Electronic Commerce Research and Applications 12 (2013) 14–23

Contents lists available at ScienceDirect

Electronic Commerce Research and Applications

journal homepage: www.elsevier.com/locate/ecra

Combining user preferences and user opinions for accurate recommendation

Hongyan Liu [a], Jun He [b,c,*], Tingting Wang [b,c], Wenting Song [b,c], Xiaoyang Du [b,c]

[a] Research Center for Contemporary Management, Key Research Institute of Humanities and Social Sciences at Universities, School of Economics and Management, Tsinghua University, Beijing 100084, China
[b] Key Labs of Data Engineering and Knowledge Engineering, Ministry of Education, China
[c] School of Information, Renmin University of China, Beijing 100872, China

ARTICLE INFO

Article history:
Received 8 June 2011
Received in revised form 12 May 2012
Accepted 12 May 2012
Available online 25 May 2012

Keywords:
Feature extraction
Opinion mining
Recommendation systems

ABSTRACT

Recommendation systems represent a popular research area with a variety of applications. Such systems provide personalized services to the user and help address the problem of information overload. Traditional recommendation methods such as collaborative filtering suffer from low accuracy because of data sparseness though. We propose a novel recommendation algorithm based on analysis of an online review. The algorithm incorporates two new methods for opinion mining and recommendation. As opposed to traditional methods, which are usually based on the similarity of ratings to infer user preferences, the proposed recommendation method analyzes the difference between the ratings and opinions of the user to identify the user's preferences. This method considers explicit ratings and implicit opinions, an action that can address the problem of data sparseness. We propose a new feature and opinion extraction method based on the characteristics of online reviews to extract effectively the opinion of the user from a customer review written in Chinese. Based on these methods, we also conduct an empirical study of online restaurant customer reviews to create a restaurant recommendation system and demonstrate the effectiveness of the proposed methods.

1. Introduction

Given the development and popularization of the Internet, the amount of available information has increased dramatically, resulting in the problem of information overload. A personalized recommendation system (Adomavicius and Tuzhilin 2005) is one effective way to solve this problem and has been used in many applications (Ali and Van Stam 2004, Bennett and Lanningm 2007, Goldberg et al. 2001, Linden et al. 2003). However, the sparseness of a data set makes many recommendation methods such as collaborative filtering suffer from low accuracy. Collaborative filtering (Breese et al. 1998, Resnick et al. 1994) and multi-criteria recommendation methods (Adomavicius and Kwon 2007, Adomavicius et al. 2010, Tang and McCalla 2009) are usually based on commonality among customers. Similar users or items are found by measuring the similarity of the common rating scores of users. However, the insufficiency of data makes this method unsatisfactory, because the number of items rated by both users is usually very small.

We propose a very different method to infer the preference of the user, and this method is called *preference and opinion*-based recommendation. This method recommends items based on the preferences of the user on each feature of an item, and the preferences are inferred based on the difference of the opinions of the user from others' opinions. The concept behind this method is that if a user frequently gives a lower score about a feature than other users after eliminating the effect of the rating habit of the user, he may have a higher requirement for this feature. Thus, the features and opinions for which users expressed their comments (feature-opinion extraction problem) need to be extracted, and the polarity of the opinions (sentiment analysis problem) should be analyzed. Effective measures to describe the preference of the user and a design recommendation strategy (recommendation problem) based on the preferred features of the user also need to be developed.

Therefore, we will focus on two problems: the feature-opinion extraction problem and the recommendation problem. We discuss these issues based on online restaurant reviews written in Chinese. Existing studies on the feature-opinion extraction problem were mainly conducted on reviews written in English (Ding and Liu 2007, Hu and Liu 2004) and other languages such as Japanese (Kobayashi et al. 2006, Morinaga et al. 2002). Little work has been done on Chinese reviews.

The contributions of this paper are as follows. (1) We propose a new method to extract features and opinions from online user reviews. The method is called the adverb-based opinion-feature extraction method, which uses Chinese expression characteristics to enhance extraction accuracy. (2) We develop two novel measures, *concern* and *requirement*, to capture the preference of the user toward features of an item. Based on these measures, we design a recommendation method. (3) We propose an algorithm called PORE to incorporate these above-mentioned methods for

* Corresponding author.
E-mail address: hejun@ruc.edu.cn (J. He).

1567-4223/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.elerap.2012.05.002

online review-based recommendation. We also conduct an empirical study to evaluate the performance of the algorithm. Experimental results obtained from real online reviews show that the methods proposed are effective in dealing with insufficient data and are more accurate and efficient than existing traditional methods.

The rest of the paper is organized as follows. In Section 2, we formally define two research problem, and introduce some definitions and notation. The feature-opinion extraction method and the proposed recommendation method are described in Sections 3 and 4. The performance of the proposed methods is evaluated in Section 5. Related work is discussed in Section 6. We conclude in Section 7.

2. Problem definition

An online review is about an *object*, such as a product like a camera, or about a service, such as that of a restaurant. Each object is called an *item*. We consider online restaurant reviews, and each restaurant that commented by a user is an item. Suppose we have a collection of M reviews, $D = \{d_1, d_2, \ldots, d_M\}$, about a certain kind of item. Let U be the set of N users who wrote these reviews, $U = \{u_1, u_2, \ldots, u_N\}$, and R be the set of K items users commented on, $R = \{r_1, r_2, \ldots, r_K\}$. Let F be a set of L features users comment on, $F = \{f_1, f_2, \ldots, f_L\}$, and let P be a set of Q opinions that users use to express their feelings, $P = \{p_1, p_2, \ldots, p_Q\}$. The definitions of the terms *feature* and *opinion* are given below (Aggarwal and Zhai 2012).

Definition 1 (*Feature*). Each object is called an item. Item e is associated with a set of features, each of which is an attribute or any component or sub-component of the item.

For example, for a review of one restaurant (the restaurant is an item), users often comment about its price, environment, taste, and service, each of which is called a *feature*. For a review about a certain product such as a camera, a component like the camera lens or an attribute like the price are all features of the camera.

Definition 2 (*Opinion*). An opinion is represented by words that express a positive or negative sentiment, attitude, emotion, or appraisal regarding a feature of an item in a review sentence.

For example, in the review sentence "The taste is very good," "taste" is a feature and "good" is the opinion word. In "I like the service of this restaurant," "service" is a feature, and "like" is the opinion word.

Definition 3 (*Explicit and implicit features*). If a feature f is explicitly presented as the description of the object in the review sentence, it is called an explicit feature. If a feature f is not shown in the sentences, but is implied by some opinion words, the feature is called an implicit feature.

For example, in the review sentence "It is too expensive," "expensive" implies that the feature described here relates to "price." Therefore, "price" is an implicit feature. However, if the review sentence is "The price is pretty high," the feature is considered to be an explicit feature.

Definition 4 (*Feature-opinion pair*). A feature and an opinion expressed for the feature in a review sentence form a feature-opinion pair.

Definition 5 (*Polarity*). The polarity of a feature-opinion pair is the sentiment orientation of the opinion, which can be positive or negative.

We use 1 to represent positive and −1 to represent negative. To further consider the intensity of the opinion, positive can be weighted from 0.8, 1, to 1.2; and negative can be weighted from −1.2, −1, to −0.8. For example, slightly good is 0.8, good is 1, and very good is 1.2. Some opinions have a fixed polarity independent of their context. These are referred to as *context-independent* opinions. Others have a different polarity with a different context, and these are called *context-dependent* opinions. For example, "good" is a context-independent opinion, and "high" is a context-dependent opinion. The polarity of *high* quality is positive, and *high* price is negative.

Definition 6 (*Support of feature*). Given a collection of reviews D and a feature $f \in F$, the support of the feature, denoted by support (f), is defined as the number of reviews in D that contain a user opinion about feature f.

Definition 7 (*Feature value*). Given a user $u_i \in U$ and a feature $f_j \in F$, if the user gives a score to the feature f_j for item $r_k \in \mathrm{R}$ in a particular review, the feature value, S_{ijk}, is defined as the score. If the user comments on the feature f_j of item r_k through an opinion $p \in P$ in a review sentence, the feature value S_{ijk} is defined as the polarity of opinion p. If a user commented on an item feature f_j of item r_k more than once at different times, the latest one is included and the old ones are disregarded.

Based on these definitions and notations, the two mining problems studied in this paper are described as follows.

2.1. Feature-opinion extraction task

Given a *collection D* of M reviews about a certain kind of item, $D = \{d_1, d_2, \ldots, d_M\}$ written by a set of N users $U = \{u_1, u_2, \ldots, u_N\}$, feature and opinion pairs are extracted from each review sentence. The output of this task is a collection of quadruples (user, item, feature-opinion pair, sentence ID).

Based on the output of opinion-feature extraction, the *sentiment analysis technique* is used to determine the polarity of each feature opinion pair. Many algorithms can be used. An existing one is chosen, and its details are omitted because of space limitation. Afterwards, each of the quadruple becomes (user, item, feature-opinion pair, polarity). Together with the scores given by users in each review, a collection of quadruples (user, item, feature, feature value) is obtained.

2.2. Item recommendation task

Given a collection of quadruples (user, item, feature, feature value) output by the feature-opinion extraction module for the review collection D, the recommendation task is to recommend k items to a user based on the reviews of the user in D.

3. Opinion-feature extraction

Existing feature opinion extraction methods can be classified into two types, namely, *supervised learning* (Liu et al. 2005, Ghani et al. 2006) and unsupervised learning (Hu and Liu 2004, Popescu and Etzioni 2005). We focus on *unsupervised learning*, which is domain-independent and also saves time spent manually labeling training data sets.

Existing popular feature opinion extraction methods (Hu and Liu 2004, Liu et al. 2005) first find features and then make use of them to obtain the opinions around the features. Features are found based on *association rule mining*. *Part-of-speech* (POS) *tagging* is performed for each review. Each review sentence is transformed into a *transaction line*, which contains a noun or noun phrases.

Then, association miner CBA (Liu et al. 1998) is used to find frequent *itemsets* (minimum support threshold is 1%). Each itemset is regarded as a candidate feature, and two kinds of pruning are then conducted. *Compactness pruning* removes those features whose words do not appear together in a specific order, and *redundancy pruning* removes single-word redundant features. For those sentences containing frequent features, adjective words are extracted from the sentences as opinions. Finally, infrequent features are found by extracting the nearest noun or noun phrases around the opinions. Thus, opinion words that have no features present around them cannot be found, and implicit features are difficult to find (Popescu and Etzioni 2005). Moreover, some frequently occurring noun phrases that are not features can also be mistaken as features, because sentences that do not contain any opinion are also considered.

We downloaded 54,208 reviews from a review website (beijing.koubei.com) to study the characteristic of reviews written in Chinese. Each review was split into sentences according to punctuation, thereby obtaining 555,334 sentences. We then manually categorized the sentences into two classes, subjective and objective. *Subjective sentences* contain user opinions, whereas *objective sentences* ones do not. Of the 555,334 sentences, 277,972, or 55%, were subjective sentences; objective sentences accounted for 45% of the total. The subjective sentences were further divided into two types. As shown in Fig. 1a, the first type is the *feature + opinion pattern*, which accounts for 93%, and the second type is *verb + feature pattern*, where the verb is used to express emotion and is called an *emotion verb*. For example, the sentence "The taste is very good" belongs to Type 1, and "I like the taste there" belongs to Type 2. For Type 2 sentences, features can be extracted from them using a list of emotion verbs. Therefore, our focus was on dealing with a Type 1 sentence. Among all the subjective sentences, 128,514 sentences, or 46%, satisfy the *adverb + adjective pattern*, as shown in Fig. 1b. That is to say, Chinese review writers are used to expressing their opinion with adverbs before adjectives. The number of adverbs frequently used was small compared with the number of nouns, verbs, or adjectives. Moreover, the list of adverbs did not change significantly across different domains. Based on this observation, we developed a new method for feature-opinion extraction to take advantage of these characteristics.

Most existing feature-opinion extraction algorithms can be regarded as a feature-based opinion extraction method. In contrast to this method, our new method first extracts opinions according to domain-independent adverbs, and then extracts features nearby based on these opinions. Recursively, extracted features can contribute to more new opinions, and so on. This method can improve the precision and recall of feature and opinion extraction from Chinese reviews. It also enables the discovery of opinions without having features paired with them and can eliminate some non-review sentences from further analysis.

Next, we describe how to extract opinion words, how to do an opinion and feature mutual extraction, and how to merge synonym features.

3.1. Opinion word extraction

Extracting opinions is challenging, because opinions may vary a lot in reviews of different domains. For example, opinion words used in car reviews are very different from those used in cosmetic product reviews. However, as mentioned previously, when people express their opinions in Chinese, especially in spoken Chinese, they not only express their emotional tendency, but also express the degree to which they feel using adverbs. For example, in the sentence "It is very nice," an adverb "very" comes before the opinion word "nice." Furthermore, the adverbs are domain-independent words compared with the opinion words; thus, domain-dependent opinion words are extracted based on domain-independent adverb words.

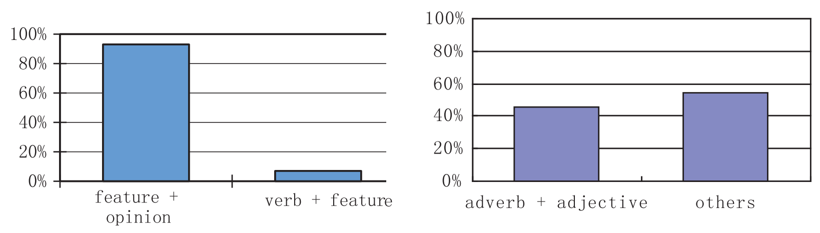
Based on this observation, we establish a rule for opinion extraction: *if an adjective follows an adverb, then it is an opinion word*. An experiment conducted on randomly selected 4000 reviews shows that the false positive rate of this rule is about 4.3%. According to this rule, during opinion extraction and given a set of adverbs, *adverbList*, we assign every adjective following an adverb into an opinion set, *P*. Simultaneously, we assign the feature-opinion tuple *(user_id, item_id, sentence_id, feature,adj*) with empty feature currently into a global feature-opinion pair set (FOP). According to opinions already found, we can then expand the adverb set. The two steps can go on until no new adverb and opinion can be found.

Before extracting features and opinions, we use word segmentation and POS tagging software *ICTCLAS* 3.0 proposed by the Chinese Academy of Sciences (Zhang et al. 2010) to preprocess each review in *D*. In the experiments, we obtain the seed adverbs from the Web (Dong and Dong 2010).

3.2. Opinion and feature mutual extraction

With the initial opinion set extracted based on adverbs, we then iteratively find features and new opinions. As explicit features are usually nouns or noun phrases, each *n*-gram centered at a noun within an opinion window is added to the candidate feature set, and the corresponding quintuple in *FOP* are updated. Then, the adjective within a feature window is also extracted and updated to *FOP*. These two steps are repeated until both the feature set and opinion set do not expand any more. Lastly, we use a user-specified threshold of minimum support to prune the features with small occurrences.

We extract the feature and opinion from each review sentence (each clause separated by punctuation is called a sentence in this paper). The distance between two neighbor words is 1. For the opinion window in a sentence, we view the opinion as the center point, and each noun or noun phrase with a maximum of *n* words in the opinion window and with a distance to the center less than 5 is extracted. The meaning of the feature window is similar to that of the opinion window. Algorithm 1, the opinion and feature extraction algorithm, *OpinionFeatureExtraction*, is shown below.



(a) Patterns *feature+opinion* versus *verb+feature* (b) Patterns adverb+adjective versus others

Fig. 1. Patterns of subjective sentences.

Algorithm 1. *OpinionFeatureExtraction*

```
      Input: a collection of reviews, $D=\{d_1, d_2, \ldots, d_M\}$, and a set
      of commonly used seed adverbs, adverbList, emotion verb
      list V, minimum support threshold minsup, minimum
      similarity threshold minsim, and length of n-gram n
      Output: a set of opinions, P, and a set of features, F
1.    P = ExtractOpinion(D, adverbList)
2.    F=$\emptyset$, tempAdj = P;
3.    Do
4.      tempF=$\emptyset$;
5.      For each sentence s of each review d in D
6.        For each adj in tempAdj
7.          nounPhrases=ExtractNoun(s,adj);
8.          For each n-gram phrase nph in nounPhrases
9.            insert nph into tempF and accumulate support if
      it is repeated;
10.           if f$\in$FOP such that f.feature is null, f.s_id=s.s_id
      and f.adj=adj
11.             f.feature=nph
12.           else insert(d.user_id,d.item_id,s.s_id,nph,adj) into
      FOP;
13.     If tempF is empty return;
14.     tempAdj=$\emptyset$;
15.     For each sentence s of review d in D
16.       For each adj extracted from s based on tempF
17.         if(adj is not in P)
18.           insert adj into tempAdj;
19.           insert (d.user_id, d.item_id, s.s_id, $\emptyset$,adj) into
      FOP;
20.     P = P tempAdj, F = F tempF;
21.   while (tempAdj $\neq \emptyset$)
22.   for each sentence s of review d that contains no feature
      in F and no opinion in P
23.     if s contains any emotion verb $v \in V$
24.       insert (d.user_id,d.item_id,s.s_id, $\emptyset$,v) into FOP
25.       nounPhrase=ExtractNounE(s,v); //extract n-gram
      noun phrases around verb v
26.       Insert nounPhrase into F and accumulate support for
      repeated ones;
27.       For each n-gram nph in nounPhrase
28.          insert (user_id,item_id,s_id,nph,v) into FOP;
29.   MergeSynonymFeature(FOP,F, minsim); //merge the
      synonym features in F
30.   Remove from FOP and F features that have support less
      than threshold minsup;
```

In the algorithm *OpinionFeatureExtraction*, we first extract an initial set of opinions through ExtractOpinion (Step 1), which is described in Section 3.1. Then, according to this set, each *n*-gram noun phrase is extracted as a candidate feature, and its frequency is counted during extraction (Steps 5 to 12). The candidate feature and opinion mapping is done as the noun phrase is added into feature set *tempF* (Steps 6 to 12). Based on these features, opinions are extracted (Steps 10 to 13), and the feature-opinion pair set *FOP* is also expanded (Steps 18 to 21). Newly extracted opinions are added to the initial opinion set, candidate features found are put into *F*, and the support of features that already exist are accumulated. The two procedures are repeated until no new opinion is found. Then, for those sentences containing an emotion verb, which are collected manually from the list of positive and negative Chinese word lists (Dong and Dong 2010), *n*-gram around the verb is extracted and added to *F*, and the feature-emotion verb pairs are added into the *FOP* set (Steps 22 to 28). If the pair already exists in it, its support is increased (Step 26). Afterwards, procedure *MergeSynonymFeature* is applied to merge synonym candidate features, and the *FOP* is updated during the merging (Step 29, the details of which are described in the following sub-section). Finally, candidate features in *F* and the corresponding quintuple in *FOP* are pruned by the support threshold *minsup* (Step 19). Those that remained are output as features.

3.3. Synonym feature identification

Many synonyms are found in feature expressions. The identification and combination of these synonyms can afford them larger support and can also help the user understand them better. The identification method is based on the similarity of features. Two factors are considered when computing the similarity of two features:

(1) The length of the longest common substring of the two features. In the Chinese language, a word is composed of characters, and commonly, the semantics of characters indicate the meaning of the word. Hence, the longer the common substring is, the more similar the two features are.
(2) The number of times the two features co-occur in review sentences. The less the co-occurrence of the two features in review sentences, the more likely they are similar. On one hand, different people likely use different words to describe the same feature of a product because of different linguistic expression habits, which cause the synonym phenomena. On the other hand, for one person, one is unlikely to use different words to describe the same feature. Thus, if two features always co-occur in the review of one person, the likelihood of being synonyms is low.

Therefore, the similarity of features f_i and f_j can be measured by Eq. (1):

$$\text{Similarity}(f_i, f_j) = \frac{\text{length}(\text{substring}(f_i, f_j))}{\max(\text{co-occurrence}(f_i, f_j), 0.5)} \tag{1}$$

In Eq. (1), *substring*(f_i,f_j) denotes the common characters shared, and *length*( ) represents the number of common characters. Additionally, *co-occurrence*(f_i,f_j) represents the number of times the two features appear in the same review. When they do not appear in any review, *max*(*co-occurrence*(f_i,f_j), 0.5) = 0.5.

For any two features f_i and f_j, if their similarity measured by *similarity*(f_i,f_j) is greater than the user-specified similarity threshold *minsim*, they are considered synonym features, and one of them with less support is replaced by the other one. The major steps are given below as Algorithm 3.

Algorithm 3. Merge Synonym Feature

```
MergeSynonymFeature (FOP: a set of quintuples, F: a set of
  features, minsim: minimum similarity threshold)
for each feature $f_i \in F$
  For each feature $f_j \neq f_i \in F$
    If similarity $(f_i, f_j) \geqslant$ minsim and suppose support $(f_i)$
  > support$(f_j)$
    Replace the feature $f_j$ in F with $f_i$;
    Replace the feature $f_j$ of element in FOP with $f_i$;
```

4. Combining user preference and opinion for recommendation

The traditional recommendation methods such as collaborative filtering are based mainly on the idea that similar customers may have similar interests. In other words, similar customers may like similar products or services. These methods focus on the commonality among customers or among items. However, the difference between customers is another angle toward solving the problem. Besides, collaborative filtering uses only the rank or overall score information. In fact, online customer reviews provide rich information. We study how to make use of free text reviews as well as the scores users assign for the features to make a recommendation. For free text reviews, we first use the opinion mining technique to extract features and opinions, as well as the polarity of each opinion. Then, we combine the score, and feature and opinion polarities to infer the preference of customers and make recommendations based on the preferences.

In this section, we first present the way to model restaurants and users. We then introduce the proposed recommendation algorithm, as well as a novel method to capture the user's preferred features of a restaurant.

4.1. The restaurant model

We use a simple model to describe the relationship between each restaurant and each feature. To describe restaurant r_k, we use vector $\vec{r}_k = (s_{1k}, s_{2k}, \ldots, s_{Lk})$ obtained using Eq. (2), where L is the total number of features, and $1 \leqslant k \leqslant K$.

$$S_{jk} = \frac{1}{n_{jk}} \sum_i s_{ijk} \tag{2}$$

where n_{jk} is the number of users who rated restaurant r_k on feature $f_j \in F$.

4.2. The user preference model

To recommend the correct items to a particular user, we must identify that user's preferences for items. Each item has many features. Different users may care about different features of the same item. How can we know the real concerns of a user? To address this issue, we develop two novel measures, *concern* and *requirement*, based on the following two assumptions, respectively:

(1) If a user comments on a feature more frequently than on average, he (or she) cares about this feature more.
(2) If a user rates a feature frequently lower than other users for different restaurants, his requirement for this feature is higher.

Let s_{ijk} represent the score that the reviewer u_i rates or comments on the feature f_j for restaurant r_k. Based on Assumption 1, to measure the degree to which user u_i cares about feature f_j, we define the measurement *concern* using Eq. (3):

$$concern(u_i, f_j) = \frac{count\_r(u_i, f_i) + 1}{count\_r(u_i)} \times \frac{N_r}{count\_r(f_i) + 1} \tag{3}$$

where $count\_r(u_i, f_j)$ represents the number of restaurants for which the user u_i rated feature f_j, $count\_r(u_i)$ is the number of restaurants that user u_i reviewed, N_r is the number of restaurants with customer reviews, and $count\_r(f_j)$ is the number of restaurants with the feature f_j being commented on. If user u_i comments on feature f_j many times but feature f_j is not often reviewed overall, the value of *concern*(u_i, f_j) becomes high. Hence, user u_i pays more attention to this feature than others do. In such a case, we assume that the user prefers this feature.

The second assumption indicates that if user u_i usually ranks feature f_j lower than common users do, we argue that user u_i has a higher requirement on this feature compared with the others. However, before we use this assumption, we need to consider first the ranking habit of the user. Different users usually have different standards when they comment on or rate something. Stringent users may rate all features lower than other users, whereas lenient users usually rate higher.

To eliminate the effect of the ranking habit of the user, we transform the score s_{ijk} in Eq. (4):

$$S'_{ijk} = s_{ijk} - avg(u_i, r_k) + MAX, \text{where } avg(u_i, u_k) = \frac{1}{m} \sum_{j=1}^{m} s_{ijk}, \tag{4}$$

where m is the number of features the user u_i rated or commented on for restaurant r_k. We then replace s_{ijk} with S'_{ijk}.

Based on Assumption 2, we develop a measure called *requirement* for user u_i on feature f_j in Eq. (5), where R_{ij} denotes the set of restaurants wherein the user u_i rates its feature f_j, and R_j denotes the set of restaurants whose feature f_j has been rated.

$$requirement(u_i, f_j) = \begin{cases} \dfrac{\sum_{r_k \in R_{ij}} \delta(u_i, f_j, r_k)}{count\_r(u_i, f_j)} & if \quad count\_r(u_i, f_j) \neq 0 \\ \dfrac{1}{count\_r(f_j)} \sum_{r_k \in R_j} \dfrac{0.1}{avg(f_j, r_k)} & otherwise \quad (case\ 3) \end{cases} \tag{5}$$

$$\text{Where } \delta(u_i, f_j, r_k) = \begin{cases} \dfrac{avg(f_j, r_k) - s_{ijk} + 1}{avg(f_j, r_k)} & \text{if} \, avg(f_j, r_k) > S_{ijk} > 0 \quad \text{(case 1)} \\ \dfrac{1}{avg(f_j, r_k)} & otherwise \quad \text{(case 2)} \end{cases}$$

Three cases are considered by Eq. (5).

Case 1. User u_i has rated feature f_j for at least one restaurant (i.e., $count\_r(u_i, f_j) \neq 0$). For restaurant r_k, the rating s_{ijk} is lower than the average rating rated by all users for feature f_j and for r_k ($avg(f_j, r_k) > s_{ijk}$). In this case, the higher the $avg(f_j, r_k)$-S_{ijk}, the higher the requirement. To make sure $avg(f_j, r_k) - S_{ijk}$ is always higher than that in Case 2, numerator $avg(f_j, r_k) - S_{ijk} + 1$ is used.

Case 2. User u_i has rated feature f_j for at least one restaurant (i.e., $count\_r(u_i, f_j) \neq 0$). For restaurant r_k, the rating S_{ijk} is not lower than the average rating rated by all users for feature f_j and for r_k ($avg(f_j, r_k) \leqslant s_{ijk}$). In this case, we use the maximum difference 0 to measure the requirement.

Case 3. User u_i has never rated feature f_j for any restaurant (i.e., $count\_r(u_i, f_j) = 0$). In this case, to make the value of the requirement lesser than and comparable to that in Case 2, 0.1 is used instead of 1 in the numerator.

By combining the two measurements in Eq. (6), we can measure the importance of feature f_j to user u_i.

$$w_{u_i, f_j} = concern(u_i, f_j) \times requirement(u_i, f_j) \tag{6}$$

Thus, the preference of each user u_i can be described by a vector $\vec{u}_i = (w_{i1}, w_{i2}, \ldots, w_{iL})$ in the feature space, where L is the total number of features.

4.3. Algorithm

Based on the two models, we develop the *preference and opinion-based recommendation* (*PORE*) *algorithm*. Our idea involves ranking the restaurants for each user u_i. Based on such a ranking, we recommend the top k restaurants with the highest satisfaction scores to the user. From the preferences of the user and the description of the restaurant, we use Eq. (7) to predict how user u_i will rank restaurant r_k in terms of satisfaction.

$$\mathit{Satisfaction}(u_i, r_k) = \frac{W_{u_i f_1} \times S_{1k} + W_{u_i f_2} \times S_{2k} \ldots + W_{u_i f_L} \times S_{Lk}}{\sum_{j=1}^{L} W_{u_i f_j}} \quad (7)$$

The major steps of Algorithm 2, *PORE*, are given below.

Algorithm 2. *PORE*

```
Input: a set $S$ of (user $u_i$, restaurant $r_k$, feature $f_j$, feature value
  $s_{ijk}$), $R$, $U$, $F$
Output: the satisfaction user-restaurant satisfaction matrix
Begin
1.    For each restaurant $r_k \in R$ and each feature $f_j \in F$
2.      Compute $s_{jk}$ by Eq. (2);
3.    For each tuple in set S
4.      Transform $s_{ijk}$ by Eq. (4) and normalize it to range
      [0,1];
5.    For each user $u_i \in U$
6.      For each feature $f_j$
7.        Compute concern ($u_i$,$f_j$) by Eq. (3);
8.        Compute requirement ($u_i$,$f_j$) by Eq. (5);
9.        Compute $w_{ui,fj}$ by Eq. (6);
10.   For each user $u_i \in U$
11.     For each restaurant $r_k \in R$
12.       Compute satisfaction($u_i$,$r_k$) by Eq. (7);
13.   Return satisfaction matrix;
      End
```

Given the set of feature values s_{ijk} corresponding to user u_i, restaurant r_k, and feature f_j output by the feature-opinion extraction module, we first calculate the average value for each feature to describe each restaurant (Steps 1 and 2). Absolute feature values are then transformed to relative ones to eliminate the effect of the ranking habit of the user (Steps 3 and 4). To describe the preference of the user, we first calculate the measurement *concern* using Eq. (3) (Step 7) as well as the measurement *requirement* using Eq. (5) (Step 8). We then combine the results in Eq. (6) (Step 9). Finally, according to the restaurant's model and the preference of the user, we calculate the satisfaction score to measure how much each restaurant can satisfy each user using Eq. (7) (Steps 10 to 12). From the satisfaction scores rated by each user u_i, we can rank the restaurants not rated and commented by u_i and recommend the top k restaurants to u_i.

5. Empirical study

To evaluate the performance of the proposed approaches, we crawled a real online restaurant review dataset through a well-known website (beijing.koubei.com). A total of 54,208 reviews were collected. The restaurants with only one review were excluded. The total number of reviews is 53,734, which were written by 9992 users for 3094 restaurants.

Running environment. All experiments were conducted on a computer with a 2.66 GHz Intel Core 2 processor, 4 GB RAM, and 32-bit Windows Server 2008. All algorithms were implemented in C++.

5.1. Opinion-feature extraction

To evaluate the effectiveness of the proposed opinion-feature extraction method, we randomly selected 4000 reviews and manually extracted features and opinions from these reviews. We obtained 157 distinct features and 176 distinct opinions. We compared our approach with the one proposed by Hu and Liu (2004). Precision (the fraction of extracted features or opinions that are correct), recall (the fraction of correct features that are extracted), and $F_1 = (2 \times \mathit{precision} \times \mathit{recall})/(\mathit{precision} + \mathit{recall})$ of opinion and feature extractions by the two approaches are given in Tables 1 and 2. We tested several values for the threshold used in the algorithm by Hu and Liu (2004), and finally set it at 1%, which led to the best result. When running our algorithm *OpinionFeatureExtraction*, we set *minsup* to be 0, the length of *n*-gram to be 2, and *minsim* to be 2.

Table 1
Opinion extraction performance.

Algorithms	Precision (%)	Recall (%)	F_1
OpinionFeatureExtraction	56.68	60.23	0.59
Hu and Liu's method	56.13	41.48	0.48

Table 2
Feature extraction performance.

Algorithms	Precision (%)	Recall (%)	F_1
OpinionFeatureExtraction	65.28	29.94	0.41
Hu and Liu's method	23.67	56.69	0.33

From Table 1, we can see that in terms of opinion extraction, the precision of our method is similar to that of Hu and Liu. However, our method's recall is much higher than that of Hu and Liu. Therefore, our method has higher F_1 than that of Hu and Liu. The reason why we could find more opinion words is that our adverb-based opinion words extraction method can find those opinions that do not occur around features.

For feature extraction, *OpinionFeatureExtraction* has higher precision and lower recall than the method of Hu and Liu, as shown in Table 2. However, overall, our method has a higher F_1 value than that of Hu and Liu. Our method has higher precision because we do not extract nouns that occur frequently and do not occur with opinion words, and most of them are not features.

To verify if the synonymy feature identification method is effective, we manually found 77 pairs of synonyms among all the features. Using the synonym feature identification method, 103 pairs were regarded as synonyms. Therefore, the precision is 56.3%, recall is 75.3%, and F_1 is 64.4%. This method can still be improved. We attempted to take the number of times that two features appear with the same opinions. This, however, did not improve precision. We will leave this area of performance improvement for future work.

Finally, we compared the efficiency of our method with that of Hu and Liu. The runtime is illustrated in Fig. 2.

The total time used in the method of Hu and Liu was about 387 s, whereas that of our method was about 27 s. The reason for this difference is that the method of Hu and Liu finds features based on association rule mining, which is time consuming. Our method finds opinions based on the adverb list and finds features based on opinions.

5.2. Recommendation algorithm

To evaluate the performance of the proposed recommendation method *PORE*, we predict the value that each user will score for each restaurant. We then compare *PORE* with two collaborative filtering methods: user-based and item-based methods. The true value that each user scores for a restaurant is the average of scores on four features: environment, service, taste, and cost performance. The value of each rating feature is in a five-point scale: 1, 2, 3, 4,

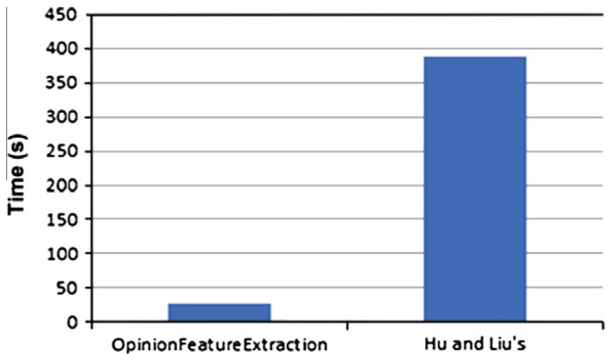


Fig. 2. Runtime comparison of our method with the method of Hu and Liu.

or 5. We use matrix A to save the true values. We also compare *PORE* with the classical similarity-based multi-criteria algorithm proposed by Tang and McCalla (2009).

The input of the collaborating filtering methods is matrix A where the value A_{ij} that corresponds to the ith row and jth column of matrix A is the average score the ith user scored for the jth restaurant. $A_{ij} = 0$ means that the user has not scored the restaurant yet. For user-based and item-based collaborative filtering methods, we used cosine similarity to measure the similarity between users or items. We selected the most similar users or items to make the prediction.

The multi-criteria algorithm we implemented is item-based multi-criteria collaborative filtering method (Tang and McCalla 2009). The input of this algorithm includes five matrices: B_i ($i = 1, 2, 3, 4, 5$). For any of the first four matrices B_i, the entry B_{ijk} that corresponds to the jth row and the kth column of matrix B_i, is the score that the kth user rated on the ith feature of the jth item. The value of B_{5jk} corresponding to the jth row and the kth column of matrix B_5 is the average score the kth user assigned on the jth item.

For the *PORE* algorithm, apart from the four features to which the user assigns a score, we use ten more features to make the prediction: compartment (包□), price (价格), traffic (交通), character (特色), quantity (分量), variety (品□), atmosphere (□氛), business (生意), dishes (菜色), and speed (上菜速度). These ten features are frequently commented on by users. We assign scores to these features according to the polarity and the intensity of the opinion words paired with them. Many methods for sentiment analysis are available to obtain the polarity of each opinion (Gamon et al. 2005, Hu and Liu 2004, Liu et al. 2005, 2008). In our experiments, we use the method proposed by Liu et al. (2008). For positive opinions, we assign values 1.2, 1, and 0.8 according to the intensity of the attitude (high, medium, and low). For negative opinions, we assign the values -1.2, -1, and -0.8. We then normalize these values into range [1,5] through min–max normalization to make them consistent with the four rating scores. After normalization, these scores become 5, 4.67, 4.33, 1.67, 1.33, and 1. We call this numeric mapping of polarity *M1*. To test its effect on the performance of the recommendation method, we develop another mapping *M2*. The common five-point rating scale is a bit coarse. To make it finer, we add 1.5 between 1 and 2, and 4.5 between 4 and 5, that is, 5, 4.5, 4, 2, 1.5, and 1. This mapping becomes more natural.

To evaluate the effectiveness of our method, we conduct a five-fold cross-validation and compute the *mean absolute error* (MAE) for each fold. Originally, we have a total of 9992 users, 3094 restaurants, and 48,607 reviews. To do the five-fold cross-validation, we remove the users who commented on less than five restaurants. Consequently, 35,027 reviews of 3094 restaurants by 1707 users remain. We split the review of each user into five partitions, where four partitions identify preference and one part tests the performance. The prediction error of *PORE* for the two different mappings is given in Table 3. In this table, MAE i ($i = 1, 2, 3, 4$, and 5) is the MAE when the ith partition of the dataset is used as the test dataset.

Table 3
MAE comparison between the two different mappings.

	MAE 1	MAE 2	MAE 3	MAE 4	MAE 5	Average MAE
M1	1.22	1.18	1.18	1.18	1.20	1.19
M2	1.26	1.25	1.25	1.25	1.26	1.25

From Table 3, we can see that mapping *M1* has a slightly better accuracy than *M2*. However, the difference is not big. The results also tell us that the difference between various intensities of polarity (high, medium, and low) is small. In the following experiments, we use mapping *M1*.

To distinguish the importance of synonym identification, we compare the prediction results between the *PORE* algorithm with and without synonym identification in Table 4. Based on the comparison, synonym identification can clearly improve prediction accuracy.

To test if the different features have different effects on prediction accuracy, we compare *PORE* (using four rating features and ten opinion features identified by the proposed method) with two different settings: *PORE* using only the four rating features (*PORE* + 4 rating features) and *PORE* using four rating features plus ten features identified by Hu and Liu's method (*PORE* + Hu's features). The result of the comparison is shown in Table 5.

From Table 5, we can see that using opinion features extracted with higher precision by the proposed method can improve the recommendation precision.

The comparison among *PORE*, two collaborative filtering (CF) methods – item-based CF and user-based CF – and the multi-criteria recommendation method (Tang and McCalla 2009) are detailed in Table 6. It shows that *PORE* has the lowest prediction error, whereas the item-based method has the highest prediction error.

The algorithm proposed by Tang and McCalla (2009) can predict the score for each restaurant by combining all four features that the user rated and the feature whose score is the average of the four features. This method computes similarity between each pair of items based on cosine similarity for each feature and then based on the weight assigned for each feature. Finally, the similarity between two items is computed by the weighted sum of the five sim-

Table 4
MAE comparison between *PORE* with and without synonym identification.

Algorithms	MAE 1	MAE 2	MAE 3	MAE 4	MAE 5	Average MAE
PORE without synonym	1.46	1.46	1.47	1.45	1.45	1.45
PORE with synonym	1.22	1.18	1.18	1.18	1.20	1.19

Table 5
MAE comparison between PORE with different features.

Algorithms	MAE 1	MAE 2	MAE 3	MAE 4	MAE 5	Average MAE
PORE + 4 rating features	1.46	1.47	1.47	1.46	1.46	1.46
PORE + Hu's features	1.47	1.47	1.47	1.46	1.47	1.47
PORE	1.22	1.18	1.18	1.18	1.20	1.19

Table 6
MAE comparison between PORE and collaborative filtering methods.

Algorithms	MAE 1	MAE 2	MAE 3	MAE 4	MAE 5	Average MAE
PORE	1.22	1.18	1.18	1.18	1.20	1.19
Item-based CF	3.11	3.10	3.09	3.09	3.10	3.10
User-based CF	2.99	2.96	2.95	2.95	2.97	2.96
Tang and McCalla	2.84	2.78	2.80	2.78	2.82	2.80

Table 7
Pairwise *t*-test between different recommendation methods.

Pairs of methods	*p*-Value
PORE vs. (*PORE* + 4 rating features)	2.58691E−07
PORE vs. (*PORE* + Hu's features)	2.37642E−06
PORE vs. item-based CF	3.33944E−13
PORE vs. user-based CF	1.19323E−15
PORE vs. Tang and McCalla	1.05462E−12

ilarity scores of the two items. We conducted many experiments on different weights and derive the best result when the weight vector is (0.15, 0.15, 0.15, 0.15, and 0.4).

To verify the significance of the MAE difference between *PORE* and each of the other methods, as shown in Tables 5 and 6, we conducted a pairwise *t*-test. The null hypothesis is H_0:$D = 0$, where D is the difference of MAE between *PORE* and any of the other methods such as item-based collaborative filtering (item-based CF). The alternative hypothesis H_1 is $D \neq 0$. The significant level α is set at 0.05. The comparison between different pairs of methods is shown in Table 7. We can see from the table that all the *p*-values are much less than 0.05, which means that the null hypothesis is rejected, that is, the differences between *PORE* and each of the other four methods are significant.

Originally, each user writes reviews and gives scores for about 5 out of 3094 restaurants. Therefore, the data are very sparse. The ability to deal with sparseness is important. To measure this ability, we compare the average number of restaurants for which each method can predict the score by each user. Using the proposed method, we can obtain the prediction of each restaurant for each user. The user-based method can only acquire approximately 21 restaurants for each user, whereas the item-based method can acquire approximately 24 restaurants for each user. The reason for such difference is that for the user-based collaborative filtering method to be able to estimate the similarity between users, two users must score at least two of the same restaurants. Similarly, for the item-based method, two items must have at least two users giving them scores. As the data are very sparse, the two collaborative filtering methods cannot make the prediction for the majority of the restaurants. As our method does not have this requirement, it can deal with data sparseness better.

Finally, we evaluate the efficiency of these algorithms. The runtime of each method on the original dataset with 9992 users and 48,607 reviews is illustrated in Fig. 3. As shown in the figure, *PORE* has the highest efficiency, whereas the user-based method has the lowest efficiency. The method proposed by Tang and McCalla has higher efficiency than the user-based method, but has lower efficiency than the item-based method. The item-based method is better than the user-based method because in our dataset, the number of users is 9992, which is far greater than the number of restaurants, which is 3094. For these three methods, we need to compute similarity of each pair of users or items, which is time consuming.

6. Related work

Recommendation methods are usually categorized into two major classes: content-based and collaborative filtering methods.

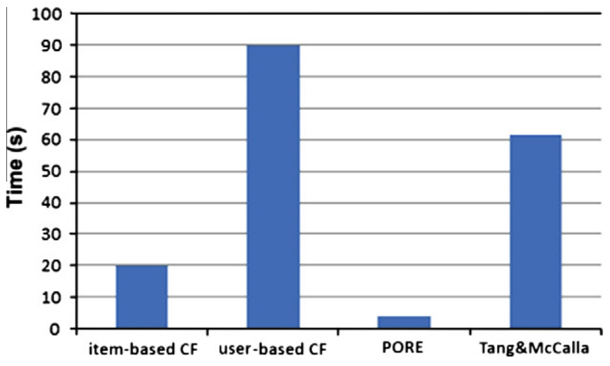


Fig. 3. Efficiency comparisons of the different recommendation methods.

Content-based methods (Lang 1995, Mooney and Roy 2000) recommend items that are similar to those the user liked before. *Collaborative filtering* (Billsus and Pazzani 1998, Breese et al. 1998, Goldberg et al. 2001, Marlin 2004, Nakamura and Abe 1998, Resnick et al. 1994) recommends items liked by similar users who share the same preferences. Our recommendation method is different from these two kinds of methods. We attempt to infer the preferences of users on the features of the items. In this method, we do not need to find similar users or similar items, which are difficult to find due to sparse data. Existing methods use a very small part of global information to make a prediction, whereas our method maximizes all information. Content-based method makes recommendations based on the features of the items. However, the values of some features are sometimes hard to determine. For example, the good services of a restaurant are difficult to determine. Therefore, in our method, we make use of the user online reviews. To capture the main points of each review, we study effective and efficient opinion-mining techniques.

Multi-component rating collaborative filtering (Adomavicius and Kwon 2007, Adomavicius et al. 2010, Sahoo et al. 2012, Tang and McCalla 2009) studies the incorporation of multi-component rating into traditional single-rating recommendation methods such as collaborative filtering. These studies show that multi-component ratings can improve recommendation accuracy to some extent. Our work is somehow related to these studies except that we use not only rating information but also opinion information. We infer the preferences of the user based on the differences among the ratings and opinions of different users.

Our study is also related to conjoint analysis. *Conjoint analysis* is a statistical technique that is frequently used in marketing research to measure the importance of different features of products or services (Green and Srinivasan 1978, 1990). The importance is inferred based on the overall rating of customers on virtual products that are designed by combining levels of different features. However, our work is different based on two aspects. First, our aim is to infer the preference of individuals on each feature, whereas conjoint analysis aims to estimate the common preference among all customers. Second, the method is different. Our analysis is based on the rating given to each feature of a real product or service, whereas conjoint analysis is based on the overall rating for a virtual product.

Existing methods for feature-opinion extraction can be classified into two categories: supervised and unsupervised. The *supervised method* is similar to the method used by Lou and Yao (2006), which requires a manual labeling training set. *Unsupervised methods* exploit the knowledge base or statistics to automatically extract features and opinions. Examples of knowledge base include wordNet (Christiane 1998, Yi et al. 2003) and domain ontology (Lou and Yao 2006). Popescu and Etzioni (2005) employed the statistical method. They first extracted features using the web-based information extraction system, KnowItAll. They subsequently obtained the architecture of the features by computing pairwise mutual information between features and a set of meronymy discriminators associated with the product category. For the Chinese language, meronymy discriminators are unavailable.

Our work is closely related to the research by Hu and Liu (Hu and Liu 2004, Liu et al. 2005). In their work, an association rule-mining method is used to calculate the concurrence of the noun and the noun phrase. Frequent feature words are first found, followed by adjacent adjectives that are considered as opinions and infrequent feature words that are based on opinion words. Our experiments conducted on Chinese reviews show that this method may find a large number of frequent nouns, many of which are not features. For example, nouns such as birthday and friends are frequently mentioned in reviews, but they are not features. Therefore, this method has low precision when extracting features. Our proposed method uses adverb and adjective combinations to initiate the opinion-feature mutual extraction process. The adverb and adjective combination rule is studied by a number of researchers for sentiment analysis (Benamara et al. 2007, Subrahmanian and Reforgiato 2008, Zhang et al. 2009). So far, this rule has not been used in feature-opinion extraction in existing studies.

7. Conclusion

We have proposed a very different method to infer a user's preferences and to recommend products to users, which we call *PORE*. This method recommends items based on the preferences of users on the features of each item. To infer a user's preferences on the features, we developed a new method to extract features and opinions from online user reviews and proposed measures to infer a user's preferences. We evaluated these methods by conducting experiments on online restaurant reviews written in Chinese. The results that we obtained show that the adverb-based opinion feature extraction method can extract the most opinions and features and has a better performance than existing methods. The experiment results also indicate that the recommendation algorithm is more capable of dealing with data sparseness, and has better prediction accuracy and efficiency.

Some future work is appropriate as a follow-up on this research. First, context-aware recommendation methods (Panniello et al. 2009) may further improve prediction accuracy. For example, time and location may be important for a person to choose a restaurant. In the future, we will study how to combine contextual information with our method. Second, we will study how to deal with the cold-start problem under the proposed recommendation framework. Making a recommendation in a mobile situation is another direction for future work (Ge et al. 2010). Finally, we will further study how our feature-opinion extraction method can perform on different language corpus and on corpus apart from product reviews such as micro-blogs (Alexander and Patrick 2010).

Acknowledgements

This work was supported by the National Natural Science Foundation of China under Grant Nos. 70871068, 70890083 and 71110107027 and the 973 program of China under Grant No. 2012CB316205.

References

Adomavicius, G., and Tuzhilin, A. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17, 6, 2005, 734–749.

Adomavicius, G., and Kwon, Y. New recommendation techniques for multi-criteria rating systems. *IEEE Intelligent Systems*, 22, 3, 2007, 48–55.

Adomavicius, G., Manouselis, N., and Kwon, Y. Multi-criteria recommender systems. In F. Ricci, L. Rokach, and P. B. Kantor (eds.), *Recommender Systems Handbook: A Complete Guide for Research Scientists and Practitioners*, Springer, New York, NY, 2010.

Aggarwal, C., and Zhai, C. *Mining Text Data*. Springer, New York, NY, 2012.

Alexander, P., and Patrick, P. Twitter as a corpus for sentiment analysis and opinion mining. In *Proceedings of the Seventh Conference on International Language Resources and Evaluation*, Valetta, Malta, 2010.

Ali, K., and Van Stam, W. TIVo: Making show recommendations using a distributed collaborative filtering architecture. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Seattle, WA, 2004, 394–401.

Benamara, F., Cesarano, C., Picariello, A., Reforgiato, D., and Subrahmanian, V. Sentiment analysis: Adjectives and adverbs are better than adjectives alone. In *Proceeding of International Conference on Weblogs and Social Media*, March 2007, San Jose, CA, 2007, 1–7.

Bennett, J., and Lanningm, S. The Netflix prize. In *Proceedings of 2007 KDD Cup and Workshop*, San Jose, CA, 2007.

Billsus, D., and Pazzani, M. J. Learning collaborative information filters. In *Proceedings of the Fifteenth International Conference on Machine Learning*, Morgan Kaufmann, San Francisco, CA, 1998, 46–54.

Breese, J. S., Heckerman, D., and Kadie, C. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, Morgan Kaufmann, San Francisco, CA, 1998, 43–52.

Christiane, F. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA, 1998.

Ding, X., and Liu, B. The utility of linguistic rules in opinion mining. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Amsterdam, Netherlands, July 23–27*, ACM Press, New York, NY, 2007, 811–812.

Dong, Z., and Dong, Q. HowNet Knowledge Database. 2010. Available at www.keenage.com/html/c_index.html.

Gamon, M., Aue, A., Corston-Oliver, S., and Ringger, E. Pulse: Mining customer opinions from free text. In *Proceedings of the 6th International Symposium on Intelligent Data Analysis, Madrid, Spain, September 8–10, 2004, Lecture Notes in Computer Science, 3646*, Springer Verlag, New York, NY, 2005, 121–132.

Ge, Y., Xiong, H., Tuzhilin, A., Xiao, K., Gruteser, M., and Pazzani, M. J. An energy-efficient mobile recommender system. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, July 25–28, 2010*, ACM Press, New York, NY, 2010, 899–907.

Ghani, R., Probst, K., Liu, Y., Krema, M., and Fano, A. Text mining for product attribute extraction. *SIGKDD Explorations*, 8, 1, 2006, 41–48.

Goldberg, K., Roeder, T., Gupta, D., and Perkins, C. Eigentaste: a constant time collaborative filtering algorithm. *Information Retrieval*, 4, 2, 2001, 133–151.

Green, P., and Srinivasan, V. Conjoint analysis in consumer research: issues and outlook. *Journal of Consumer Research*, 5, 1978, 103–123.

Green, P. E., and Srinivasan, V. Conjoint analysis in marketing: new developments with implications for research and practice. *Journal of Marketing*, 54, 1990, 3–19.

Hu, M., and Liu, B. Mining and summarizing customer reviews. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, WA, August 22–25, 2004*, ACM Press, New York, NY, 2004, 8–17.

Kobayashi, N., Iida, R., Inui, K., and Matsumoto, Y. Opinion mining on the Web by extracting subject-attribute-value relations. In *Proceedings of AAAI Spring Symposia on Computational Approaches to Analyzing Weblogs*, 2006, 86–91.

Lang, K. NewsWeeder: Learning to filter netnews. In *Proceedings of the Twelfth International Conference on Machine Learning*, Morgan Kaufmann Publishers, San Francisco, CA, 1995, 331–339.

Linden, G., Smith, B., and York, J. Amazon.com recommendations: item-to-item collaborative filtering. *IEEE Internet Computing*, 7, 1, 2003, 76–80.

Liu, B., Hsu, W., and Ma, Y. Integrating classification and association rule mining. In *Proceeding of the Third International Conference on Knowledge Discovery and Data Mining, New York, NY, August 27–31, 1998*, ACM Press, New York, NY, 1998, 80–86.

Liu, B., Hu, M., and Cheng, J. Opinion observer: Analyzing and comparing opinions on the Web. In *Proceedings of the 14th International Conference on World Wide Web, Chiba, Japan, 2005*, ACM Press, New York, NY, 2005, 342–351.

Liu, H., Yang, H., Li, W., Wei, W., He, J., and Du, X. CRO: A system for online review structurization. In *Proceedings of the 14th ACM International Conference on Knowledge Discovery and Data Mining, August 24–27, 2008, Las Vegas, NV*, ACM Press, New York, NY, 2008, 1085–1088.

Lou, D. C., and Yao, T. F. Semantic polarity analysis and opinion mining on Chinese review sentences. *Journal of Computer Applications*, 26, 11, 2006, 2622–2625.

Marlin, B. Modeling user rating profiles for collaborative filtering. In S. Thrun and K. S. B. Scholkopf (eds.). *Advances in Neural Information Processing Systems*, Vol. 16, MIT Press, Cambridge, MA, 2004, 627–634.

Mooney, R. J., and Roy, L. Content-based book recommending using learning for text categorization. In *Proceedings of the ACM International Conference on Digital Libraries, San Antonio, TX*, ACM Press, New York, 2000, 195–204.

Morinaga, S., Yamanishi, K., Tateishi, K., and Fukushima, T. Mining product reputations on the web. In *Proceedings of the Eight ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Edmonton, Alberta, Canada, July 23–26*, ACM Press, New York, NY, 2002, 341–349.

Nakamura, A., and Abe, N. Collaborative filtering using weighted majority prediction algorithms. In *Proceedings of the Fifteenth International Conference on Machine Learning*, Morgan Kaufmann Publishers, San Francisco, CA, 1998, 395–403.

Panniello, U., Tuzhilin, A., Gorgoglione, M., Palmisano, C., and Pedone, A. Experimental comparison of pre- vs. post-filtering approaches in context-aware recommender systems. In *Proceedings of the 3rd ACM Conference on Recommender Systems, October 23–25, 2009*, ACM Press, New York, NY, 2009, 265–268.

Popescu, A., and Etzioni, O. P. Extracting product features and opinions from reviews. In *Proceeding of Joint Conference on Human Language Technology/Conference on Empirical Methods in Natural, Language*, 2005, 339–346.

Resnick, P., Iacovou, N., Sushak, M., Bergstrom, P., and Riedl, J. GroupLens: An open architecture for collaborative filtering of netnews. In J. B. Smith, F. D. Smith, and T. W. Malone (eds.), *Proceedings of the Conference on Computer Supported Cooperative Work. Chapel Hill, NC*, ACM Press, New York, NY, 1994, 175–186.

Sahoo, N., Krishnan, R., Duncan, G., and Callan, J. The halo effect in multicomponent ratings and its implications for recommender systems: the case of Yahoo! movies. *Information Systems Research*, 23, 1, 2012, 231–246.

Subrahmanian, V. S., and Reforgiato, D. AVA: adjective–verb–adverb combinations for sentiment analysis. *IEEE Intelligent Systems*, 23, 4, 2008, 43–50.

Tang, T. Y., and McCalla, G. The pedagogical value of papers: a collaborative-filtering based paper recommender. *Journal of Digital Information*, 10, 2009, 2.

Yi, J., Nasukawa, T., Bunescu, R., and Niblack, W. Sentiment analyzer: Extracting sentiments about a given topic using natural language processing techniques. In *Proceedings of the 3rd IEEE International Conference on Data Mining, Melbourne, FL*, IEEE Computing Society Press, Washington, DC, 2003, 427–434.

Zhang, H., Wang, M., Lv, Z., and Zhang, X. Chinese lexical analysis system. ICTCLAS, 2010. Available at ictclas.org/Download.html.

Zhang, C., Zeng, D., Li, J., Wang, F., and Zuo, W. Sentiment analysis of Chinese documents: from sentence to document level. *Journal of the American Society for Information Science and Technology*, 60, 12, 2009, 2474–2487.