Knowledge-Based Systems 50 (2013) 44–59

Contents lists available at SciVerse ScienceDirect

Knowledge-Based Systems

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

Preference-based clustering reviews for augmenting e-commerce recommendation

Li Chen *, Feng Wang

Department of Computer Science, Hong Kong Baptist University, Hong Kong, China

ARTICLE INFO

Article history:
Received 12 December 2012
Received in revised form 9 May 2013
Accepted 9 May 2013
Available online 30 May 2013

Keywords:
Recommender system
Product reviews
Opinion mining
Multi-attribute utility theory
Preference learning
Latent class regression model
Clustering
E-commerce

ABSTRACT

In the area of e-commerce, there exists a special, implicit community being composed of *product reviewers*. A reviewer normally provides two types of info: one is the overall rating on the product(s) that s/he experienced, and another is the textual review that contains her/his detailed opinions on the product(s). However, for the high-risk products (such as digital cameras, computers, and cars), a reviewer usually commented one or few products due to her/his infrequent usage experiences. It hence raises a question of how to identify the preference similarity among reviewers. In this paper, we propose a novel clustering method based on *Latent Class Regression model (LCRM)*, which is essentially able to consider both the overall ratings and feature-level opinion values (as extracted from textual reviews) to identify reviewers' preference homogeneity. Particularly, we extend the model to infer individual reviewers' *weighted feature preferences* within the same iterative process. As a result, both the cluster-level and reviewer-level preferences are derived. We further test the impact of these derived preferences on augmenting recommendation for the active buyer. That is, given the reviewers' feature preferences, we aim to establish the connection between the active buyer and the cluster of reviewers by revealing their preferences' interrelevance. In the experiment, we tested the proposed recommender algorithm with two real-world datasets. More notably, we compared it with multiple related approaches, including the non-review based method and non-LCRM based variations. The experiment demonstrates the superior performance of our approach in terms of increasing the system's recommendation accuracy.

© 2013 Elsevier B.V. All rights reserved.

1. Introduction

Due to the explosive growth of information appearing in current Web, recommender systems have been widely developed in recent years for eliminating the information overload and providing personalized item recommendation to the user. So far, most of recommender systems, such as user-based collaborative filtering techniques [19], content-based methods [35], and matrix factorization approaches [27], have been built under the assumption that a sufficient amount of user ratings on known-items can be easily obtained (based on which the system can infer the user's preferences and identify user-user similarity). However, in the e-commerce environment especially with the so called *high-risk products* (also called *high-cost* or *high-involvement* products, such as digital cameras, computers, and cars), because a user does not buy the high-risk product very often, it is normal that s/he is not able to rate many products. For the same reason, the current buyer is often a new user because s/he would not afford to buy the same kind of high-risk product before. These phenomena can be supported by the statistics reported in [23,53]: a great portion of users (e.g., >68% in Amazon dataset and >90% in resellerratings.com dataset) only gave feedback to one product. It is hence infeasible to purely adopt the classical recommending methods to benefit users in the high-risk product domains.

To solve the "new user" problem, related works have attempted to elicit the buyer's preferences on site by asking her/his to explicitly state the preferences over the product's features (e.g., the laptop's processor speed, memory capacity, screen size, etc.). The preference model is theoretically based on the Multi-Attribute Utility Theory (MAUT) [25], according to which all products can be ranked by their matching utilities with the user's stated preferences. However, though it is possible to obtain the buyer's needs via interactive preference elicitation techniques (such as the critiquing agent [7] that will be described in Section 2.1), the elicited preferences are still less complete and accurate, given the fact that the buyer cannot state her/his full preferences when s/he is confronted with the costly, unfamiliar products. This phenomenon was principally formulated in the field of decision theory as a type of adaptive, constructive decision behavior [34].

* Corresponding author. Address: 224 Waterloo Road, Kowloon Tong, Kowloon, Hong Kong, China. Tel.: +852 34117090; fax: +852 34117892.
E-mail addresses: lichen@comp.hkbu.edu.hk (L. Chen), fwang@comp.hkbu.edu.hk (F. Wang).

0950-7051/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.knosys.2013.05.006

Therefore, with the objective of developing more effective recommender system in e-commerce, in this paper, we have exerted to exploit the deep value buried in *product reviews* as contributed by other users to benefit the current new buyer. Particularly, we are interested in deriving the reviewers' feature preferences from the textual reviews they posted. Given that the review-item matrix is sparse, we have attempted to cluster reviewers into preference-based communities and simultaneously adjust individual reviewers' preferences. Such derived data can then be potentially helpful to predict the current buyer's missing preferences and enable the system to return more accurate recommendation. The main contributions of our work are indeed that: (1) we propose a *preference-driven approach* for learning reviewers and extracting clusters; (2) we build the relevance between the current buyer and reviewers based on *their feature-level preference similarity*; and (3) we develop the *review-based recommendation method* to address the "new user" issue.

Specifically, we aim to construct a reviewer's *weighted feature preferences* from her/his written review(s): $Pref_u = \{\langle f_i, w_{ui}\rangle | 1 \leqslant i \leqslant n\}$, where w_{ui} denotes the *weight* that reflects the importance degree that the user u places on feature f_i. An example is given below to illustrate the problem.

EXAMPLE. Reviewer *A* wrote a review to camera *C*, and her overall rating to this product is 5 (in the range of 1–5).

> *"It can produce a great image in low light environment. You can usually use it in AUTO mode and expect a good result. If you don't mid a little bit heavier and bigger camera compared with most of compact cameras, this is the one you should get it. Only con I can think of is its little bit short battery life. Better to consider to buy an additional battery."*

The question is then how the system could automatically derive the reviewer's weight preferences on the features that she mentioned in the above review (e.g., *which feature(s) is more important to her?*). It might be intuitive to count the feature's occurring frequency, so that if a feature appears more frequently, it is regarded more important than others [2,28]. However, this method cannot distinguish features which are with equal occurrences. Moreover, in the cases like the above example, the less frequent feature "image" is actually more important than the feature "battery life" because its opinion is consistent with the reviewer's overall rating on the product (both are positive) while the battery life's opinion is negative. It hence suggests that the user's overall rating along with her/his opinions on different features should be all considered so as to potentially more accurately reveal her/his weights on those features.

In this paper, we have first applied the Probabilistic Regression Model (PRM) to identify the relationship between the overall rating and features' opinion values for every reviewer (see Section 4). We have accordingly proposed PRM based k-NN & k-Means recommending methods and experimentally proved that the PRM-based methods perform more accurate than the non-review based method (that is without the incorporation of any reviews) and the non-preference based method (that does not stress deriving reviewers' weight preferences). However, the PRM-based methods might be still limited in the situation with sparse reviews (i.e., one or few reviews provided by each reviewer). We have thus endeavored to additionally improve the stability of derived reviewers' weight preferences by involving the clustering process. The basic idea is that, with all reviewers' info (i.e., the overall ratings and features' opinion values), we first conduct unsupervised clustering to group these reviewers, which is targeted to build the *cluster-level preferences* to represent a cluster of reviewers' common preferences. At the same time, we employ the cluster-level preferences to adjust individual reviewer's weight preferences (i.e., *reviewer-level preferences*). During the next iterative cycle, the reviewer-level preferences are further used to polish the clustering results. Such iteration will end when both types of preferences are stabilized and not changed. To accomplish these tasks, we have particularly extended *Latent Class Regression Model (LCRM)*. LCRM has been widely applied in the marketing area to perform the market segmentation (i.e., dividing prospective buyers into subsets who share preference homogeneity) [52]. Its main property is that it can take into account the whole structure of the targeted domain to divide a number of entities into latent classes, and enable each class to contain entities which inherently possess similar regression characteristics (in our case, the regression defines the relationship between the overall rating and features' opinion values). To suit our needs, we have modified the original form of LCRM so that both cluster-level regression model (i.e., with the cluster-level preferences as the outcome) and reviewer-level regression (i.e., with the reviewer-level preferences as the outcome) can be simultaneously generated and optimized. This proposed method is called LCRM*, which is new relative to our previous work [49,50]. Moreover, we have evaluated the algorithm's accuracy and compared it to other related ones on two real-world datasets.

In the following, we will first summarize related works, by classifying them into two categories: the recommender systems developed for high-risk product domains, and the review-based recommender systems (Section 2). After discussing their respective limitations, we start to describe our system's workflow and approaches (Section 3). The methods based on the Probabilistic Regression Model (PRM) will be in detail introduced in Section 4, and the methods based on the extended Latent Class Regression Model (LCRM) will be presented in Section 5. The experiment setup, evaluation metrics and results analysis will then follow (Section 6). At the end, we conclude the major findings (Section 7).

2. Related work

2.1. Recommender systems for high-risk products

As mentioned before, for high-risk products, because it is unusual to obtain a number of ratings on many products from a single user, researchers have mainly put focus on developing effective preference elicitation techniques for solving the "new user" problem.

Related works on preference elicitation. Preference elicitation is a process engaging users in some kind of "dialog" with the system [5]. The traditional methods typically involved users in a tedious and time-consuming procedure. For example, in [25], every attribute's utility function is assessed through the mid-value splitting technique. That is, given a range of attribute value $[x_a, x_b]$, the user is first asked to specify a mid-value point x_c for which the pairs (x_a, x_c) and (x_c, x_b) are differentially value-equivalent. Therefore, if $U(x_a) = 0$, $U(x_b) = 1$ (i.e., the point's utility), it infers that $U(x_c) = 0.5$. The utilities on other points can be similarly estimated (for example, finding mid-value points respectively for $[x_a, x_c]$ and $[x_c, x_b]$). Then, the pairs of products which have indifferent preferences are used to derive the trade-offs (i.e., relative weights) among attributes. Analytic hierarchy process (AHP) is an alternative elicitation method [41]: through a series of pairwise comparisons between products, it obtains the weights of decision criteria (i.e., the attributes) and the value function of each attribute. However, as most of users cannot clearly answer these questions upfront especially in complex decision environments [34,48], in recent years, some researchers

have emphasized the *incremental preference elicitation*. Specifically, they first acquired the user's initial preferences (that might be likely uncertain and incomplete) and then stimulated the user to participate in the conversation with the system so as to discover her/his hidden needs. The representative work is the so called *critiquing-based recommender system* [8] which has emerged in the form of both natural language models [44,47] and graphical user interfaces [4,40,38]. The main interaction component of these systems is that of *recommendation-and-critiquing*, by which users can provide feedback to the recommended item by choosing to improve some of its feature values and compromising others (e.g., "I would like something lighter and with faster processor speed"). The feedback, in turn, enables the system to more accurately predict what the user truly wants and accordingly recommend some products that may better interest the user in the next conversational cycle. One typical supporting mechanism, so called *system-suggested critiquing*, pro-actively generates a set of knowledge-based critiques that users may be prepared to accept as ways to improve the current product, which has been adopted in the FindMe system for proposing unit critiques [4] and in the Dynamic-Critiquing agent for presenting compound critiques [40]. An alternative critiquing mechanism provides a facility to stimulate users to freely create critiques on their own, which was implemented in our Example Critiquing agent [38]. Previous works proved that the critiquing support allows users to obtain higher decision accuracy and preference certainty, in comparison with non-critiquing based systems such as the ranked list.

Related works on decision support. In addition to the works mentioned above, some systems have also aimed at supporting users' decision in high-risk product domains, that include *knowledge-based recommenders* [13], *case-based reasoning systems* [3,4], and *utility-based recommenders* [21,45]. For example, in [13], the product's matching degree with the user's preferences was determined via a knowledge base which comprises the domain restrictions (e.g., "the camera with higher optical zoom should be preferred to the one with lower zoom"). The recommendation problem was then transformed into the constraint satisfaction problem, for which the user's preferences were formulated as hard and/or soft constraints. In case-based reasoning systems, the recommended products were retrieved via searching for ones that are most similar to what the user preferred before. For instance, in [4], the recommendation of restaurants in a new city is based on what the user knows and likes in other cities. The utility-based recommender system was explicitly built on the Multi-Attribute Utility Theory (MAUT) to model the user's preferences as a weighted additive form under the assumption of additive independence [21]. The products with higher utility scores are recommended to the user.

However, though it is feasible to obtain the current buyer's preferences via the aforementioned elicitation techniques, our previous empirical studies pointed out that the elicited user preferences are unlikely certain and complete in complex and unfamiliar conditions [38,7]. It is thus ineffective to purely base what the user stated to determine the product's true matching degree. Moreover, the related decision support systems have primarily used the product's static specifications to model the current buyer's preferences, while neglecting the potential usefulness of incorporating other users' generated contents like product reviews (see Fig. 1).

2.2. Review-based Recommender Systems

The literature survey on this topic showed that most works have mainly utilized product reviews, as a kind of auxiliary resource, to enhance the traditional rating-based recommender systems (being oriented to experienced, low-risk product domains). According to the level of review information that they exploited, we can classify the related methods into two branches: *review-level analysis* and *feature-level analysis*.

Review-level analysis for recommendation. This branch of work has been primarily based on the review's document-level analysis results. For instance, in [16], the keywords extracted from reviews were used to generate documents respectively for the corresponding reviewer and the item. The user document was then treated as a query to search for items that are most relevant in terms of the text similarity between the user document and the item document (through TF/IDF based cosine similarity measure). Another similar work is from [46] that captured the user–user similarity by taking into account their posted reviews' text similarity (via the latent semantic analysis (LSA)). On the other hand, the sentimental classification technique (also called document-level opinion mining) has been developed with the goal of returning an overall opinion value (i.e., positive, negative, or neutral) for a review document [33,51]. For example, Pang et al. examined how to apply machine learning tools to address the sentiment classification problem in movie review data [33]. From the recommender's perspective, the target was then how to best employ such algorithms' outputs to handle the recommendation issues [29,36,56]. As a typical example, considering the lack of real ratings in some practical scenarios, Zhang et al. converted each review to a virtual score (e.g., −1 for negative, 0 for neutral, and 1 for positive) through the sentiment classification technique, which was subsequently taken as the input to the standard collaborative filtering (CF) algorithm [56].

Feature-level analysis for recommendation. However, the review-level analysis works did not go into exploring the actual value of specific opinions associated with different features, such as the reviewer's opinions expressed on the hotel's service, cleanliness, etc. Thus, lately, some researchers have started to rely on the feature-level opinion mining results to enhance the recommendation accuracy. For example, in [22], the authors employed a multi-relational matrix factorization (MRMF) method, which is an extension to low-norm matrix factorization, to model the correlations among users, movies and the opinions regarding specific features. In [55], a weighted directed graph was established to connect products, for which the edge indicates the comparative feature opinions specified between two or more products (which were called "comparative opinions" in [15]). The PageRank algorithm was then applied to rank products in this graph. Li et al. employed the probabilistic latent relation model (PLRM) to represent the reviewer's feature opinions in certain contexts (e.g., location, time), for predicting how likely s/he will like a new restaurant [31]. In [18], a review-based hotel recommender system was developed by using the labeled Latent Dirichlet Allocation (LDA) method to infer all "trip intents" related to a hotel from its set of reviews. The utility of the hotel to the current user was then computed via a linear weighted additive function to sum up these trip intents' suitability scores (each score indicates the suitability of a trip intent to the current user's need). Levi et al. further considered the reviewer's nationality in addition to the feature-associated opinions for hotel recommendation [30]. It concretely adopted an unsupervised clustering algorithm to construct a vocabulary of hotel aspects, based on which the associated opinion wearing words were extracted through the popular feature-based opinion mining tool [20]. The relative weights on these aspects were adjusted for a specific hotel, considering its associated trip intents and reviewers' nationalities, which were then mapped to the current user's preferences and background.

To the best of our knowledge, few works have utilized the feature-level review opinions to recommend high-risk products (see Table 1). The work most related to ours is [1], which

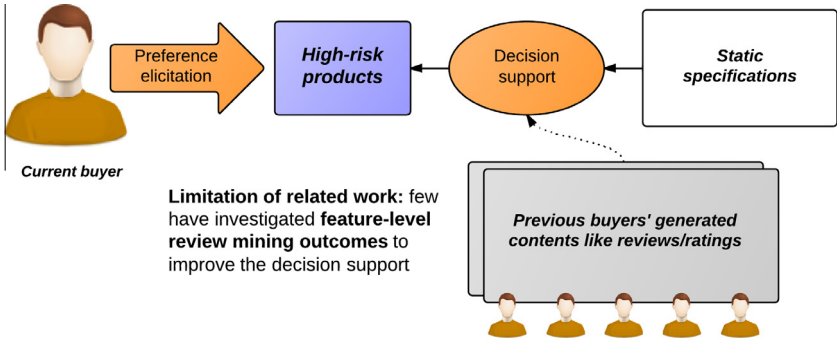

Fig. 1. Limitation of related recommender systems.

Table 1
Summary of related works on review-based recommender systems.

Citation	Product domain (low risk or high risk)	Level of review info utilized	User's preference structure	Served user (who receives recommendation)	Limitations
Esparza et al. [16]	Low risk (*movies, books, applications and games*)	Text words	Keyword vector	Reviewer (i.e., repeated user)	1. Require a certain amount of reviews provided by the target user, so cannot address "data sparsity" and "new user" problems; 2. Not consider feature-level opinions
Terzi and Whittle [46]	Low risk (*movies*)	Text words	Rating on items	Reviewer (i.e., repeated user)	Same as above
Zhang et al. [56]	Low risk (*movies*)	Document-level opinion	Rating on items	Reviewer (i.e., repeated user)	Same as above
Jakob et al. [22]	Low risk (*movies*)	Feature-level opinions	Latent factors	Reviewer (i.e., repeated user)	Require a certain amount of reviews provided by the target user
Zhang et al. [55]	High risk (*digital cameras and televisions*)	Comparative feature-level opinions	N.A.	New user	1. Not consider the user's preferences over multiple features; 2. assume that the user's stated preferences are complete
Li et al. [31]	Low risk (*restaurants*)	Contextual feature-level opinions	Rating on items	Reviewer (i.e., repeated user)	Require a certain amount of reviews provided by the target user
Hariri et al. [18]	Low risk (*hotels*)	Contextual feature-level opinions	Rating on items plus trip intent	Reviewer (i.e., repeated user)	Same as above
Levi et al. [30]	Low risk (*hotels*)	Contextual feature-level opinions	Multi-attribute preference model	New user	Assume that the user's stated preferences are complete
Aciar et al. [1]	High risk (*digital cameras*)	Feature-level opinions	Multi-attribute preference model	New user	Same as above

adopted the quantified sentiments on the camera's features, as a type of value function, to compute the product's total utility. Specifically, suppose F is the set of features extracted from a camera's reviews, the formula for computing the utility of a camera is $\sum_{f\in F} w_f * \frac{\sum O_{if} * Expertise_i}{NumberOfReviews}$, where w_f is the weight preference stated by the current buyer on feature f, O_{if} is the feature f's opinion in the *i-th* review, and $Expertise_i$ is the expertise level of reviewer who posted the *i-th* review. However, the limitation of this work is that they did not address the new buyers' "incomplete preferences" phenomenon that typically appears in high-risk product domains. Moreover, they did not empirically test whether users' decision accuracy could be actually improved in their developed system.

In Table 1, we summarize these related review-based recommenders from several aspects: (1) the focused product domain, (2) the level of utilized review information, (3) the served user, (4) the user's preference structure, and (5) major limitations.

3. Our system's workflow

With the existing preference elicitation techniques (see Section 2.1), the current new buyer's preferences over product features can be elicited and modeled based on MAUT: $Pref_u = \{\langle f_i, w_{ui}\rangle | 1 \leqslant i \leqslant n\}$, but the fact is that the elicited preferences are likely incomplete (so i can be any number in the range $[1,n]$; for example, the buyer just stated preferences on features f_2, f_3, f_6). Thus, in order to generate accurate recommendation to the buyer, our core idea is to identify her/his inherent preference similarity to product reviewers. The research problems that we have been engaged in solving are: (1) recovering reviewers' multi-feature preferences from the review information that they provided; (2) building the preference relevance between the current buyer and reviewers; and (3) predicting the buyer's full preferences and making recommendation.

As discussed in the introduction, purely counting the feature's occurrence frequency in a review cannot truly reflect its weight for a reviewer. It is neither straightforward to adopt some text analysis techniques (such as Latent Semantic Analysis (LSA)) because they cannot incorporate features' opinion values into deriving the reviewer's weight preferences. Therefore, more sophisticated learning method, that can well take into account both reviewers' overall ratings and feature-level opinions, should be investigated. Moreover, given that a single reviewer's generated information is limited, the developed method should involve multiple reviewers for revealing their preference similarity and build-

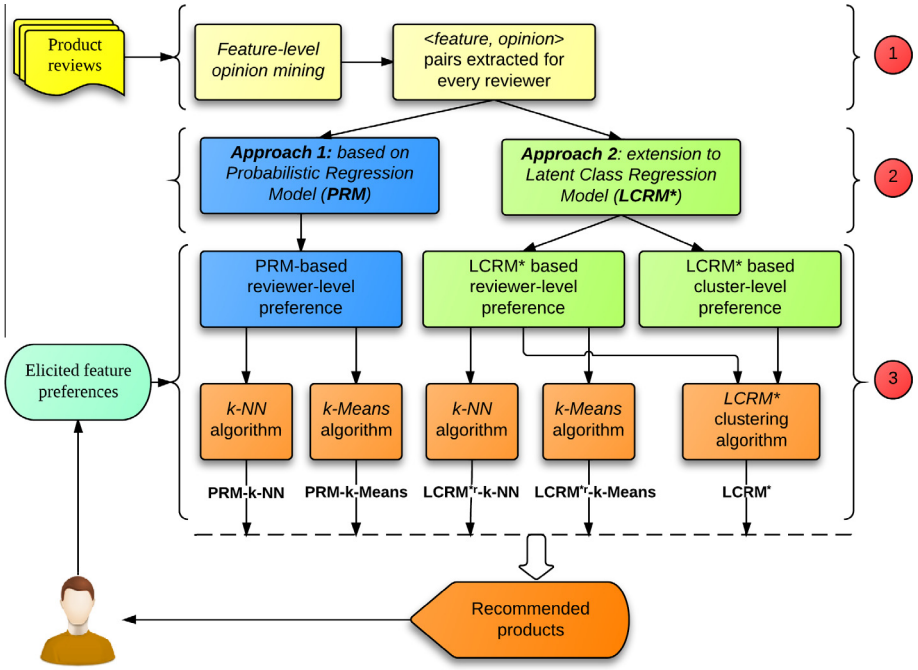

Fig. 2. System's workflow with two major branches of proposed approaches, which result in five different recommending methods.

ing cluster-level preferences. In the following, we first give our system's workflow, and then in detail describe the methods that we have developed.

3.1. System workflow

The workflow of our system mainly consists of three steps (see Fig. 2).

(1) The first step is conducting *feature-level opinion mining* to identify ⟨*feature*,*opinion*⟩ pairs from every review, where *opinion* indicates positive, neutral, or negative sentiment that a reviewer expressed on a *feature*. In this step, we have attempted to improve current opinion mining (or called sentiment analysis) methods [20,9], with the aim to fully exploit the values of review features and their associated opinions for deriving reviewers' weight preferences.

(2) Then, our primary focus is on inferring the reviewers' weight preferences. In this step, we have developed two alternative approaches. Particularly, in addition to building the probabilistic regression model (PRM-based approaches; see Section 4), we have investigated the effect of the latent class regression model (LCRM) on enhancing the stability of single reviewer's preferences (called *reviewer-level*) and producing *cluster-level* preferences for a group of reviewers simultaneously. Principally, four review elements are integrated into this model: the reviewer's overall rating on a product; the opinion associated with each feature in the review; the feature's occurring frequency (as a type of prior knowledge to be modeled); and the product that the reviewer commented.

(3) Resulting from the Step (2), there are three types of outcomes: PRM-based reviewer-level preferences, LCRM* based reviewer-level preferences, and LCRM* based cluster-level preferences (see Fig. 2). We have accordingly implemented different recommendation methods. With the reviewer-level preferences, we have tried (1) k-nearest neighbor algorithm (k-NN) for locating a group of reviewers who have similar weight preferences to the current buyer (shorted as PRM-k-NN and LCRM^{*r}-k-NN); and (2) k-Means algorithm for clustering reviewers based on their respective preferences and then identifying a cluster which is most relevant to the current buyer (shorted as PRM-k-Means and LCRM^{*r}-k-Means). With the cluster-level preferences as resulted from LCRM*, we have directly used it to perform the clustering. Thus, there are in total five different recommendation methods (see Fig. 2). Every recommendation method can return top-N products, with which our evaluation task is to test whether the current buyer's target choice (i.e., the product that s/he is prepared to buy) can be located in the list of N recommended products or not.

Table 2 lists the notations that will be used throughout the paper.

3.2. Pre-process: extracting feature-opinion pairs from product reviews

Before deriving reviewers' weight preferences, we need to first analyze their raw textual reviews and convert them into structured ⟨*feature*,*opinion_value*⟩ pairs. Previously, we have compared different learning methods for mining the feature-level opinions from reviews, and identified the respective advantages of model-based and statistical approaches [9]. Thus, inspired by prior findings, in the current system, we have concretely carried out three steps for identifying the feature-opinion pairs, which include: (1) extracting features from a review and grouping synonymous features; (2) locating opinions that are associated with various features in the review; and (3) quantifying the opinion value in the normalized range [1,6].

Specifically, to identify the prospective feature candidates, we used a Part-of-Speech (POS) tagger included in the

Table 2
Notations used in the formulas.

Notation	Description
$REV = \{rev_1, \ldots, rev_M\}$	A set of M reviewers
$\mathcal{P} = \{p_1, \ldots, p_{\lvert\mathcal{P}\rvert}\}$	A set of $\lvert\mathcal{P}\rvert$ products
$\mathcal{S} \subseteq REV \times \mathcal{P}$	A set of reviewer-product pairs, where $(rev_i, p_j) \in S$ indicates that a reviewer rev_i posted a review to product p_j
$\mathcal{F} = \{f_1, \ldots, f_n\}$	The n distinct features extracted from all reviews
r_{ij}	The review written by reviewer rev_i to product p_j
R_{ij}	The overall rating reviewer rev_i gave to product p_j
$\boldsymbol{X}_{ij} = [x_{ij1}, \ldots, x_{ijn}]$	The opinion values related to the features' set $\mathcal{F}$ in review r_{ij}
$\boldsymbol{W}_{rev_i} = [w_{i1}, \ldots, w_{in}]$	The reviewer rev_i's weight preferences, where w_{il} is the weight on feature $f_l \in \mathcal{F}$, which can be *None* if the reviewer did not express opinion on that feature
$\mathcal{C} = \{c_1, \ldots, c_K\}$	The K clusters of reviewers
$\boldsymbol{W}_{c_k} = [w_{c_k1}, \ldots, w_{c_kn}]$	The cluster c_k's preferences where w_{c_kl} is the cluster-level weight preference on feature $f_l \in \mathcal{F}$
$\boldsymbol{z} = \{z_1, \ldots, z_M\}$	The cluster membership of M reviewers, with $z_i = k$ denoting that reviewer rev_i belongs to cluster c_k

Note: Different notations are used for distinguishing "reviewers" and "the current buyer", i.e., rev_i ($1 \leqslant i \leqslant n$) for the ith reviewer and u for the current buyer.

Core-NLP package[1] to extract the frequent nouns (and noun phrases). Moreover, considering that reviewers often use different words for the same product feature (e.g., "picture" and "appearance" for "image"), we manually defined a set of seed words and then grouped the synonymous features by computing their lexical similarity to the seed words. The lexical similarity was concretely determined via WordNet [14]. Given that unsupervised methods normally take the risk of returning inaccurate results while supervised methods often require demanding human-labeling efforts, our approach was targeted to achieve the ideal balance between accuracy and effort. The previous experiment showed that such procedure can help identify reliable feature expressions in review text and effectively group them [9].

We then located all opinions which are associated with each feature in a review sentence. Most of existing works depended on the co-occurrence of product features and opinion bearing words for this purpose [20,37]. However, these methods cannot identify opinions that are not so close to a feature. Therefore, we took advantage of a syntactic dependency parser,[2] because it can return the syntactic dependency relations between words in a sentence. For example, after parsing the sentence "*it takes great photos and was easy to learn how to use*", "great" is identified with dependency relation AMOD with "photos" (meaning that "great" is an adjectival modifier of the noun word "photos"), and *"easy"* has COMP dependency relation with *"learn"* (indicating that "easy" is an open clausal complement of "learn"). In another example "*the photos are great*", "great" has NSUBJ relation with "photos" (suggesting that "photos" is the subjective of "great"). Thus, any words that are with such dependency relations with a feature are taken as opinion words.

Note that there might be some noisy information contained in the review text such as grammatical and syntactical errors, that is why we chose the publicly recognized POS tagger. Moreover, the syntactic dependency parser we employed is based on probabilistic model, that means it can be inherently able to handle the noisy inputs because it produces the most likely analysis to a sentence after checking all possibilities. Actually, the experiment proved that it can well recover the structure of noisy sentences [24]. Besides, we conducted a cleaning process to correct the misspelled words in review text via a statistical spell checker[3] and remove the duplicate, unnecessary punctuation marks in sentences (e.g., "!!!" that appears at the end of some sentences).

Near the end of this step, we need to assess every opinion word's sentiment strength (also called polarity value). For this task, we applied SentiWordNet [12] because it can provide with a triple of polarity scores for each opinion word: positivity, negativity and objectivity, respectively denoted as $Pos(s)$,$Neg(s)$, and $Obj(s)$, for the word s. Each ranges from 0.0 to 1.0, and $Pos(s) + Neg(s) + Obj(s) = 1$. The triple scores can then be merged into a single sentiment value: $O_s = Neg(s) * R_{min} + Pos(s) * R_{max} + Obj(s) * \frac{R_{min}+R_{max}}{2}$ (where R_{min} and R_{max} represent the minimal and maximal scales respectively. We set them as $R_{min} = 1$ and $R_{max} = 5$, so that O_s ranges from 1 to 5). In addition, we considered negation words (such as "not", "don't", "no", "didn't"): if the odd number of such words appears in a sentence, the polarity of related opinion will be reversed. Then, in the case that there are multiple opinion words associated with a feature in a review, we calculated a weighted average for which the opinion word's sentiment value behaves as the weight, so that the extremely positive or negative polarization is less susceptible to shift. For instance, if two opinion words "good" and "great" are both associated with a feature, the feature's final opinion value is: $\frac{4\times4+5\times5}{4+5} = 4.55$, where 4 and 5 are the sentiment values of the two words "good" and "great" respectively.

4. Approach 1: probabilistic regression model (PRM) based recommendation

After extracting the pairs $\langle feature, opinion\_value \rangle$ from every review, our focus is then on deriving corresponding reviewer's weighted feature preferences. The first approach we have developed is based on Probabilistic Regression Model (PRM) [54] to learn the weights for individual reviewers, i.e., to generate *reviewer-level preferences*.

Specifically, we treat the relationship between the overall rating that a reviewer assigned to a product and her/his opinion values being associated with the product's features as a regression problem. More formally, a reviewer's (rev_i) overall rating R_{ij} on a product p_j can be considered as a dependent variable, being the function of a set of independent opinion values $\boldsymbol{X}_{ij}$ in respect of the set of features $\mathcal{F}$. The regression coefficients can then be interpreted as the weight preferences of the reviewer $\boldsymbol{W}_{rev_i}$ because they essentially define the relative contributions of various features to determine the overall rating:

$$R_{ij} = \widehat{R}_{ij} + \varepsilon = \boldsymbol{W}_{rev_i}^T \boldsymbol{X}_{ij} + \varepsilon \tag{1}$$

where ε is a noise term.

Since the overall rating R_{ij} and opinion values $\boldsymbol{X}_{ij}$ are known, we can derive the weight preferences $\boldsymbol{W}_{rev_i}$ via Bayesian treatment because it can incorporate additional information, such as prior knowledge, to improve the model. Concretely, the noise term ε is drawn from a Gaussian distribution with zero as the mean:

$$\varepsilon \sim \mathcal{N}(0, \sigma^2) \tag{2}$$

in which σ^2 is the variance parameter that controls the model's precision. The conditional probability that a reviewer rev_i gives the overall rating R_{ij} to a product p_j can hence be defined as follows:

[1] http://nlp.stanford.edu/software/corenlp.shtml.
[2] http://nlp.stanford.edu/software/lex-parser.shtml.
[3] http://norvig.com/spell-correct.html.

$$Pro(R_{ij}|\boldsymbol{X}_{ij}, \boldsymbol{W}_{rev_i}) = \mathcal{N}\left(R_{ij}|\boldsymbol{W}_{rev_i}^T \boldsymbol{X}_{ij}, \sigma^2\right) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\left(R_{ij} - \boldsymbol{W}_{rev_i}^T \boldsymbol{X}_{ij}\right)^2}{2\sigma^2}\right) \quad (3)$$

According to the Bayes theory, the posterior probability of $\boldsymbol{W}_{rev_i}$ can be defined as the product of Eq. (3) and the prior probability:

$$Pro(\boldsymbol{W}_{rev_i}|\mathcal{S}) \propto \prod_{(rev_i,p_j)\in S} Pro(R_{ij}|\boldsymbol{X}_{ij}, \boldsymbol{W}_{rev_i}) \times Pro(\boldsymbol{W}_{rev_i}|\mu, \Sigma) \times Pro(\mu, \Sigma) \quad (4)$$

where $\mathcal{S}$ denotes the set of reviewer-product pairs (see Table 2). $Pro(\boldsymbol{W}_{rev_i}|\mu, \Sigma)$ is the prior probability of $\boldsymbol{W}_{rev_i}$ which can be drawn from a multivariate Gaussian distribution with μ as the mean and Σ as the covariance matrix:

$$Pro(\boldsymbol{W}_{rev_i}|\mu, \Sigma) \sim \mathcal{N}(\mu, \Sigma) \quad (5)$$

We further incorporate the feature's occurrence frequency, as the prior knowledge (μ_0), into $\mathcal{N}(\mu, \Sigma)$. The prior probability of the distribution $\mathcal{N}(\mu, \Sigma)$ is consequently formulated as:

$$Pro(\mu, \Sigma) = \exp(-\psi \cdot KL(\mathcal{N}(\mu, \Sigma)|\mathcal{N}(\mu_0, \boldsymbol{I}))) \quad (6)$$

where $KL(\cdot|\cdot)$ is the KL-divergence for calculating the difference between two distributions $\mathcal{N}(\mu, \Sigma)$ and $\mathcal{N}(\mu_0, \boldsymbol{I})$, and ψ is the trade-off parameter (which is default set as 50 in the experiment) to control the strength of μ_0 in the model.

We then apply the expectation–maximization (EM) algorithm [10] to identify the optimal values for the set of parameters $\Psi = \{\boldsymbol{W}_{rev_1}, \ldots, \boldsymbol{W}_{rev_M}, \mu, \Sigma, \sigma^2\}$ which contain the reviewer's weight preferences $\boldsymbol{W}_{rev_i}$:

$$\Psi^* = \arg\max_{\Psi} \mathcal{L}(\Psi|S) = \sum_{(rev_i,p_j)\in S} \log(Pro(\boldsymbol{W}_{rev_i}|S)) \quad (7)$$

4.1. Generating recommendation

4.1.1. PRM-based recommendation via k-NN (PRM-k-NN)

Given individual reviewers' weight preferences, the intuitive way to generate recommendation is based on the k-nearest neighbor (k-NN) method [43]. That is, we can first identify a group of k reviewers who have similar feature preferences to the current buyer, and then locate recommendations from products that were highly rated by those similar reviewers. Formally, suppose the elicited preferences from the current buyer are $\boldsymbol{W}_u$, her/his similarity to a reviewer can be computed via:

$$sim(\boldsymbol{W}_u, \boldsymbol{W}_{rev_i}) = \frac{1}{1 + \sqrt{\sum_{w_{f_l}\in \boldsymbol{W}_u} (w_{f_l}(u) - w_{f_l}(rev_i))^2}} \quad (8)$$

where $w_{f_l}(u)$ is the current buyer's weight preference on feature f_l, and $w_{f_l}(rev_i)$ is the ith reviewer's. We then retrieve k reviewers who are with higher similarity scores to the current buyer (k is optimally set through the experiment; see Section 6), and get a pool of products that were rated by these reviewers. Each product p_j from the pool is finally calculated with a prediction score to indicate how much it would interest the current buyer:

$$PredictionScore(u, p_j) = \frac{\sum_{rev_i\in\mathcal{K}} sim(\boldsymbol{W}_u, \boldsymbol{W}_{rev_i}) \times R_{ij}}{\sum_{rev_i\in\mathcal{K}} sim(\boldsymbol{W}_u, \boldsymbol{W}_{rev_i})} \quad (9)$$

where $\mathcal{K}$ denotes the group of k similar reviewers, and R_{ij} is the rating that reviewer rev_i gave to the product (it is zero if the reviewer did not review it). The top-N products with higher prediction scores are returned to the buyer as the recommendation. In our experiment, we tested the algorithm's accuracy when N = 10 or 20.

4.1.2. PRM-based recommendation via k-Means (PRM-k-Means)

As an alternative recommending solution, we turn to conduct the clustering process at first for identifying clusters, each of which is composed of multiple reviewers who possess similar preferences among each other. The current buyer is then matched to the most relevant cluster, within which the k-nearest neighbor algorithm is further performed. Actually, the clustering process has been recognized as an effective way to increase the recommender system's efficiency [26,42]. For instance, the clustering based collaborative filtering (CF) system has been found obtaining comparable prediction accuracy to the basic CF approach, while achieving significantly higher efficiency [42]. We were thus motivated to perform the clustering within reviewers before locating similar ones to the current buyer. Specifically, through the classical clustering technique such as k-Means, during each round, a reviewer will be moved from one cluster to another if this process can minimize its distance to the cluster's centroid. The centroid is formally denoted as $\boldsymbol{W}_{c_k\_centroid}$, which is calculated by averaging the preferences of the cluster's currently contained reviewers. The distance is calculated via $1/sim(\boldsymbol{W}_{rev_i}, \boldsymbol{W}_{c_k\_centroid})$ where $sim(\boldsymbol{W}_{rev_i}, \boldsymbol{W}_{c_k\_centroid})$ is defined similar to Eq. (8).

Upon the clustering process ends, we get K disjoint clusters $\{c_1, \ldots, c_K\}$. The current buyer's preferences are then used to compute her/his distance to all clusters' centroids, and the cluster with the shortest distance is regarded most relevant to the buyer. Within this cluster, we conduct PRM-k-NN to retrieve k nearest neighbors and then calculate each candidate product's prediction score via Eq. (9). Still, top-N products with higher scores are recommended to the buyer.

4.2. Discussion

As mentioned in the introduction, for high-risk products, there is usually one or few reviews posted by every reviewer. Therefore, a single reviewer's provided information is very limited. This sparsity phenomenon might cause the overfitting problem in PRM-based methods, because the deviation of each reviewer's weight preferences is purely contingent on her/his own review(s). Moreover, according to PRM, the derived weight preferences $\boldsymbol{W}_{rev_i}$ tend to be around the mean μ of the Gaussian Multivariate distribution $\mathcal{N}(\mu, \Sigma)$ (see Eq. (5)). The outcomes can be subject to be biased towards the mean μ, and cannot fully reflect the reviewer's true preferences. As a result, the accuracy of PRM based k-NN and k-Means recommendation algorithms (PRM-k-NN and PRM-k-Means) would be impaired.

5. Approach 2: latent class regression model (LCRM) based recommendation

5.1. Background of LCRM

Given the above-mentioned limitation of PRM-based methods, as the follow-up work, we have aimed at more accurately estimating a reviewer's weight preferences by taking into account additional information, such as her/his inherent preference similarity to other reviewers. For this purpose, we have studied Latent Class Regression Model (LCRM) [52]. Historically, LCRM originated from the area of marketing for conducting the market segmentation task, which is targeted to divide the set of prospective consumers into relatively smaller groups for revealing their preference homogeneity. In the field of machine learning, LCRM has been regarded as a specific branch of mixture models [32], to mainly handle the data with regression characteristics. Specifically, LCRM assumes that the whole population can be defined by a finite number of distributions (every distribution represents a cluster of consumers in

the case of market segmentation), so its primary goal is to estimate each distribution's regression model *at the cluster level*. Therefore, LCRM does not require the knowledge of single entity's regression values (e.g., from a single consumer), but focuses on exploiting the whole population's structure to generate the clusters directly. Every entity is assigned to a cluster only when this assignment has the highest membership probability.

Due to these properties, we believe it might be useful to address the limitation of PRM-based methods. Because the standard LCRM can be used to only produce the cluster-level preferences (i.e., the cluster-level regression model that represents a group of entities which are "reviewers" in our case), we have extended it to LCRM*, in order to derive every reviewer's preferences simultaneously. The concrete idea of LCRM* is that the cluster-level preferences might be leveraged to polish individual reviewers' weight preferences, so that not only the reviewer's own information is taken into account in this regard, but her/his inherent similarity to other reviewers can be incorporated together so as to stabilize her/his preferences. The advantage of LCRM* against LCRM is thus that it can potentially uncover the *preference heterogeneity* among reviewers within a cluster.

5.2. LCRM*: deriving both cluster-level and reviewer-level feature preferences

In this section, we in detail describe how the LCRM is extended to achieve the above-mentioned objectives. In Section 5.3, we will explain how the results are further applied to generate recommendation. Concretely, four types of review elements are unified in this model: the reviewer's overall rating on a product; the opinion associated with each feature in the review; the feature's occurring frequency as a type of prior knowledge; and the product that the reviewer commented.

According to the basic LCRM model, we first assume that all reviewers can be divided into K clusters $\mathcal{C} = \{c_1, c_2, \ldots, c_K\}$. The likelihood probability function for the overall rating R_{ij} (the dependent variable) is defined as:

$$Pro(R_{ij}|\boldsymbol{X}_{ij}, \Phi) = \sum_{k=1}^{K} \pi_k Pro(R_{ij}|\boldsymbol{X}_{ij}, c_k) \tag{10}$$

where Φ denotes the set of all parameters, π_k denotes the prior probability of cluster c_k, and $\boldsymbol{X}_{ij}$ is the vector of opinion values associated with features $\mathcal{F}$ w.r.t. reviewer rev_i. In the above formula, the right component $Pro(R_{ij}|\boldsymbol{X}_{ij}, c_k)$ gives the conditional probability of the overall rating R_{ij} in the case that reviewer rev_i belongs to the cluster c_k:

$$Pro(R_{ij}|\boldsymbol{X}_{ij}, c_k) = Pro(R_{ij}|\boldsymbol{X}_{ij}, \boldsymbol{W}_{rev_i}) \cdot Pro(\boldsymbol{W}_{rev_i}|c_k) \tag{11}$$

in which $\boldsymbol{W}_{rev_i} \in \mathbb{R}^n$ denotes the reviewer rev_i's weight preferences, and $Pro(R_{ij}|\boldsymbol{X}_{ij}, \boldsymbol{W}_{rev_i})$ gives the likelihood of $\boldsymbol{W}_{rev_i}$ given the overall rating R_{ij} and features' opinion vector $\boldsymbol{X}_{ij}$. Here, we can assume that each reviewer's preference is drawn from the cluster-level preference distribution, which can be a Multivariate Gaussian with $\boldsymbol{W}_{c_k}$ (the cluster-level preferences) as the mean and Σ_k as the covariance matrix:

$$Pro(\boldsymbol{W}_{rev_i}|c_k) = Pro(\boldsymbol{W}_{rev_i}|\boldsymbol{W}_{c_k}, \Sigma_k) \sim \mathcal{N}(\boldsymbol{W}_{rev_i}|\boldsymbol{W}_{c_k}, \Sigma_k) \tag{12}$$

Furthermore, the uncertainty of the distribution of the cluster-level preferences $\mathcal{N}(\boldsymbol{W}_{rev_i}|\boldsymbol{W}_{c_k}, \Sigma_k)$ can be modeled based on the KL-divergence as follows:

$$Pro(\boldsymbol{W}_{c_k}, \Sigma_k) = \exp(-\psi \cdot KL(\mathcal{N}(\boldsymbol{W}_{c_k}, \Sigma_k)|\mathcal{N}(\mu_0, \boldsymbol{I}))) \tag{13}$$

where μ_0 is the set of occurrence frequencies of features in the reviews.

Because the overall rating R_{ij} is known, the probability that a reviewer belongs to a cluster can hence be estimated. Formally, a reviewer rev_i is placed into a cluster c_k if $q_k(rev_i) > q_h(rev_i)\ \forall c_k \neq c_h$, where

$$q_k(rev_i) = \prod_{(rev_i, p_j) \in \mathcal{S}} \frac{\pi_{jk} \cdot Pro(R_{ij}|\boldsymbol{X}_{ij}, c_k)}{\sum_{c_h \in \mathcal{C}} \pi_{jh} \cdot Pro(R_{ij}|\boldsymbol{X}_{ij}, c_h)} \tag{14}$$

In addition, it is reasonable to assume that reviewers who commented the same product should be more preference relevant. Thus, with her/his commented product p_j, the distribution $\pi_j = \{\pi_{j1}, \ldots, \pi_{jK}\}$ can be modeled as the prior probability that the reviewer rev_i belongs to a certain cluster. The full mixture log-likelihood with all observations $\mathcal{S}$ (i.e., the collection of reviewer-product pairs) can hence be defined as:

$$\mathcal{L}(\Phi|\mathcal{S}) = \sum_{(rev_i, p_j) \in \mathcal{S}} \log \left(\sum_{k=1}^{K} \pi_k Pro(R_{ij}|\boldsymbol{X}_{ij}, c_k) \right) \tag{15}$$

We further derive the following two formulas (Eqs. (16) and (18)), respectively for inferring cluster-level preferences and reviewer-level preferences:

$$\widehat{\boldsymbol{W}}_{c_k} = \left(N_k \Sigma_k^{-1} + \psi \cdot \boldsymbol{I} \right)^{-1} \left(\Sigma_k^{-1} \sum_{z_i = k}^{M} \boldsymbol{W}_{rev_i} + \psi \cdot \boldsymbol{I} \cdot \mu_0 \right) \tag{16}$$

where

$$\widehat{\Sigma}_k = \left[\left[\frac{1}{\psi} \sum_{z_i = k}^{M} (\boldsymbol{W}_{rev_i} - \boldsymbol{W}_{c_k})(\boldsymbol{W}_{rev_i} - \boldsymbol{W}_{c_k})^T + \left(\frac{N_k - \psi}{2\psi} \right)^2 \boldsymbol{I} \right]^{\frac{1}{2}} - \frac{N_k - \psi}{2\psi} \boldsymbol{I} \right]^T \tag{17}$$

$$\widehat{\boldsymbol{W}}_{rev_i} = \frac{1}{N(rev_i)} \sum_{(rev_i, p_j) \in S} \left(\frac{\boldsymbol{X}_{ij} \boldsymbol{X}_{ij}^T}{\sigma^2} + \Sigma_k^{-1} \right)^{-1} \left(\frac{(R_{ij} - \boldsymbol{W}_{rev_i}^T \boldsymbol{X}_{ij})}{\sigma^2} + \Sigma_k^{-1} \boldsymbol{W}_{c_k} \right) \tag{18}$$

in which $N(rev_i)$ is the number of reviews posted by reviewer rev_i.

The parameters' set $\Phi = \{z_1, \ldots, z_M, \boldsymbol{W}_{c_1}, \ldots, \boldsymbol{W}_{c_K}, \Sigma_1, \ldots, \Sigma_K, \boldsymbol{W}_{rev_1}, \ldots, \boldsymbol{W}_{rev_M}\}$ is then estimated through the expectation–maximization (EM) algorithm, which seeks to identify the maximal log-likelihood by iteratively performing the following two steps.

- **Expectation step (E-step):** In this step, with individual reviewer's preferences $\boldsymbol{W}_{rev_i}$, we aim to update the reviewer's cluster assignment, the cluster-level preference distribution, and the prior probability of clusters.
  1. The cluster assignment z_i ($z_i = k$ if reviewer rev_i belongs to cluster c_k) is determined via:
  $$z_i = \arg\max_k q_k(rev_i) \tag{19}$$
  where $q_k(rev_i)$ is referred to Eq. (14). One reviewer is assigned to a cluster only when the highest probability is obtained.
  2. For each cluster, the cluster-level preferences $\boldsymbol{W}_{c_k}$ are updated using the Eq. (16).
  3. The prior probability of clusters (i.e., $\pi_j = \{\pi_{j1}, \ldots, \pi_{jK}\}$) can be treated as multinomial distribution and updated through the Laplace smoothing:
  $$\pi_{jk} = \frac{\sum_{(rev_i, p_j) \in S} \boldsymbol{1}_{z_i = k} + \lambda}{N(p_j) + K \times \lambda} \tag{20}$$
  in which $N(p_j)$ is the number of reviews posted to product p_j, and the scale variable $\lambda \in [0, 1]$ is the smoothing parameter.

- **Maximization step (M-step):** in this step, we aim to update each reviewer's preferences $\boldsymbol{W}_{rev_i}$ through Eq. (18) (with other parameters fixed).

E- and M-steps are repeated until the Eq. (15) converges. As the result, all reviewers are divided into K disjoint clusters, plus the cluster-level preferences $\boldsymbol{W}_{c_k}$ generated for each cluster and the reviewer-level preferences $\boldsymbol{W}_{rev_i}$ for every reviewer. It is worth mentioning that Eqs. (13)–(20) are our proposed extension to the basic LCRM. The major algorithm steps of LCRM* are shown in Fig. 3.

As for the algorithm's time complexity, the E-step costs $O(max(|\mathcal{S}|, n) * K * n^2)$ operations, and the M-step costs $O(K * n^3 + |\mathcal{S}|n^2)$ operations, where K is the number of clusters and n is the number of product features. Suppose LCRM* converges after t iterations, the computational complexity of LCRM* is $O(t * max(|\mathcal{S}|, n) * K * n^2)$. In comparison, because the probabilistic regression model (PRM) (see Section 4) requires to compute the determinant of the covariance matrix that takes $O(n^3)$ operations, its complexity is $O(t*M*n^3)$ in which t is still the number of iterations of EM steps and M is the number of reviewers.

5.3. Generating recommendation

To generate recommendation for the current buyer, we first classify him/her to the most relevant cluster of reviewers. The preference similarity between the buyer and a cluster is computed via:

$$Sim(\boldsymbol{W}_u, \boldsymbol{W}_{c_k}) = \frac{1}{1 + \sqrt{\sum_{i=1}^{n}(w_{f_i}(u) - w_{f_i}(c_k))^2}} \tag{21}$$

where $\boldsymbol{W}_u$ is the buyer's stated weight preferences and $\boldsymbol{W}_{c_k}$ denotes the cluster-level preferences of cluster c_k. The cluster with the highest similarity value is then chosen. Within it, we further retrieve k reviewers who are most similar to the current buyer based on their respective reviewer-level preferences. The formula for calculating the similarity between a reviewer and the current buyer can be referred to Eq. (8). Then, a pool that contains products rated by these k reviewers is generated, and a prediction score is computed for each product p_j to indicate its matching degree with the buyer's potential interests:

$$PredictionScore(u, p_j) = \frac{\sum_{rev_i \in c_l \cap \mathcal{K} \wedge (rev_i, p_j) \in \mathcal{S}} sim(\boldsymbol{W}_u, \boldsymbol{W}_{rev_i}) \times R_{ij}}{\sum_{rev_i \in c_l \cap \mathcal{K} \wedge (rev_i, p_j) \in \mathcal{S}} sim(\boldsymbol{W}_u, \boldsymbol{W}_{rev_i})} \tag{22}$$

where c_l denotes the most relevant cluster, $\mathcal{K}$ is the set of k nearest reviewers, R_{ij} is the overall rating that a reviewer gave to the product, and $sim(\boldsymbol{W}_u, \boldsymbol{W}_{rev_i})$ is the preference similarity between the buyer u and the reviewer rev_i. Top-N products with higher prediction scores are recommended to the buyer.

The time complexity of this step is $O(|c_l|^2 * |\mathcal{P}| * |\mathcal{K}| + |\mathcal{P}|^2)$, where $|\mathcal{P}|$ denotes the number of all products.

Data: The observations $\mathcal{S}$; each review is with the overall rating R_{ij} and the extracted feature-opinion vector $\boldsymbol{X}_{ij}$
Result: K disjoint clusters, along with the cluster-level preferences $\boldsymbol{W}_{c_k}$ for each cluster, and the reviewer-level preferences $\boldsymbol{W}_{rev_i}$ for each reviewer in the cluster
Start:
initialize reviewers' cluster assignment randomly;
initialize cluster-level preferences $\boldsymbol{W}_{c_k} \sim \mathcal{N}(0, \boldsymbol{I})$;
initialize reviewer-level preferences $\boldsymbol{W}_{rev_i} \sim \mathcal{N}(0, \boldsymbol{I})$;
while *Equation 15 does not converge* **do**
  // E-step;
  assign each reviewer to a cluster through Equation 14;
  update the cluster-level preferences $\boldsymbol{W}_{c_k}$ using Equation 16;
  update the covariance matrix Σ_k using Equation 17;
  update the prior probability of clusters using Equation 20;
  // M-step;
  update each reviewer's preferences $\boldsymbol{W}_{rev_i}$ using Equation 18;
end

Fig. 3. Algorithm steps of LCRM*.

5.4. Discussion: summary of developed methods

The main differences between the PRM-based approaches (i.e., PRM-k-NN and PRM-k-Means) and LCRM* can be found in Table 3. In contrast with PRM-based approaches, individual reviewers' preferences are derived inter-dependently in LCRM*. That is, both cluster membership and reviewers' commonly reviewed products are considered when adjusting each reviewer's preferences. Indeed, this algorithm attempts to explain every reviewer's behavior by involving her/his inherent similarity to some of other reviewers.

On the other hand, in comparison to the heuristic clustering methods such as k-Means, LCRM*, as a type of model-based clustering approach, does not require the pre-knowledge of reviewer-level preferences and the pre-definition of entity–entity distance metric. Therefore, its efficiency and accuracy might be higher, especially in the condition that the information provided by a single reviewer is very limited. As for the recommending procedure, LCRM* and PRM-k-Means both include the process of clustering reviewers, which might potentially help increase the algorithm's performance and prediction power, whereas PRM-k-NN might be limited as it purely relies on the buyer's stated preferences to locate similar reviewers.

In addition, in order to identify the specific effect of LCRM*-based method on generating *reviewer-level preferences*, we have developed two variants of LCRM*: k-NN recommendation based on LCRM* (shorted as LCRM*^{r}-k-NN) and k-Means recommendation based on LCRM* (shorted as LCRM*^{r}-k-Means). In these two methods, the cluster-level preferences as produced by LCRM* are not utilized. We compared them respectively to their counterparts, i.e., LCRM*^{r}-k-NN vs. PRM-k-NN, and LCRM*^{r}-k-Means vs. PRM-k-Means, because the two in each pair just differ in terms of inducing individual reviewers' preferences (with other steps identical). Moreover, we compared LCRM*^{r}-k-Means with LCRM* given their exclusive difference regarding the clustering process, so as to identify which clustering approach is more effective. Besides, it is worth mentioning that the original LCRM is not included in the experimental comparison, mainly because it can only produce cluster-level preferences, which however are not useful to carry out the user–user similarity measure and hence limit the recommendation process.

6. Experiment

6.1. Experiment setup and dataset

We prepared two real-world datasets for conducting the experiment: digital camera dataset and laptop dataset, which were crawled from a commercial website www.buzzillions.com. In both datasets, each textual review is accompanied by an overall rating that ranges from 1 to 5 stars as assigned by the corresponding reviewer. Before the experiment, we cleaned the datasets: (1) removing the reviews which contain less than 4 features (including the ones that are too short or with meaningless characters), and (2) removing the products that contain less than 10 reviews. The cleaning process ensured that each review contains a certain amount of information, and each product consists of sufficient reviews to be analyzed. After this step, the digital camera dataset has 112 cameras along with totally 18,251 reviews, and the laptop dataset has 155 laptops for which there are 6024 reviews in total.

Table 3
Summary of developed methods' properties.

		PRM-k-NN	PRM-k-Means	LCRM*
Deriving reviewer-level preferences	Leveraging cluster-level preferences	X	X	√
	Considering reviewers' commonly reviewed products	X	X	√
Recommending procedure	Clustering initiated	X	√	√

Table 4
Description of two tested datasets (digital camera and laptop).

	Digital camera	Laptop
Total reviewers	18,251	6024
Total products	122	155
Min. reviews per product	15	10
Average reviews per product	149.6 (st.d. = 171.835)	33 (st.d. = 34.475)
Max. reviews per product	1052	222
Min. features per review	4	4
Average features per review	5.16 (st.d. = 1.266)	5.03 (st.d. = 1.119)
Max. features per review	10	8

The analysis on both datasets shows that every reviewer posted only one review to a product, which is consistent with the statistics reported in [23,53]. The details of these two datasets are in Table 4.

Following the *leave-one-out* evaluation mechanism [39], we are able to simulate a set of "buyers" from the dataset. That is, a reviewer can be considered as a simulated "buyer", if s/he satisfies two criteria: (1) s/he only commented one product, and (2) her/his overall rating on the reviewed product is 5 (full star), indicating that s/he strongly likes this product. Therefore, this highly rated product can be taken as the simulated buyer's *target choice*, and the reviewer's feature preferences can be taken as the simulated buyer's full preferences. It turns out that 4002 reviewers satisfy these criteria in the camera dataset and 1330 reviewers in the laptop dataset. Therefore, at a time, one of these reviewers was excluded from the dataset to be a "buyer". The aim of the experiment was hence to measure whether the buyer's target choice can be located in the list of recommended items as returned by the tested algorithm. Moreover, in addition to performing the testing on the buyer's full preferences (i.e., over 10 features in the camera dataset and over 8 features in the laptop dataset), we randomly took out subsets of his/her preferences to represent the partial preferences that s/he may state in the real situation (e.g., preferences over 4, 6, or 8 features, out of full 10 features, in the camera dataset; and preferences over 4 or 6 features, out of full 8 features, in the laptop dataset). Thus, in total, there are $4002 * 4 = 16{,}008$ testings performed in the camera dataset, and $1330 * 3 = 3990$ testings performed in the laptop dataset. The number of features involved in each input preferences is called the corresponding buyer's *preference size*.

6.2. Compared baseline methods

In addition to the methods described in Sections 4 and 5, we also implemented two baselines: one does not consider product reviews (shorted as Non-Review); and another primarily utilizes reviews to perform product ranking, but does not attempt to derive the reviewers' feature preferences (shorted as Review-Rank) [1]. As mentioned in the related work (Section 2), except Review-Rank which is most similar to our focus, the other review-based recommender systems have been typically oriented to serve low-risk, frequently-experienced products (e.g., books, movies), so they were not included in our experiment.

6.2.1. Non-review based recommendation (Non-Review)

This is a baseline that does not incorporate reviews' feature-level opinions into computing recommendation. It is simply based on the product's static feature values to determine how much it matches to the buyer's stated preferences $\boldsymbol{W}_u$:

$$ProductScore(u, p_j) = \sum_{w_{f_l}(u) \in \boldsymbol{W}_u} w_{f_l}(u) \times s_{f_l}(p_j) \tag{23}$$

where p_j is the product, $s_{f_l}(p_j)$ is the utility of every feature f_l (which is normalized in the range from 0.0 to 1.0), and $w_{f_l}(u)$ denotes the current buyer's weight preference on feature f_l. More specifically, the feature's utility is computed by assessing the feature's fixed specification (e.g., for the camera's logical zoom, it is "the higher, the better", and for price, it is "the cheaper, the better"). The utility is also called value function in [25]. A default utility function was defined for each feature based on the domain knowledge.

As shown in Eq. (23), the utility of a feature is multiplied with the buyer's weight preference, and the weighted additive sum score that involve all features is then computed to indicate the product's satisfying degree. Top-N products with higher scores are displayed to the buyer in the list of recommendation.

6.2.2. Review-based product ranking (Review-Rank)

This approach extracted features and opinions from reviews, while its major difference from our methods is that it did not target to derive reviewers' multi-feature preferences from these extracted data. Actually, it is just based the feature's opinion value to calculate its utility:

$$FeatureScore_{f_l}(p_j) = \frac{\sum_{(rev_i, p_j) \in \mathcal{S}} x_{ijl}}{m} \tag{24}$$

where x_{ijl} denotes the feature f_l's opinion value in review r_{ij} w.r.t. product p_j, and m denotes the number of reviews to the product p_j. Therefore, it can be seen that the feature's score ($FeatureScore_{f_l}$) regarding a product p_j is computed by averaging all opinions that were associated with this feature in the product's reviews. Then, the product's satisfying score is calculated via:

$$ProductScore(u, p_j) = \sum_{w_{f_l}(u) \in \boldsymbol{W}_u} w_{f_l}(u) \times FeatureScore_{f_l}(p_j) \tag{25}$$

Still, top-N products with higher scores are recommended to the buyer.

6.3. Evaluation metrics

To choose appropriate evaluation metrics, we have considered Kendall's tau, Hit-Ratio and MRR (Mean Reciprocal Rank), because they have all been applied to measure the algorithm's accuracy in preference-based recommender systems [11,17]. However, Kendall's tau turns out being unsuitable for our case because it assumes that each user has multiple target choices, and aims to compute the similarity between the estimated preference ranking over these choices and the true preference ranking, while in our condition each user only has one target choice. Thus, we have finally decided to use Hit-Ratio and MRR as the metrics in our experiment.

Table 5
List of tested algorithms in the experiment.

Abbreviation	Algorithm description	Details
Non-review	Generating recommendation without considering product reviews	Section 6.2
Review-rank	Ranking products by taking into account features' review opinions	Section 6.2
PRM-k-NN	PRM-based deriving reviewer-level preferences and k-NN based generating recommendation	Section 4
PRM-k-Means	PRM-based deriving reviewer-level preferences and k-Means based generating recommendation	Section 4
LCRM∗	Extending LCRM to derive both reviewer-level and cluster-level preferences	Section 5
LCRM^{*r}-k-NN	LCRM*-based deriving reviewer-level preferences and k-NN based generating recommendation	Section 5.4
LCRM^{*r}-k-Means	LCRM*-based deriving reviewer-level preferences and k-Means based generating recommendation	Section 5.4

- *H@N* (Hit Ratio @ top-N recommendations) mainly measures whether the user's target choice appears in the set of N recommendations or not (in the experiment, N is set as 10 or 20). It concretely returns the percent of hits among all users:

$$H@N = \frac{\#\text{The number of successes within the top} - \text{N}}{\#\text{The total number of test cases}} \tag{26}$$

- *MRR* (Mean Reciprocal Rank) is a statistic measure for evaluating the ranking position of the target choice in the whole list:

$$MRR = \frac{\sum_{t=1}^{T}(0 + 1_{rank_t \leqslant 20})\frac{1}{rank_t}}{T} \tag{27}$$

in which T is the number of test cases, and $rank_t$ is the ranking position of the target choice when testing the t-th case. 1_{rank_t} is an indicator function that equals to 1 if $rank_t \leqslant 20$ (i.e., if the target choice appears in the top 20 products), and 0 otherwise.

6.4. Results analysis

In this section, we first show the results from comparing the three major methods: LCRM*, PRM-k-NN, PRM-k-Means.[4] We then present the results from testing two variants of LCRM*, i.e., LCRM^{*r}-k-NN and LCRM^{*r}-k-Means, in comparison with their counterparts. Finally, we identify the performance difference between LCRM* and LCRM^{*r}-k-Means, with the particular focus on their clustering process. A summary of these abbreviations' descriptions can be referred to Table 5.

6.4.1. Overall comparison

Tables 6 and 7 list the results in terms of both Hit-Ratio and MRR metrics on digital camera dataset and laptop dataset respectively. The superscript annotations in tables indicate the significant level from pair-wise comparisons. The concrete significance values (by t-test) are listed in Tables A.1 and A.2 in the Appendix.

First of all, it can be seen that PRM-k-NN, PRM-k-Means, and LCRM* all perform significantly better than the two baseline methods (i.e., Non-Review and Review-Rank) with respect to both metrics. Specifically, Review-Rank that simply utilizes features' opinion values to perform product ranking cannot compete with PRM-based methods and LCRM*, as the latter ones target at building the feature preferences based similarity relationship between the buyer and reviewers. Moreover, it shows that the outperforming accuracy of these *preference-based review-incorporated methods* is more obvious when the buyer's preferences are less complete (e.g., over 4 or 6 features in camera dataset) as shown in Tables 6 and 7, inferring that they can more accurately predict the buyer's un-stated preferences by relating her/him to like-minded reviewers.

Furthermore, LCRM* is shown more accurate than PRM-k-NN and PRM-k-Means on both datasets in most conditions. For example, when the buyer's preference is stated over 6 features in camera dataset (i.e., the preference size is 6), the Hit-Ratio achieved by LCRM* is 0.305 when N is 10, which is up to 45.2% higher than the one by PRM-k-Means, and 43.9% than the one by PRM-k-NN. The MRR value of LCRM* is also significantly higher than the ones of PRM-k-NN and PRM-k-Means. Combining the two metrics' results, we can infer that LCRM* not only increases the chance of including users' target choice in the recommendation list, but also ranking the target choice in top positions in the list. Moreover, the comparison between LCRM*/PRM-k-Means and PRM-k-NN reveals the positive impact of integrating the clustering process on identifying inherently more similar reviewers to the current buyer.

6.4.2. Evaluation on reviewer-level preferences

As mentioned in Section 5.4, in order to distinguish the particular effect of LCRM* on deriving reviewer-level preferences, we tested its two variants, LCRM^{*r}-k-NN and LCRM^{*r}-k-Means (referred to Table 5). The comparison between LCRM^{*r}-k-NN and PRM-k-NN indicates that the former is superior to the latter at varied sizes of the buyer's preferences, w.r.t. both Hit-Ratio and MRR (see Tables 6 and 7, and Fig. 4). For instance, LCRM^{*r}-k-NN achieves significantly higher *H*@10 than PRM-k-NN at preference size 6, e.g., 0.251 vs. 0.210, $t = 5.047$, $p < 0.01$ in camera dataset. The similar finding appears in the comparison between LCRM^{*r}-k-Means and PRM-k-Means. Taking *H*@10 and preference size @ 6 as an example, the Hit-Ratio of LCRM^{*r}-k-Means is 0.260 in camera dataset, which is significantly higher than the one of PRM-k-Means (that is 0.212; $t = 5.802$, $p < 0.01$; similar phenomena are shown in laptop dataset). These observations highlight the positive impact of extension that we made to the original LCRM. In order words, it is demonstrated that LCRM* is more accurate in terms of deriving single reviewer's feature preferences than PRM based methods.

In addition, from Fig. 4, we can verify again the effect of integrating clustering process on improving recommendation accuracy, since the clustering based LCRM^{*r}-k-Means performs better than the k-nearest neighbor based recommendation method LCRM^{*r}-k-NN in both datasets (the same appears in the comparison between PRM-k-Means and PRM-k-NN), although some differences do not reach at significant level. We were then driven to further compare LCRM* and LCRM^{*r}-k-Means, with the focus on their exclusive difference in respect of the clustering procedure, so as to identify which clustering method is more effective.

6.4.3. Evaluation on cluster-level preferences

LCRM* and LCRM^{*r}-k-Means differ only in the way of clustering reviewers and generating cluster-level preferences (with other steps identical between them): LCRM* is model-based, while LCRM^{*r}-k-Means performs the heuristic k-Means clustering. The comparative results show that LCRM*-based clustering method is more accurate than the k-Means based, given that both Hit-Ratio and MRR returned by LCRM* are significantly better in most cases

[4] For each method, the parameters' optimal values were first tuned through the experiment. In the digital camera dataset, the optimal number of clusters is 6 (i.e., $K = 6$) for LCRM*, LCRM^{*r}-k-Means and PRM-k-Means, and the optimal neighborhood size in all k-NN based methods is 15 (i.e., $k = 15$). In the laptop dataset, $K = 6$ and $k = 40$ for LCRM* and LCRM^{*r}-k-Means, $K = 8$ and $k = 15$ for PRM-k-Means, $k = 40$ for LCRM^{*r}-k-NN, and $k = 20$ for PRM-k-NN.

Table 6
Comparison of algorithms w.r.t. *Hit Ratio* and *MRR* with varied preference sizes in digital camera dataset (the maximal preference size is 10).

Preference size	Method	Evaluation metrics		
		H@10	H@20	MRR
4 Features	$_1$Non-Review	0.109	0.123	0.012
	$_2$Review-Rank	0.119^{1}	0.124^{1}	0.036^{1}
	$_3$PRM-k-NN	$0.207^{1,2}$	$0.258^{1,2}$	$0.041^{1,2}$
	$_4$PRM-k-Means	$0.201^{1,2}$	$0.256^{1,2}$	$0.059^{1,2,3}$
	$_5$LCRM*	$\mathbf{0.234}^{1,2,3,4}$	$0.274^{1,2,3}$	$0.060^{1,2,3}$
	$_6$LCRM*^{r}-k-NN	$0.234^{1,2,3,4}$	$0.291^{1,2,3,4}$	$0.061^{1,2,3,4}$
	$_7$LCRM*^{r}-k-Means	$0.234^{1,2,3,4}$	$0.292^{1,2,3,4}$	$0.061^{1,2,3,4}$
6 Features	$_1$Non-Review	0.124	0.146	0.016
	$_2$Review-Rank	0.117	0.136	0.038^{1}
	$_3$PRM-k-NN	$0.210^{1,2}$	$0.266^{1,2}$	$0.061^{1,2}$
	$_4$PRM-k-Means	$0.212^{1,2}$	$0.277^{1,2,3}$	$0.064^{1,2,3}$
	$_5$LCRM*	$\mathbf{0.305}^{1,2,3,4,6,7}$	$\mathbf{0.353}^{1,2,3,4,6,7}$	$0.067^{1,2,3,4,6}$
	$_6$LCRM*^{r}-k-NN	$0.251^{1,2,3,4}$	$0.304^{1,2,3,4}$	$0.064^{1,2,3}$
	$_7$LCRM*^{r}-k-Means	$0.260^{1,2,3,4}$	$0.307^{1,2,3,4,}$	$0.068^{1,2,3,4,6}$
8 Features	$_1$Non-Review	0.124	0.146	0.029
	$_2$Review-Rank	0.114	0.131	0.048^{1}
	$_3$PRM-k-NN	$0.220^{1,2}$	$0.269^{1,2}$	$0.059^{1,2}$
	$_4$PRM-k-Means	$0.226^{1,2,3}$	$0.288^{1,2,3}$	$0.068^{1,2,3,6}$
	$_5$LCRM*	$\mathbf{0.360}^{1,2,3,4,6,7}$	$\mathbf{0.316}^{1,2,3,4,6,7}$	$\mathbf{0.070}^{1,2,3,6,7}$
	$_6$LCRM*^{r}-k-NN	$0.249^{1,2,3,4}$	$0.298^{1,2,3,4}$	$0.064^{1,2,3}$
	$_7$LCRM*^{r}-k-Means	$0.281^{1,2,3,4,6}$	$0.309^{1,2,3,4,6}$	$0.067^{1,2,3}$
10 Features	$_1$Non-Review	0.110	0.121	0.023
	$_2$Review-Rank	0.120^{1}	0.138^{1}	0.039^{1}
	$_3$PRM-k-NN	$0.215^{1,2}$	$0.269^{1,2}$	$0.072^{1,2}$
	$_4$PRM-k-Means	$0.230^{1,2,3}$	$0.287^{1,2,3}$	$0.060^{1,2}$
	$_5$LCRM*	$\mathbf{0.323}^{1,2,3,4,6,7}$	$\mathbf{0.371}^{1,2,3,4,6,7}$	$0.071^{1,2,4,6,7}$
	$_6$LCRM*^{r}-k-NN	$0.261^{1,2,3,4}$	$0.322^{1,2,3,4,7}$	$0.066^{1,2,3}$
	$_7$LCRM*^{r}-k-Means	$0.264^{1,2,3,4,6}$	$0.319^{1,2,3,4}$	$0.064^{1,2,3,4}$

Note: The superscript indicates that the corresponding algorithm's accuracy is significantly lower ($p < 0.05$).

Table 7
Comparison of algorithms w.r.t. *Hit Ratio* and *MRR* with varied preference sizes in laptop dataset (the maximal preference size is 8).

Preference size	Method	Evaluation metrics		
		H@10	H@20	MRR
4 Features	$_1$Non-Review	0.033	0.033	0.021
	$_2$Review-Rank	0.032	0.032	0.031^{1}
	$_3$PRM-k-NN	$0.176^{1,2}$	$0.231^{1,2}$	$0.054^{1,2}$
	$_4$PRM-k-Means	$0.176^{1,2}$	$0.223^{1,2}$	$0.056^{1,2}$
	$_5$ LCRM*	$\mathbf{0.214}^{1,2,3,4,6,7}$	$\mathbf{0.298}^{1,2,3,4,6,7}$	$\mathbf{0.062}^{1,2,3,4,6,7}$
	$_6$LCRM*^{r}-k-NN	$0.201^{1,2,3,4}$	$0.246^{1,2,3,4}$	$0.058^{1,2,3,4}$
	$_7$LCRM*^{r}-k-Means	$0.201^{1,2,3,4}$	$0.273^{1,2,3,4,6}$	$0.061^{1,2,3,4}$
6 Features	$_1$Non-Review	0.053	0.042	0.025
	$_2$Review-Rank	0.043	0.043	0.032^{1}
	$_3$PRM-k-NN	$0.195^{1,2}$	$0.231^{1,2}$	$0.061^{1,2,4}$
	$_4$PRM-k-Means	$0.183^{1,2}$	$0.237^{1,2}$	$0.057^{1,2}$
	$_5$LCRM*	$\mathbf{0.231}^{1,2,3,4,6,7}$	$\mathbf{0.321}^{1,2,3,4,6,7}$	$\mathbf{0.068}^{1,2,3,4,6,7}$
	$_6$LCRM*^{r}-k-NN	$0.225^{1,2,3,4}$	$0.239^{1,2}$	$0.060^{1,2,4}$
	$_7$LCRM*^{r}-k-Means	$0.230^{1,2,3,4}$	$0.288^{1,2,3,4,6}$	$0.066^{1,2,3,4,6}$
8 Features	$_1$Non-Review	0.042	0.042	0.028
	$_2$Review-Rank	0.044	0.044	0.030^{1}
	$_3$PRM-k-NN	$0.204^{1,2}$	$0.252^{1,2}$	$0.063^{1,2}$
	$_4$PRM-k-Means	$0.205^{1,2,3}$	$0.268^{1,2,3}$	$0.071^{1,2,3}$
	$_5$LCRM*	$\mathbf{0.239}^{1,2,3,4}$	$\mathbf{0.331}^{1,2,3,4,6,7}$	$\mathbf{0.078}^{1,2,3,4,6}$
	$_6$LCRM*^{r}-k-NN	$0.235^{1,2,3,4}$	$0.267^{1,2,3}$	$0.072^{1,2,3,4}$
	$_7$LCRM*^{r}-k-Means	$0.238^{1,2,3,4,6}$	$0.285^{1,2,3,4,6}$	$0.074^{1,2,3,4,6}$

Note: The superscript indicates that the corresponding algorithm's accuracy is significantly lower ($p < 0.05$).

(see Tables 6 and 7, and Fig. 5). For instance, in the camera dataset, LCRM* achieves 0.305 at *N*@10 when the preference size is 6, versus 0.260 by LCRM*^{r}-k-Means ($t = 3.665$, $p < 0.05$). When the buyer's preferences become more complete, say to preference size 8, the accuracy of LCRM*^{r}-k-Means method is slightly increased to 0.281, but still lower than the one of LCRM* (which is 0.360; $t = 2.386$, $p < 0.05$). More significance analysis results are listed in Tables A.1 and A.2 in the Appendix.

It hence suggests that the LCRM* clustering based approach, which lies on the division of the whole population of reviewers according to their membership probability, is shown more effective than the heuristic k-Means clustering based method.

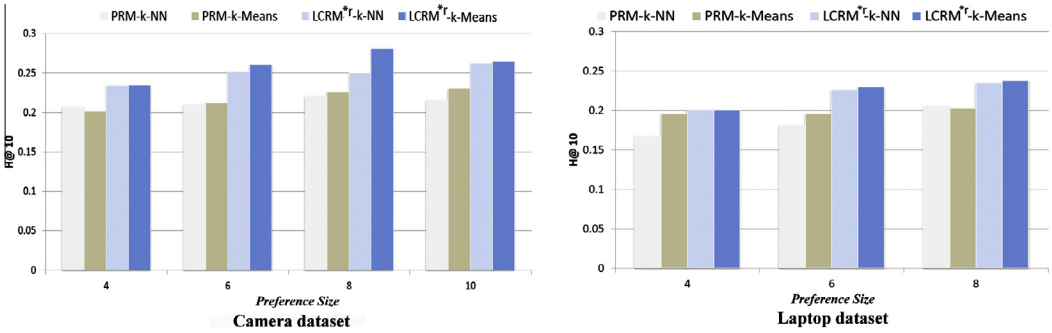

Fig. 4. Comparison among PRM-k-NN, PRM-k-Means, LCRM^{*r}-k-NN, and LCRM^{*r}-k-Means, w.r.t. *H@10*.

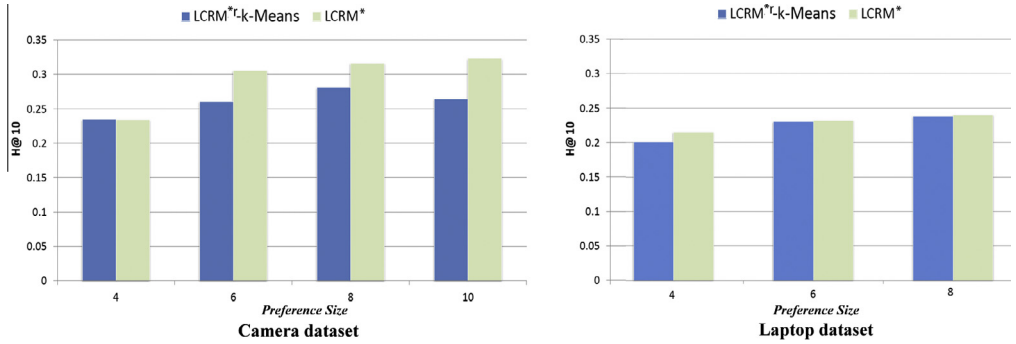

Fig. 5. Comparison between LCRM^{*r}-k-Means and LCRM*, w.r.t. *H@10*.

6.5. Discussion

The experimental results thus validate several hypotheses we brought forward at the beginning: (1) the proposed two branches of approaches, respectively based on Probabilistic Regression Model (PRM) and Latent Class Regression Model (LCRM), are both more effective than the two baselines Non-Review and Review-Rank; (2) as for deriving the reviewer-level preferences, LCRM* performs better than the PRM-based methods (e.g., LCRM^{*r}-k-NN vs. PRM-k-NN); and (3) as for the clustering of reviewers, first of all, the methods that involve the clustering process are better than the ones without (e.g., LCRM* and LCRM^{*r}-k-Means vs. LCRM^{*r}-k-NN); secondly, LCRM* acts more positive in terms of clustering reviewers and generating the cluster-level preferences, as per LCRM* vs. LCRM^{*r}-k-Means. As a summary, we can infer the following advantageous relation among these compared methods:

LCRM* > LCRM^{*r}-k-Means > LCRM^{*r}-k-NN > PRM-k-Means > PRM-k-NN > Review-Rank > Non-Review.

Therefore, it implies that LCRM* well addresses the data sparsity issue in the high-risk product domains, as it considers the preference homogeneity among reviewers when clustering them and further leverages the clustering outcomes into refining individual reviewers' preferences. In comparison, PRM purely relies on each reviewer's self-provided information to derive her/his preferences, which is unavoidably subject to be biased and likely result in overfitting phenomenon in the situation with sparse reviews. The comparison results regarding the two variants of LCRM*, i.e., LCRM^{*r}-k-NN and LCRM^{*r}-k-Means, highlight the particular value of LCRM* in unifying the ideal solutions to derive both *reviewer-level* and *cluster-level* preferences within the same framework. Specifically, the cluster-level preferences can be exploited to identify the truly like-minded reviewers and accelerate the filtering process, while the reviewer-level preferences can be utilized to adjust each reviewer's contribution when calculating a product's prediction score. More notably, our main idea of constructing *reviewers' multi-feature preferences from reviews* is empirically demonstrated. They can help identify inherently more relevant reviewers to the current buyer and hence more accurately retrieve the buyer's target choice. The method's practical usage in real e-commerce sites for saving the system's preference elicitation effort is thus suggested.

In the future, we will be interested in further improving the LCRM* algorithm from various aspects. Actually, though the algorithm's accuracy was proved in two datasets which are in different scales of reviews (6024 in the laptop dataset vs. 18,251 in the camera dataset), it would be still constructive to conduct a sensitivity analysis for testing the results' stability in the change of the quantity of reviews. Theoretically, as the LCRM* method is clustering based, a certain amount of sample observations (i.e., reviews in our case) should be ideally required. This is also related to the issue of identifying the optimal K (the number of clusters) and k (the number of neighbors in k-NN based methods). Normally, if with fewer reviews, the neighborhood size k would be higher for addressing the data sparsity problem. The number of clusters should be accordingly adjusted for the purpose of maximizing the preference homogeneity within a cluster and the preference heterogeneity across clusters. Thus, we will perform more experiments with the goal of determining the threshold of reviews above which our method will keep the advantageous performance and the right choices of parameters in different data conditions. Another consideration is about the clustering process. In our current algorithm, the clusters are disjoint, meaning that each reviewer belongs to only one cluster. This assumption might be relaxed by allowing non-disjoint clusters, so each reviewer is assigned to multiple clusters. On the other hand, the current buyer might also be matched to more than one cluster of reviewers (i.e., fuzzy matching), instead of current hard matching strategy. We will develop these possible extensions for assessing their actual effect on increasing the algorithm's performance.

7. Conclusion

In conclusion, this article in depth studied how to leverage the product reviews into improving recommendation for the active buyer in high-risk product domains. Given that the reviews posted by each reviewer are rather sparse in reality, few works have actually explored their usage in the development of decision support, but put more focus on eliciting the buyer's preferences on site since s/he is likely new. However, the previous user studies showed that even though it is feasible to elicit the buyer's feature preferences, such preferences are unlikely complete and accurate [6,38]. Therefore, in the current work, we have proposed to learn *reviewers*' preferences and employ such information to predict the current buyer's true preferences. To address the review sparsity phenomenon, we have emphasized deriving reviewers' *weighted feature preferences* from both textual reviews and overall ratings that they provided. More specifically, we have investigated two regression models. The first is the Probabilistic Regression Model (PRM) based on which we incorporated the opinion values associated with various features into inferring the weight a reviewer placed on each feature. Such preferences were used to perform the k-nearest neighborhood and k-Means recommendation algorithms (i.e., PRM-k-NN and PRM-k-Means). We have further extended Latent Class Regression Model (LCRM) with the aim to derive both cluster-level preferences and reviewer-level preferences simultaneously (so called LCRM*). Concretely, the clustering was performed based on the whole population's structure and the membership probability, which was then leveraged to refine individual reviewers' preferences so that the inherent preference similarity between reviewers can be taken into account. We have additionally implemented two variants of LCRM*: LCRM*^{r}-k-NN and LCRM*^{r}-k-Means, for the comparison in the experiment. In total, seven methods were tested, including two baselines Non-Review and Review-Rank. The experiment demonstrates the outperforming accuracy of LCRM* from several aspects: (1) deriving more stable reviewer-level preferences; (2) performing more effective clustering of reviewers; and (3) generating more accurate recommendation even when the buyer's stated preferences were less complete.

Our research hence highlights the value of deriving reviewers' multi-feature preferences from product reviews, as well as the impact of LCRM* on developing the *preference-based*

Table A.1
Significance analysis results via Student *t*-Test (assuming equal variances) in camera dataset.

		4 Features			6 Features			8 Features			10 Features		
		t(H@10)	*t*(H@20)	*t*(MRR)	*t*(H@10)	*t*(H@20)	*t*(MRR)	*t*(H@10)	*t*(H@20)	*t*(MRR)	*t*(H@10)	*t*(H@20)	*t*(MRR)
LCRM* vs.	LCRM*^{r}-k-Means	0.12	−1.28	−1.36	3.66**	4.10**	−0.30	2.38*	11.18***	2.25*	68,247***	185,061***	2.38*
	LCRM*^{r}-k-NN	0.21	−1.79	−1.11	4.93**	3.85**	2.75*	15.76***	13.93***	4.06**	3064.88***	35987.8***	5.48***
	PRM-k-Means	3.32*	1.80	0.14	9.08***	7.53***	3.29*	29.25***	19.70***	1.98	460.02***	42.81***	4.26**
	PRM-k-NN	2.83*	1.58	6.95***	9.66***	8.26***	5.04***	17.84***	17.90***	7.69***	5381.28***	85,010***	1.49
	Review-Rank	12.70***	15.04***	8.88***	19.11***	22.75***	22.03***	49.74***	40.10***	13.50***	9.80***	65.88***	29.74***
	Non-Review	12.79***	15.22***	18.31***	18.37***	22.05***	45.72***	65.15***	45.68***	34.29***	56.02***	47.43***	48.27***
LCRM*^{r}-k-Means vs.	LCRM*^{r}-k-NN	−0.11	−0.64	0.73	1.23	−0.64	3.77**	2.98*	−1.98*	2.20	2.18*	2.17*	1.05
	PRM-k-Means	5.96***	5.79***	45.13***	5.80***	2.59*	5.81***	5.05***	1.87*	6.60***	15.88***	17.71***	6.72***
	PRM-k-NN	5.34***	5.46***	51.5***	6.48***	3.61**	5.99***	5.16***	4.55**	6.60***	52.87***	90.66****	7.06***
	Review-Rank	26.79***	35.92***	38.18***	18.96***	19.52***	27.20***	14.49***	23.56***	12.93***	7.26***	51.14***	17.45***
	Non-Review	23.27***	30.64***	106.70***	17.76***	18.76***	63.69***	14.34***	25.51***	38.43***	46.40***	37.59***	28.24***
LCRM*^{r}-k-NN vs.	PRM-k-Means	6.20***	8.26***	2.39*	4.42**	4.03**	0.73	5.65***	5.94***	−2.66*	9.22***	11.45***	1
	PRM-k-NN	5.60***	7.94***	27.20***	5.04***	5.09***	3.02*	4.64**	7.95***	4.98**	16.37***	17.95***	−2.38*
	Review-Rank	27.61***	64.78***	27.60***	18.11***	24.83***	23.55***	28.41***	31.42***	11.16***	7.19***	51.14***	24.56***
	Non-Review	23.77***	41.64***	62.79***	16.86***	23.55***	58.68***	33.51***	35.96***	34.46***	47.72***	37.34***	42.45***
PRM-k-Means vs.	PRM-k-NN	−1.32	−0.47	1.19	0.60	2.04*	2.59*	1.14	3.75**	1.03	5.35***	6.50***	−2.58*
	Review-Rank	22.71***	34.18***	16.94***	28.04***	40.96***	22.09***	32.55***	27.68***	13.40***	5.80***	35.75***	14.68***
	Non-Review	19.22***	27.51***	37.45***	21.79***	34.74***	54.37***	52.55***	31.67***	40.41***	37.88***	28.44***	24.80***
PRM-k-NN vs.	Review-Rank	28.22***	36.37***	8.17***	50.14***	28.75***	14.49***	18.47***	20.43***	5.00**	5.00**	36.53***	20.64***
	Non-Review	22.24***	28.66***	68.63***	28.19***	25.84***	33.64***	19.83***	22.32***	18.42***	27.47***	27.34***	33.02***
Review-Rank vs.	Non-Review	11.62***	2.98	33.54***	0.26	0.55	24.86***	1.99	0.13	12.26***	0.41	−1.93	14.48***

Note: *t*-value is shown in the cell.
* ($p < 0.05$).
** ($p < 0.01$).
*** ($p < 0.001$).

Table A.2
Significance analysis results via Student *t*-Test (assuming equal variances) in laptop dataset.

		4 Features			6 Features			8 Features		
		t(H@10)	*t*(H@20)	*t*(MRR)	*t*(H@10)	*t*(H@20)	*t*(MRR)	*t*(H@10)	*t*(H@20)	*t*(MRR)
LCRM* vs.	LCRM*^{r}-k-Means	4.62**	7.43***	−0.29	0.79	8.80***	3.53**	0.19	14.77***	1.24
	LCRM*^{r}-k-NN	1.30	15.01***	5.80***	3.64**	17.60***	9.15***	1.13	24.99***	3.19*
	PRM-k-Means	13.32***	14.57***	8.51***	8.77***	12.55***	25.72***	10.62***	22.15***	3.84**
	PRM-k-NN	13.37***	12.34***	11.23***	8.66***	14.40***	8.41***	11.19***	27.93***	9.58***
	Review-Rank	65.00***	75.74***	16.06***	59.87***	68.84***	74.30***	90.03***	129.48***	32.58***
	Non-Review	61.56***	79.31***	63.97***	55.81***	67.68***	72.07***	72.48***	82.49***	33.89***
LCRM*^{r}-k-Means vs.	LCRM*^{r}-k-NN	0.71	26.37***	8.68***	3.98**	14.21***	7.78***	1.80	11.47***	5.71***
	PRM-k-Means	16.63***	12.61***	12.76***	9.11***	8.59***	31.95***	19.65***	11.35***	7.31***
	PRM-k-NN	16.25***	9.77***	14.81****	9.26***	10.71***	6.96***	20.45***	22.27***	19.05***
	Review-Rank	121.90***	164.94***	16.57***	66.62***	149.44***	90.43***	103.61***	137.31***	56.22***
	Non-Review	101.69***	226.34***	101.46***	61.69***	134.75***	95.40***	79.29***	75.65***	76.90***
LCRM*^{r}-k-NN vs.	PRM-k-Means	15.78***	5.73***	5.06***	8.06***	0.62	4.01**	62.40***	−2.24*	2.13
	PRM-k-NN	15.44***	3.46**	9.15***	7.97***	1.72	0.78	65.20***	30.03***	15.89***
	Review-Rank	79.47***	116.68***	14.77***	70.79***	78.28***	35.76***	122.68***	123.18***	55.05***
	Non-Review	72.48***	140.22***	20.30***	64.85***	74.86***	43.44***	85.96***	69.08***	77.18***
PRM-k-Means vs.	PRM-k-NN	0.53	−1.32	−1.20	−2.03*	0.79	− 12.11***	4.81**	5.78***	32.45***
	Review-Rank	79.47***	116.68***	13.95***	70.79***	78.28***	74.23***	122.68***	123.18***	53.05***
	Non-Review	72.48***	140.22***	122.25***	64.85***	74.86***	80.89***	85.96***	69.08***	73.82***
PRM-k-NN vs.	Review-Rank	15.89***	26.81***	33.96***	23.01***	43.22***	15.12***	59.12***	43.21***	14.13***
	Non-Review	68.12***	102.45***	46.10***	88.75***	25.16***	98.13***	31.62***	42.87***	35.77***
Review-Rank vs.	Non-Review	−0.87	0.70	6.66***	−2.18	0.33	15.38***	−0.00	0.73	5.30***

Note: *t*-value is shown in the cell.
* ($p < 0.05$).
** ($p < 0.01$).
*** ($p < 0.001$).

review-incorporated recommender algorithm. As the practical implication, this developed recommending component can be usefully plugged into an online system to be adopted in real e-commerce sites. More concretely, in such system, our previously proposed critiquing agent can be responsible for eliciting the current buyer's feature preferences [7], while the LCRM* algorithm can take charge of producing recommendation via incorporating product reviews. We will be engaged in empirically testing such system in different product domains by conducting user evaluations.

Acknowledgments

This research work was supported by Hong Kong Research Grants Council under Project ECS/HKBU211912.

Appendix A

See Tables A.1 and A.2.

Appendix B. Supplementary material

Supplementary data associated with this article can be found, in the online version, at http://dx.doi.org/10.1016/j.knosys.2013.05.006.

References

[1] S. Aciar, D. Zhang, S. Simoff, J. Debenham, Informed recommender: basing recommendations on consumer product reviews, IEEE Intelligent Systems 22 (2007) 39–47.
[2] T. Ahmad, M.N. Doja, Ranking system for opinion mining of features from review documents, Journal of Computer Science 9 (2012).
[3] D. Bridge, M.H. Göker, L. McGinty, B. Smyth, Case-based recommender systems, Knowledge Engineering Review 20 (2005) 315–320.
[4] R.D. Burke, K.J. Hammond, B.C. Young, The findme approach to assisted browsing, IEEE Expert: Intelligent Systems and their Applications 12 (1997) 32–40.
[5] L. Chen, P. Pu, Survey of preference elicitation methods, in: Technical Report No. IC/200467, Lausanne, Switzerland, 2004.
[6] L. Chen, P. Pu, Evaluating critiquing-based recommender agents, Proceedings of the 21st National Conference on Artificial Intelligence – AAAI'06, vol. 1, AAAI Press, 2006, pp. 157–162.
[7] L. Chen, P. Pu, Interaction design guidelines on critiquing-based recommender systems, User Modeling and User-Adapted Interaction 19 (2009) 167–206.
[8] L. Chen, P. Pu, Critiquing-based recommenders: survey and emerging trends, User Modeling and User-Adapted Interaction 22 (2012) 125–150.
[9] L. Chen, L. Qi, F. Wang, Comparison of feature-level learning methods for mining online consumer reviews, Expert Systems with Applications: An International Journal 39 (2012) 9588–9601.
[10] A.P. Dempster, N.M. Laird, D.B. Rubin, Maximum likelihood from incomplete data via the EM algorithm, Journal of the Royal Statistical Society 39 (1977) 1–38.
[11] M. Deshpande, G. Karypis, Item-based top-n recommendation algorithms, ACM Transactions on Information Systems 22 (2004) 143–177.
[12] A. Esuli, F. Sebastiani, Sentiwordnet: a publicly available lexical resource for opinion mining, in: Proceedings of the 5th Conference on Language Resources and Evaluation, LREC'06, 2006, pp. 417–422.
[13] A. Felfernig, G. Friedrich, D. Jannach, M. Zanker, An integrated environment for the development of knowledge-based recommender applications, International Journal of Electronic Commerce 11 (2006) 11–34.
[14] C. Fellbaum, WordNet: An Electronic Lexical Database, MIT Press, Cambridge, MA, 1998.
[15] M. Ganapathibhotla, B. Liu, Mining opinions in comparative sentences, in: Proceedings of the 22nd International Conference on Computational Linguistics, COLING'08, Association for Computational Linguistics, vol. 1, Stroudsburg, PA, USA, 2008, pp. 241–248.
[16] S. Garcia Esparza, M.P. O'Mahony, B. Smyth, Mining the real-time web: a novel approach to product recommendation, Knowledge-Based Systems 29 (2012) 3–11.
[17] A. Gunawardana, G. Shani, A survey of accuracy evaluation metrics of recommendation tasks, Journal of Machine Learning Research 10 (2009) 2935–2962.
[18] N. Hariri, B. Mobasher, R. Burke, Y. Zheng, Context-aware recommendation based on review mining, in: Proceedings of the 9th Workshop on Intelligent Techniques for Web Personalization and Recommender Systems, ITWP'11, Barcelona, Spain, 2011.
[19] J. Herlocker, J.A. Konstan, J. Riedl, An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms, Information Retrieval 5 (2002) 287–310.
[20] M. Hu, B. Liu, Mining and summarizing customer reviews, in: Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'04, ACM, New York, NY, USA, 2004, pp. 168–177.

[21] S.L. Huang, Designing utility-based recommender systems for e-commerce: evaluation of preference-elicitation methods, Electronic Commerce Research and Applications 10 (2011) 398–407.
[22] N. Jakob, S.H. Weber, M.C. Müller, I. Gurevych, Beyond the stars: exploiting free-text user reviews to improve the accuracy of movie recommendations, in: Proceedings of the 1st International CIKM Workshop on Topic-Sentiment Analysis for Mass Opinion, TSA'09, ACM, New York, NY, USA, 2009, pp. 57–64.
[23] N. Jindal, B. Liu, Opinion spam and analysis, in: Proceedings of the International Conference on Web Search and Web Data Mining, WSDM'08, ACM, New York, NY, USA, 2008, pp. 219–230.
[24] T. Kakkonen, Robustness evaluation of two CCG, a PCFG and a link grammar parsers, 2008, arXiv:0801.3817.
[25] R. Keeney, H. Raiffa, Decisions with Multiple Objectives: Preferences and Value Tradeoffs, Wiley, New York, 1976.
[26] T.H. Kim, S.B. Yang, An effective recommendation algorithm for clustering-based recommender systems, in: Proceedings of the 18th Australian Joint Conference on Advances in Artificial Intelligence, AI'05, Springer-Verlag, Berlin, Heidelberg, 2005, pp. 1150–1153.
[27] Y. Koren, R. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, Computer 42 (2009) 30–37.
[28] T. Lappas, G. Valkanas, D. Gunopulos, Efficient and domain-invariant competitor mining, in: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'12, ACM, New York, NY, USA, 2012, pp. 408–416.
[29] C.W.K. Leung, S.C.F. Chan, F. Chung, Integrating collaborative filtering and sentiment analysis: a rating inference approach, in: Proceedings of the ECAI Workshop on Recommender Systems, 2006, pp. 62–66.
[30] A. Levi, O. Mokryn, C. Diot, N. Taft, Finding a needle in a haystack of reviews: cold start context-based hotel recommender system, in: Proceedings of the 6th ACM Conference on Recommender Systems, RecSys'12, New York, NY, USA, 2012, pp. 115–122.
[31] Y. Li, J. Nie, Y. Zhang, B. Wang, B. Yan, F. Weng, Contextual recommendation based on text mining, in: Proceedings of the 23rd International Conference on Computational Linguistics: Posters, COLING'10, Stroudsburg, PA, USA, 2010, pp. 692–700.
[32] G.J. McLachlan, D. Peel, Finite Mixture Models, John Wiley and Sons, New York, 2000.
[33] B. Pang, L. Lee, S. Vaithyanathan, Thumbs up?: sentiment classification using machine learning techniques, in: Proceedings of the ACL-02 conference on Empirical Methods in Natural Language Processing, EMNLP'02, Association for Computational Linguistics, vol. 10, Stroudsburg, PA, USA, 2002, pp. 79–86.
[34] J. Payne, J. Bettman, E. Johnson, The Adaptive Decision Maker, Cambridge University Press, 1993.
[35] M.J. Pazzani, D. Billsus, Content-based recommendation systems, in: The Adaptive Web: Methods and Strategies of Web Personalization, Lecture Notes in Computer Science, vol. 4321, Springer, Berlin, Heidelberg, 2007, pp. 325–341.
[36] D. Poirier, F. Fessant, I. Tellier, Reducing the cold-start problem in content recommendation through opinion classification, Proceedings of the International Conference on Web Intelligence and Intelligent Agent Technology – WI-IAT'10, vol. 01, IEEE Computer Society, Washington, DC, USA, 2010, pp. 204–207.
[37] A.M. Popescu, O. Etzioni, Extracting product features and opinions from reviews, in: Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing, HLT'05, Association for Computational Linguistics, Stroudsburg, PA, USA, 2005, pp. 339–346.
[38] P. Pu, L. Chen, Integrating tradeoff support in product search tools for e-commerce sites, in: Proceedings of the 6th ACM Conference on Electronic Commerce, EC'05, ACM, New York, NY, USA, 2005, pp. 269–278.
[39] J. Reilly, K. McCarthy, L. McGinty, B. Smyth, Dynamic critiquing, in: Proceedings of 7th European Conference on Advances in Case-Based Reasoning, Lecture Notes in Computer Science, vol. 3155, Springer, 2004, pp. 763–777.
[40] J. Reilly, K. McCarthy, L. McGinty, B. Smyth, Incremental critiquing, Knowledge-Based Systems 18 (2005) 143–151.
[41] T.L. Saaty, A scaling method for priorities in hierarchical structures, Journal of Mathematical Psychology 15 (1977) 234–281.
[42] B. Sarwar, G. Karypis, J. Konstan, J. Riedl, Recommender systems for large-scale e-commerce: scalable neighborhood formation using clustering, in: Proceedings of the 5th International Conference on Computer and Information Technology, vol. 1, 2002.
[43] G. Shakhnarovich, T. Darrell, P. Indyk, Nearest-Neighbor Methods in Learning and Vision: Theory and Practice (Neural Information Processing), MIT Press, 2006.
[44] H. Shimazu, Expertclerk: a conversational case-based reasoning tool for developing salesclerk agents in e-commerce webshops, Artificial Intelligence Review 18 (2002) 223–244.
[45] M. Stolze, M. Ströbel, Dealing with learning in ecommerce product navigation and decision support: the teaching salesman problem, in: Proceedings of the 2nd Interdisciplinary World Congress on Mass Customization and Personalization, Munich, German, 2003.
[46] M. Terzi, M.A. Ferrario, J. Whittle, Free text in user reviews: their role in recommender, in: Proceeding of the 3rd ACM RecSys Workshop on Recommender Systems and the Social Web, Chicago, Illinois, USA, 2010, pp. 45–48.
[47] C.A. Thompson, M.H. Göker, P. Langley, A personalized system for conversational recommendations, Journal of Artificial Intelligence Research 21 (2004) 393–428.
[48] P. Viappiani, B. Faltings, P. Pu, Preference-based search using example-critiquing with suggestions, Journal of Artificial Intelligence Research 27 (2006) 465–503.
[49] F. Wang, L. Chen, Recommendation based on mining product reviews' preference similarity network, in: 6th Workshop on Social Network Mining and Analysis, 2012 ACM SIGKDD Conference on Knowledge Discovery and Data Mining, SNA-KDD'12.
[50] F. Wang, L. Chen, Recommending inexperienced products via learning from consumer reviews, in: Proceedings of the 2012 IEEE/WIC/ACM International Conferences on Web Intelligence, WI'12, IEEE Computer Society, 2012, pp. 596–603.
[51] S. Wang, D. Li, L. Zhao, J. Zhang, Sample cutting method for imbalanced text sentiment classification based on brc, Knowledge-Based Systems 37 (2013) 451–461.
[52] M. Wedel, W.A. Kamakura, Market Segmentation – Conceptual and Methodological Foundations, vol. 9, Springer, 2000.
[53] S. Xie, G. Wang, S. Lin, P.S. Yu, Review spam detection via temporal pattern discovery, in: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'12, ACM, New York, NY, USA, 2012, pp. 823–831.
[54] J. Yu, Z.J. Zha, M. Wang, T.S. Chua, Aspect ranking: identifying important product aspects from online consumer reviews, in: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, HLT'11, Association for Computational Linguistics, vol. 1, Stroudsburg, PA, USA, 2011, pp. 1496–1505.
[55] K. Zhang, R. Narayanan, A. Choudhary, Voice of the customers: mining online customer reviews for product feature-based ranking, in: Proceedings of the 3rd Conference on Online Social Networks, WOSN'10, CA, USA, 2010, pp. 11–11.
[56] W. Zhang, G. Ding, L. Chen, C. Li, C. Zhang, Generating virtual ratings from chinese reviews to fuse into collaborating filtering algorithms, ACM Transactions on Intelligent Systems and Technology 4 (2013).