV-R and AFV-T methods. We randomly split the dataset into five subsets and each time use one subset as testing set and the remaining subsets as training set (aka. five-fold validation), and compute the MAE for different methods except for TBN, since this method estimates the probability that a user will like a particular item instead of predicting rating. We vary the parameters of our method to find the settings that give the best performance. This occurs when cluster size M=20, feature number K=110, λ_1=2, λ_2=25, λ_3=10, λ_4=0.02, *lrate*=0.005, s=0.1, and the times of iteration are 200. The results are shown in Figure 3.

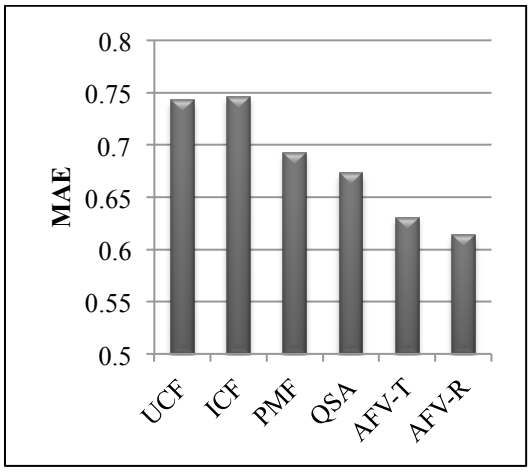

Fig. 3. Rating Prediction Accuracy (Lower MAE indicates higher accuracy)

To analyze the statistical significance of the results, we conduct a one tail t-test. Table 5 shows the p-value of comparing the results of AFV-T and AFV-R with other methods respectively. The results indicate that both of our AFV-T and AFV-R methods significantly outperform the existing methods. Specifically, AFV-T reduces the prediction error of the state-of-art CF algorithm and the state-of-art tag-based method by 8.90% and 6.33% respectively, and AFV-R reduces the prediction error of the two state-of-art algorithms by 11.27% and 8.77% respectively.

Table 5. P-value of Accuracy Significance Test

	P-value(AFV-T)	P-value(AFV-R)
BCF	0.0000	0.0000
ICF	0.0000	0.0000
PMF	0.0010	0.0005
QSA	0.0040	0.0010

3.3 Results of Recommendation Coverage

In addition to item rating prediction accuracy, we also compare the coverage of our methods with other methods. The results on the tag-dense subset are shown in Figure 4. PMF, QSA, and AFV-T achieve 100% coverage and the two traditional CF approaches also achieve high coverage. Since we apply ICF in AFV-R, the coverage of AFV-R is the same as ICF. However, coverage of TBN, which is purely based on tags, is only 58%, even if the data subset is tag-dense. Although the QSA method achieves 100% coverage, it is not the case in reality, since it requires every user provides tags and every movie receives tags. Actually, in the original dataset, only 5.60% of users provide tags to movies, and only 71.16% of movies receive tags from users.

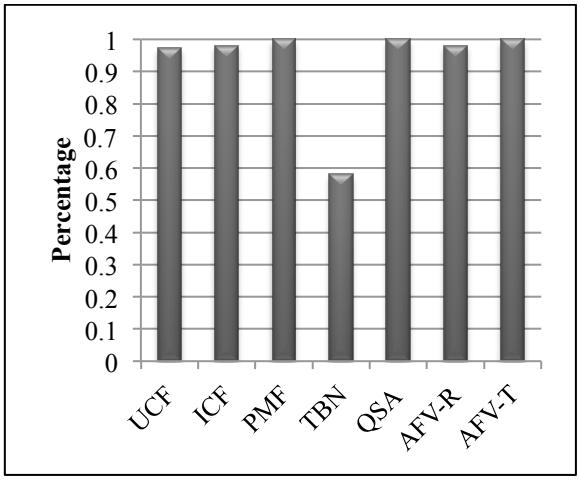

Fig. 4. Coverage on Tag-dense Subset

We also evaluate the coverage of different methods in the full dataset, and the results are shown in Figure 5. In the full dataset, QSA only achieves 9.07% coverage and TBN is even worse, which is only 3.45%. Since the proposed AFV-T and AFV-R methods use the external reviews and do not require any tags from user, they achieve high coverage. Specifically, the coverage of AFV-R is the same as ICF, which is 96.4%, and the coverage of AFV-T is independent of user ratings but is determined by the proportion of movies having user reviews, which is close to 100%.

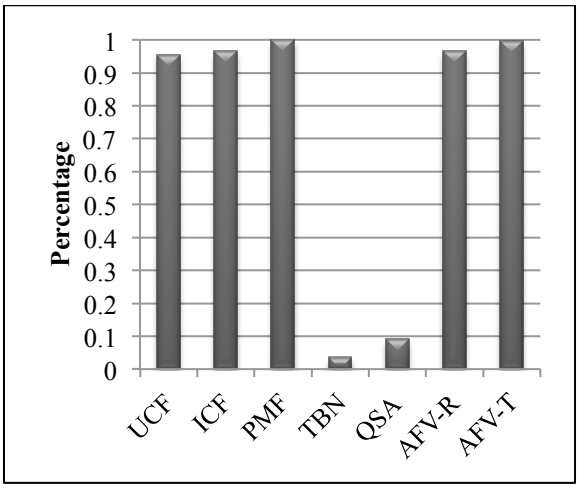

Fig. 5. Coverage on Full Dataset

3.4 Results of Recommendation Explanation

The transparency of recommender system is often ignored by most CF approaches, while our method is able to provide explanation of recommendation for the system users. We are going to show the qualitative results of recommendation explanation using our method.

Applying the proposed AFV-R method which has the highest prediction accuracy, we recommend 5 movies for each user. We arbitrarily select three users who have 2 common movies in their recommendation list to illustrate the qualitative results of recommendation explanation given by our method. The two common movies are *My Neighbor Totoro* and *Grave of the Fireflies*. For each user, Table 6 shows the 8 listed features for each movie, as the explanation of recommending this movie to him.

As is shown by the results, for different users, the explanation of recommending the same movie is personalized, taking both the movie features and the users' tastes on each feature into consideration. In addition to providing the recommended movie, the explanation gains the users more insights into the recommendation mechanism, and therefore makes the recommendation more trustful and more acceptable to the users.

Table 6. Recommendation Explanation

	My Neighbor Totoro	*Grave of the Fireflies*
User A	curious, suitable, imaginative, warm, magical, friendly, sentimental, sweet	magical, suitable, cold, gorgeous, sentimental, beautiful, happy, extraordinary
User B	endearing, suitable, imaginative, giant, cute, poetic, boundless, fantastical	giant, astonished, live, engrossing, animated, suitable, poetic, gentle
User C	lovely, delightful, happy, gentle, spectacular, engaged, curious, magic	lovely, gentle, happy, magical, afraid, heartfelt, cold, engrossing

4 Related Work

There have been a large amount of recommendation research, most of which belongs to two main streams, i.e. CF approach and content-based approach. Our proposed method adopts the concept of SVD from CF approach into content-based approach to address the data sparsity problem and transparency problem. We also extend the existing automatic tag generation methods to extract adjective features from external user reviews for the purpose of recommendation. In the following, we review the related work to our proposed method.

4.1 Collaborative Filtering (CF) Recommendation

Neighborhood Models. Owing to its compelling simplicity and good quality of recommendations, CF has been deeply explored in the last ten years and represents the most commonly used recommendation technique. The most common approaches to CF are based on neighborhood models. Typically, these approaches can be classified into two types, i.e. user-based [1] and item-based [16]. Both the user-based and item-based CF works on the same basic principal. First a distance (typically cosine distance) is measured between users or items based on which user liked which item. Next, this distance is used to measure the similarity between two items or two users.

Latent Factor Models. Latent factor models such as probabilistic matrix factorization (PMF) provide an alternative approach to CF by mapping both items and users to the same latent factor space, which is automatically inferred from user feedback and tries to explain ratings by characterizing both users and items on the latent factors. Singular Value Decomposition (SVD) [18] is a well-known method for matrix factorization that provides the best lower rank approximations of the original matrix.

Transparency in Recommender Systems. According to a rule stem from psychology, people are generally more comfortable with what they are familiar with and understand [7]. However, most of the collaborative filtering techniques seem to com-

pletely ignore this important rule and work as the black boxes. HCI research on recommender systems highlights the role that transparency plays in recommendation. Rashmi et al. [5] showed that both mean liking and mean confidence were significantly increased in a recommender system which was more transparent. They argued that from the users' perspective, a good algorithm that generates accurate recommendations is not enough to constitute a useful system. The system also needs to convey its inner logic and why a particular recommendation is suitable to the user. Henriette et al. [6] conducted an experiment and the results also showed that acceptance of the recommendations was increased by explaining to the user why a recommendation was made. To summarize, transparency in recommender systems is an important issue that cannot be ignored. To address the shortcoming of the black box approaches, our proposed approach incorporates the characteristics of content-based recommendation to offer explanation for recommendation.

4.2 Content-based Recommendation

The intuition of content-based recommendation is to learn to recommend items that are similar to the ones that the user liked in the past. The similarity between items is calculated based on the features associated with the compared items. For example, if a user has positively rated a movie that belongs to the comedy genre, then the system can learn to recommend other movies from this genre. Content-based techniques have a natural limit in the number and type of features that are associated with the items that these systems recommend.

Recently, due to the popularity of social tagging systems, tags as a specific kind of user-generated contents have drawn attention from researchers. Tags are generated by users who collaboratively annotate and categorize resources of interests with freely chosen keywords [19]. Compared to descriptive attributes, tags cover more features of the items and are more comprehended by the users. Wei et al. [8] proposed a unified framework for recommendation by modeling the quaternary relationship among users, items, tags and ratings as a 4-order tensor and performed a multi-way latent semantic analysis, representing the state-of-the-art.

4.3 Automatic Tag Generation

Researches on automatic tag generation stem from information retrieval and text mining. There are two streams of such researches. The first stream tries to suggest relevant tags for new document based on existing tags (aka. tag assignment). In practice, tags are chosen from a controlled vocabulary of terms, and documents are classified according to their content into classes that correspond to elements of the vocabulary. Tags generated by these approaches have high generalizability and they can well identify the common features of the similar documents. However, the representativeness of these tags may be weak so that it is hard to differentiate the documents.

Another stream of work aims at automatically extracting tags from textual documents on the basis of properties such as frequency, without any predefined vocabulary (aka. tag extraction). Many approaches can be used in tag extraction, like term fre-

quency, TF-IDF, etc. Tags given by extraction methods are able to highly represent the associated documents; however, they face the problem of diverging vocabulary and have difficulty in exposing the common features among similar documents. For the purpose of recommendation, it is important to balance the representativeness and generalizability of the extracted features, which is addressed in our approach.

5 Conclusion

In this work, we have shown that adjective features embedded in user reviews can be used to characterize movies as well as users' taste. We employ POS tagging and propose introducing cluster frequency into traditional TF-IDF term weight scheme to extract adjective features from external user reviews, alleviating the problem diverging vocabulary, and balancing the representativeness and generalizability of the extracted features. We also propose a novel method which adapts stochastic gradient descent optimization from RSVD to construct item feature vectors and user feature vectors. The experiment results show that the proposed method outperforms the state-of-art CF algorithm and state-of-art tag-based method in terms of rating prediction accuracy. Specifically, our proposed R-AFV method reduces the prediction error of the state-of-art CF algorithm and the state-of-art tag-based method by 11.27% and 8.77% respectively. Our proposed method does not require any tags from users. On the one hand, it relieves the users' efforts to provide tags; on the other hand, it can always achieve high coverage of recommendation while the coverage of tag-based methods is extremely low when the tags are sparse, which is always the case in reality. Moreover, the proposed method is able to provide personalized explanation of recommendation for the users, increasing the users' trust on the recommendation.

There are some limitations of our work. Firstly, we only consider the adjective features and ignore other descriptive attributes of movies. Our method can be extended to incorporate other descriptive attributes to generate more accurate recommendation and higher quality of explanation. Secondly, we do not consider the semantic relationship between the adjective features. Future work can focus on the semantic analysis of the adjective features.

Although our recommendation framework focuses on the movie domain, it can be easily applied in other domains where user reviews are available. Since our method captures the users' taste in a higher and more abstract level, it can be also applied in cross-domain recommendation.

References

1. Resnick, P., Iacovou, N., Suchak, M., Bergstorm, P., Riedl, J.: GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In: Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work, pp. 175-186. ACM, (1994)
2. Lops, P., Gemmis, M., Semeraro, G.: Content-based Recommender Systems: State of the Art and Trends. In: Ricci, F., Rokach, L., Shapira, B. (eds.) Recommender Systems Handbook, pp. 73-105. Springer (2011)

3. Su, X., Khoshgoftaar, T.M.: A survey of collaborative filtering techniques. Advances in Artificial Intelligence 2009, 4 (2009)
4. Adomavicius, G., Tuzhilin, A.: Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. Knowledge and Data Engineering, IEEE Transactions on 17, 734-749 (2005)
5. Sinha, R., Swearingen, K.: The role of transparency in recommender systems. In: CHI '02: Extended Abstracts on Human Factors in Computing Systems, pp. 830-831. ACM, (2002)
6. Cramer, H., Evers, V., Ramlal, S., van Someren, M., Rutledge, L., Stash, N., Aroyo, L., Wielinga, B.: The effects of transparency on trust in and acceptance of a content-based art recommender. User Modeling and User-Adapted Interaction 18, 455-496 (2008)
7. Massa, P., Avesani, P.: Trust Metrics in Recommender Systems. In: Karat, J., Vanderdonckt, J., Golbeck, J. (eds.) Computing with Social Trust, pp. 259-285. Springer London (2009)
8. Wei, C., Hsu, W., Lee, M.: A unified framework for recommendations based on quaternary semantic analysis. In: Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval, pp. 1023-1032. ACM, (2011)
9. Jakob, N., Weber, S.H., M, M.C., Gurevych, I.: Beyond the stars: exploiting free-text user reviews to improve the accuracy of movie recommendations. Proceedings of the 1st international CIKM workshop on Topic-sentiment analysis for mass opinion, pp. 57-64. ACM, Hong Kong, China (2009)
10. Faridani, S.: Using canonical correlation analysis for generalized sentiment analysis, product recommendation and search. In: Proceedings of the fifth ACM conference on Recommender systems, pp. 355-358. ACM, (2011)
11. Leung, C.W.K., Chan, S.C.F., Chung, F.: Integrating Collaborative Filtering and Sentiment Analysis: A Rating Inference Approach. In: Proceedings of The ECAI 2006 Workshop on Recommender Systems, pp. 62-66. (2006)
12. Toutanova, K., Klein, D., Manning, C.D., Singer, Y.: Feature-rich part-of-speech tagging with a cyclic dependency network. Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1, pp. 173-180. Association for Computational Linguistics, Edmonton, Canada (2003)
13. Blei, D., Ng, A., Jordan, M.: Latent Dirichlet Allocation. Journal of Machine Learning Research 3, 993-1022 (2003)
14. http://sifter.org/simon/journal/20061211.html
15. Shani, G., Gunawardana, A.: Evaluating Recommendation Systems. In: Ricci, F., Rokach, L., Shapira, B. (eds.) Recommender Systems Handbook, pp. 73-105. Springer (2011)
16. Sarwar, B., Karypis, G., Konstan, J., Reidl, J.: Item-based collaborative filtering recommendation algorithms. In: World Wide Web, pp. 285-295. (2001)
17. Salakhutdinov, R., Mnih, A.: Probabilistic matrix factorization. Advances in neural information processing systems 20, 1257-1264 (2008)
18. Takács, G., Pilászy, I., Németh, B., Tikk, D.: Matrix factorization and neighbor based algorithms for the netflix prize problem. Proceedings of the 2008 ACM conference on Recommender systems, pp. 267-274. ACM, Lausanne, Switzerland (2008)
19. de Gemmis, M., Lops, P., Semeraro, G., Basile, P.: Integrating tags in a semantic content-based recommender. In: Proceedings of the 2008 ACM conference on Recommender systems, pp. 163-170. ACM, (2008)