Analysis of Economic Impact of Online Reviews: An Approach for Market-Driven Requirements Evolution

Wei Jiang[1], Haibin Ruan[2], and Li Zhang[1]

[1] School of Computer Science and Engineering, Beihang University, Beijing, China
jiangwei@cse.buaa.edu.cn, lily@buaa.edu.cn
[2] Investment Department, Central University of Finance and Economics, Beijing, China
nivadacufe@gmail.com

Abstract. As a novel market data, online reviews can manifest user demands in real contexts of use. Thereby, this paper proposes a demand-centered approach for requirements evolution by mining and analyzing online reviews. In our approach, it is challenging to improve the accuracy of opinion mining techniques for huge volume of noisy review data. Furthermore, how to quantitatively evaluate the economic impact of user opinions for determining candidate requirements changes is also a challenging problem. In this paper, an opinion mining method augmented with noise pruning techniques is presented to automatically extract user opinions. After automatic synthesizing the information extracted, a utility-oriented econometric model is employed to find causal influences between the system aspects frequently mentioned in user opinions and common user demands for revising current requirements. A case study shows that the presented method of opinion mining achieves good precision and recall even if there is a large amount of noisy review data. The case study also validates the effectiveness of our approach that it discovers the candidate requirements changes related to the software revenue, especially the ones that are ignored by software developers.

Keywords: Electronic market, requirements evolution, online reviews, opinion mining, econometric analysis.

1 Introduction

Evolution is an inherent attribute of software requirements due to changing user needs and application environment [1]. In today's competitive market, it is crucial for the software system to respond to the social environment where users form opinions based on their experience with it [2]. With user generated content becoming mainstream in Web platforms, consumers are willing to publish online reviews to express their opinions about software systems. As market data, these reviews manifest user demands in real contexts of use, which have become a very important resource for eliciting requirements for designing future systems. Therefore, our goal is to explore a demand-centered approach for requirements evolution through mining and analyzing online reviews. We first automatically extract software features and relevant user opinions from online reviews. Second, we determine candidate requirements changes by econometric analysis of user opinions.

Currently, some automated techniques, such as text mining, information retrieval, and machine learning are utilized in identifying the aspects of the software system and associated opinions mentioned in user comments [3, 4]. However, due to different content quality of reviews, it is challenging to ensure the accuracy of automated techniques for huge volume of review data. In this paper, we augment the existing opinion mining method with noise pruning techniques. Experiment results show that our method achieves good performance even if large amounts of noisy review data.

Furthermore, the system aspects frequently commented on in user opinions are useful evidence to design requirements for future systems. Several approaches have provided a set of techniques and processes about how to analyze and determine requirements in accordance with user feedback [5, 6]. However, content analysis in these approaches relies more on the manual effort. Moreover, the evaluation models of user opinions cannot consider market elements. How to quantitatively evaluate the economic impact of user opinions for determining candidate requirements changes is also a challenging problem. In our approach, we first adopt the text clustering technique and heuristic method to group and rate user opinions. Then, we employ a utility-oriented econometric model to find causal influences between the system aspects frequently commented on in user opinions and common user demands for revising current requirements. A case study validates the effectiveness of our approach that it discovers the candidate requirements changes related to user demands, especially the ones that tend to be ignored by software developers.

The contributions of our research are as follows: (1) our opinion mining method can accurately deal with large amounts of noisy reviews; and (2) our approach can support analysts to make wise decision on requirements evolution in the open market by suggesting the requirements changes that are more economically valuable.

The rest of this paper is organized as follows. Section 2 describes the method for opinion mining. Section 3 elaborates econometric analysis for requirements evolution according to the exracted user opinions. To evaluate our approach, Section 4 presents a case study in which the candidate requirements changes for the future system are derived from the review data of Kaspersky Internet Security 2011. Finally, Section 5 and Section 6 discuss related work, conclusions and future work.

2 User Opinion Mining

The user opinion in a review of the software system is defined as a pair of *feature* and *opinion*. The feature refers to a software feature that is a prominent or distinctive user-visible aspect, quality, or characteristic of the system [10]. The opinion is a subjective user evaluation expressed on the feature. Opinions and features are often related syntactically and their relations can be modeled using the dependency grammar [4]. Therefore, we adopt the syntactic relation-based propagation approach (SRPA) [7] for fine-gained opinion mining. However, once there is a large amount of noisy review data, SRPA may introduce much noise and result in the decrease in the accuracy. We augment SRPA with pruning noisy opinion words and features. Finally, we recover complete expressions of user opinions according to the extracted individual words.

2.1 Extracting User Opinions Using SRPA

SRPA is a bootstrapping approach, which uses the syntactic relations that link opinion words and features to expand the initial opinion lexicon and extract features. A rule-based strategy is used to iteratively perform the extraction task through propagation. The extration rules modeling noun features and adjective opinion words are defined according to the syntactic dependency relations in which people often express their opinions. Through the identification of extration rules, the propagation algorithm is first bootstrapped by the seed opinion lexicon to extract opinion words and features, then applies newly extracted words to further extraction, and finally stops until no more new words are extracted. In this way, SRPA has good results of precision and recall even if the initial opinion lexicon is small.

For opinion extraction, we first redefine the exraction rules based on Stanford syntactic parser[1]. Second, we take raw review data of the software system and a seed lexicon as the input of SRPA and then obtain the set of extracted features and the expanded opinion lexicon. Third, we identify the reveiw sentences matching the extraction rules and extract <*feature*, *opinion*> pairs whose features and opinion words are respectively involved in the extracted feature set and expanded opinion lexicon. Finally, we adopt the noise pruning method in SRPA to remove the user opinions for other competitive systems.

2.2 Pruning Noisy Opinion Words

Although some adjectives modify features, they do not have any positive, negative or neutral sentiment polarity. For example, in the sentence "*The <u>latest</u> <u>version</u> is <u>good</u>*", *good* is a positive opinion word whereas *latest* is an ordinary adjective. However, *latest* is extracted incorrectly. This means that SRPA may introduce noisy opinion words, as the extraction rules are unconstrained.

We determine whether the extracted adjectives are noise using the following heuristic rules, which are defined from observing contexts of the reviews. **Rule 1**: When two adjectives modify the same feature in a clause, the adjective before it as the adjectival modifier is an ordinary adjective if the adjective after it as the predicative adjectives is an opinion word. **Rule 2**: Two adjectives in a clause that are connected by a coordinating conjunction or have a coordinating relation are either opinion words or ordinary adjectives. **Rule 3**: When two adjectives modify the same feature in different sentences, the adjective after it as the predicative adjectives is also an opinion word if the adjective before it as the adjectival modifier is an opinion word.

1 `http://nlp.stanford.edu/software/lex-parser.shtml`

The extracted adjectives are initially recognized as opinion words if they are involved in an opinion word lexicon, such as MPQA[2] and Liu's opinion words[3]. The reasoning is performed according to the above three rules. Most adjectives may be identified as either ordinary adjectives or opinion words. However, some adjectives are still not determined on the context evidence. Such adjectives are regarded as opinion words for avoiding that excessive pruning decrease the recall of opinion mining.

2.3 Pruning Noisy Features

In expressions of user opinions, features refer to domain-specific nouns that appear more frequently in a certain domain and less in other domains. However, most opinion words are domain-independent adjectives that can modify any object. As a result, SRPA increases the risk of introducing ordinary nouns when using opinion words to extract features. For example, there are two sentences "*Kaspersky occupies* <u>*many resources*</u>" and "*I have used Kaspersky for* <u>*many years*</u>". The ordinary noun *years* is incorrectly extracted through the opinion word *many* modifying the feature *resources*.

Observing the way in which people express their opinions, we can see that features are frequently modified by opinion words. Based on such observation, we count the numbers of opinon words and ordinary adjectives modifying each extracted noun and employ the standard Support Vector Machine (SVM) algorithm [8] to identify correct features. The SVM inputs a set of pre-classified nouns with high frequency opinion words or ordinary adjectives, and then trains a binary linear classifier to predict the category of a new noun.

In the training phase, suppose $F=\{f_1, f_2, \ldots, f_n\}$ is the training feature set. For $\forall f_i \in F$, it is assigned a $(\mathbf{x}_i, y_i)$ where $\mathbf{x}_i$ is a 2-dimensional integer vector $\mathbf{x}_i=\{x_{i1}, x_{i2}\}$ corresponding to the nubmers of opinon words and ordinary adjectives of f_i and $y_i \in (1, -1)$ determines whether f_i is a feature. The SVM model trains a classifier as

$$L(\mathbf{x}_i)=\mathbf{w}\mathbf{x}_i+b \tag{1}$$

where $\mathbf{w}=\{w_1, w_2\}$ is the weight vector, b is a real offset. In accordance with maximal margin that wants to find decision boundary that is as far away from the data of both classes as possible, we can get the optimal vector. Furthermore, when a new noun comes, we can determine whether it is a feature through the trained SVM classifier.

2.4 Recovering Expressions of User Opinions

Base on the propagation algorithm, the extracted features and opinion are individual words. However, the feature can be a noun phrase and the opinion may be an adjective phrase including an opinion word and its adverb modifiers. For example, in the sentence "*The* <u>*user interface*</u> *is* <u>*not very intuitive*</u>", the feature *user interface* is a noun phrase and the opinion *not very intuitive* is a negative adjective phrase. We thereby

[2] `http://mpqa.cs.pitt.edu/lexicons/subj_lexicon/`

[3] `http://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html`

use shallow parsing techniques to recover complete expressions of user opinions. The Stanford parser can output the phrase structure tree of a sentence. In the tree, the noun/ adjective phrase that contains the extracted feature/opinion word is recognized as the feature/opinion phrase. Moreover, a negative word in the clause modifying the opinion word is also identified as a part of the opinion phrase even if it is not directly contained in the adjective phrase.

3 Econometric Opinion Analysis

High ratings of user satisfaction are widely believed to be the best indicator of the company's future profits [9]. In requirements analysis, we thereby emphasize the economic benefits that result from the system improvement with changing user demands. A utility-oriented econometric model is employed to find the candidate requirements changes for system improvement. The detailed steps are as follows: (1) organizing the similar extracted features into a system aspect relevant to an overall, functional or quality requirement, (2) rating the associated opinions with the extracted features, (3) analyzing the causal relationships between the utilities of system aspects and software sales and then determine the important aspects for revising current requirements, and finally (4) generating a meaningful report on the candidate requirements changes.

3.1 Categorizing Features

As what features may be mentioned in the reviews is unknown, the k-means clustering [20] is adopted to categorize the extracted features into system aspects that depict their semantic commonalities. The k-means clustering aims to divide n data points into k clusters so as to minimize the mean squared distance from each point to the center within a cluster. There are two potential problems using the k-means algorithm for categorizing features: how to compute the distance between features and how to choose k seeds for avoiding poor clusters.

For the first issue, the distance between features is defined as the difference between 1 and their semantic similarity. We rely on an explicit semantic analysis algorithm (ESA) to compute semantic similarity. The idea of ESA is to use machine learning techniques to represent the meaning of any text as a weighted vector of Wikipedia-based concepts and then assess the relatedness of texts in this space amounts to comparing the corresponding vectors using conventional metrics [10]. ESA can enhance the feature representations described by nouns/noun phrases and further improve the accuracy of clustering. For the second issue, we now choose a larger value (50-100) for k to cover as more feature categories as possible and then use k-means++ algorithm [21] for optimizing k seeds to cluster the extracted features.

Each feature category can represent an overall description or a user-perceived functional/quality aspect of the software system. As the value of k is larger, there may be duplicated feature categories. To solve the problem, we regard the feature that is nearest to the center within a cluster as the name of each feature category and manually merge those categories whose names are similar.

3.2 Rating User Opinions

We propose a two-stage strategy for rating the associated opinion with an extracted feature on a five-point scale. In the first stage, a heuristic method based on the context evidence [7, 11] is adopted for assigning polarities to opinion words. The heuristic rules are defined according to the observations about the consistent manner in which people often express their opinions. The same polarity is propagated between features and opinion words based on the extraction rules unless there are explicit *contrary words* or *negative words* in the clauses. There are new opinion words extracted by some features without polarities. Their polarities are inferred using the overall review polarity. In the second stage, the opinion word is rated based on its polarity and the ratings that users place on the reviews. The steps are indicated in Fig. 1.

```
Input: Opinion word $o$ and its polarity $p$,
       Review data set $R$, Ratings of review data $Rat$
Output: Rating of $o$ $Rat(o)$
1.  if $p$ is neutral
2.  then $Rat(o)=3$
3.  else if $p$ is positive
4.          then
5.                  compute the sum $S_4$ of $o$ mentioned in $R$ with its $Rat=4$
6.                  compute the sum $S_5$ of $o$ mentioned in $R$ with its $Rat=5$
7.                  $Rat(o)=S_5>S_4?5:4$
8.          else
9.                  compute the sum $S_1$ of $o$ mentioned in $R$ with its $Rat=1$
10.                 compute the sum $S_2$ of $o$ mentioned in $R$ with its $Rat=2$
11.                 $Rat(o)=S_2>S_1?2:1$
12.         endif
13. endif
```

Fig. 1. Rating opinion words

3.3 Econometric Model-Based Requirements Analysis

The importance of a system aspect represents its priority for requirements evolution. It is stated that its importance to revising the requirements for future releases lies on the proportional to the number of reviews relevant to the system aspect [4, 12]. From the market's perspective, however, the importance of a system aspect depends more on its economic benefits. If changes in the system aspect can improve the ability of the software system satisfying changing user demands, its improvement will result in the company's future profits. Thus, we present a utility-oriented econometric model to analyze which system aspects are required in the requirements for future releases.

In economics, utility is the ability that a good or service satisfies consumer wants. The rating of a user opinion represents the degree to which the system satisfies a certain user about the feature. Thus, the utility of a system aspect is measured by average ratings of user opinions related to the aspect. It is computed as follows:

$$U(a)=\frac{1}{n_a}\sum_{i=1}^{n_a}Rat(f_i,\quad o_i) \tag{2}$$

where n_a is the number of features belonging to aspect a and $Rat(f_i, o_i)$ is the rating of associated opinion o_i with feature f_i.

In marketing communication, online reviews are a novel word-of -mouth that has positive impact on the product sales [13]. Furthermore, the change in software sales is triggered by the change in user satisfaction of some system aspects frequently mentioned on in the reviews. In order to estimate the importance of a system aspect from the market's perspective, we need to find the causal influence between the utilities of system aspects in real contexts of use and software sales. We adopt the Granger causality model (GCM) to analyze whether the utility of a system aspect is useful in forecasting the software sales. GCM is a linear regression method often used in econometrics to quantify the causal influence from time series variables. It has better results than Bayesian network and information theory [14]. Its improvement of prediction may reduce the influence of coincidental causality.

Suppose there are two time series X and Y. X is said to Granger-cause Y if Y can be better predicted using the histories of both X and Y than the history of Y alone [15]. We test whether X causes Y or not by estimating the following regressions:

$$Y(t) = \alpha_0 + \sum_{i=1}^{n} \alpha_i Y(t-i) + \varepsilon_1 \tag{3}$$

$$Y(t) = \alpha_0 + \sum_{i=1}^{n} \alpha_i Y(t-i) + \sum_{i=1}^{n} \beta_i X(t-i) + \varepsilon_2 \tag{4}$$

where n is the maximal time yield, ε_i is a random distribution. If (4) is a significantly better than (3) through hypothesis testing, time series X causes time series Y.

Estimating the importance of a system aspect based on GCM follows a series of steps. First, the review set R of a software system S is divided into a set of sequential groups $G=\{g_1, g_2, \ldots, g_z\}$ according to the chosen time yield. The relevant sales $S(S)_{t=i}$ and utility of a system aspect $U(a)_{t=i}$ to group g_i are measured and then the time series $X=\{U(a)_t\}_{t\in Z}$ and $Y=\{S(S)_t\}_{t\in Z}$ ($Z=\{1, 2, \ldots, z\}$) are constructed based on their measurements from the group set G. Second, Equation (3) and (4) are used for examining the Granger causal relationships between time series X and Y without considering the correlations among system aspects. Finally, in light of the Granger causality test, the system aspects whose utilities have significant positive correlations with the software sales are discovered as the useful evidence to revise the requirements for future releases. The correlations of system aspects indicate their importance for market-driven requirements evolution.

3.4 Generating Report

The goal of generating the report is to provide a meaningful information for developers to improve the software system. The report contains system aspects and associated statements that describe candidate requirements changes for system improvement. The statements are used to interpret the meanings of candidate requirements changes. They can be mapped to the user opinions mentioned in the reviews through the corresponding system aspects. Although system aspects for designing furture systems are discovered in previous analysis, whether they are related to candidate requirements changes cannot be determined through the Granger causality test. We distinguish these system aspects in accordance with the statistical characteristics of their utilities. We first compute the mean *EU*(*a*) and standard deviation *SU*(*a*) of the utility of each system aspect and the overall user satisfaction *OS*(*R*) over review set *R*. *OS*(*R*) is measured by the average of ratings that users place on the reviews. Second, we determine the system aspects relevant to candidate requirements changes using the following steps depicted in Fig. 2.

Input: aspect *a*, mean *EU*(*a*) and standard deviation *SU*(*a*) of its utility overall user satisfaction *OS*(*R*) over review set *R*
Output: the category of candidate requirements changes relevant to aspect *a* *Req*(*a*)
Note: λ is the threshold

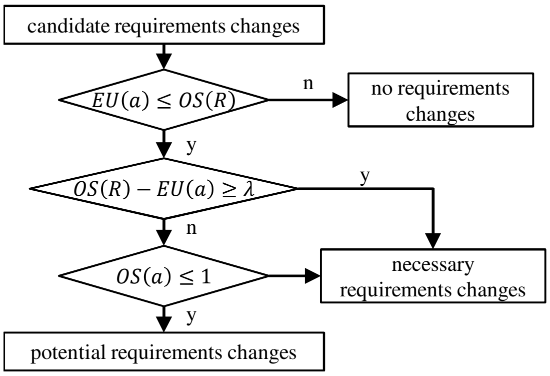

candidate requirements changes
$EU(a) \leq OS(R)$
n
no requirements changes
y
$OS(R) - EU(a) \geq \lambda$
y
n
necessary requirements changes
$OS(a) \leq 1$
y
potential requirements changes

Fig. 2. A strategy of determining candidate requirements changes

We therefore check the software requirements specification (SRS) to revise the corresponding statements to the system aspects relevant to candidate requirements changes. We conclude the new statement based on several negative review sentences revelant to such system aspect with the highest probability to alter the old one in SRS.

4 Case Study

A case study was carried out to elicit the requirements for system improvement using the reviews of Kaspersky Internet Security 2011 (KIS 2011). In the study, we first evaluate the validity of the opinion mining technique on dealing with large amounts of noisy review data and then evaluate the effectiveness of the generated report for system improvement.

4.1 Experiments for Mining User Opinions of KIS 2011

Experimental Design.
Correct user opinions derived from large amounts of noisy review data are the evidence to requirements elicitation for system improvement. Based on the review data of KIS 2011, we compare our method P-SRPA with SRPA and discuss the advantages and disadvantages according to all metrics including recall, precision and F-score.

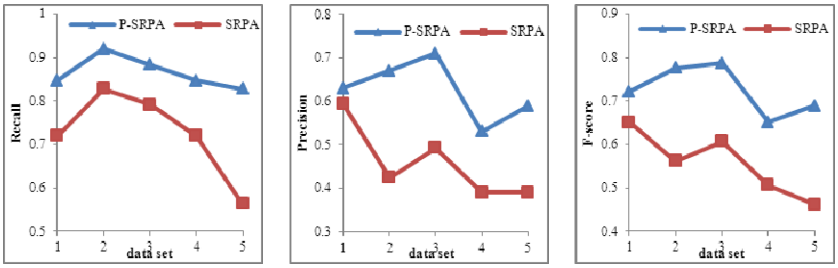
The raw review data of KIS 2011 used in this paper was scraped from Amazon.com. The data set contains 380 reviews and 3200 sentences from September 2, 2010, to February 25, 2012. Each review has a user rating on a five-point scale. All sentences of the review data are descended by the helpfulness of a review that is computed through our previous work [16]. Then these ordered sentences are divided into five subsets $D=\{d_1, .., d_5\}$. For each subset, the potential features and opinions in the sentences are manually labeled as the testing data for the comparison experiment.

We perform P-SRPA and SRPA using the seed opinion lexicon provided by Hu and Liu [11], which involves 654 positive and 1098 negative opinion words. Recall is computed as the number of true positives divided by the sum of the number of true positives and the number of false negatives. Precision is computed as the number of true positives divided by the sum of the number of true positives and the number of false positives. F-score is computed as follows:

$$\text{F-score} = 2\times\frac{\text{precision}\times\text{recall}}{\text{precision}+\text{recall}} \tag{5}$$

Comparison Results and Discussion.

The Fig. 3 shows the results of recall, precision and F-score of P-SRPA and SRPA using different subsets. From Fig. 3 (a), the average recall of P-SRPA is 0.86 and that of SRPA is 0.77. This shows that the propagation is reasonable in achieving high recall. Clearly, the recall may be affected by natural language processing (NLP) techniques. The sentences in high noisy data subsets often have many errors of spelling and grammatical structure so that automatic tagging and parsing don't work correctly. Fortunately, the recall values of P-SRPA do not decrease significantly with the increase of noisy data, only from 0.92 to 0.82.



(a) Recall (b) Precision (c) F-score

Fig. 3. Results of mining user opinions from KIS 2011 review data

Observing Fig. 3 (b), we can see P-SRPA outperforms SRPA in precision. In the low noisy data subsets, the precision values of P-SRPA and SRPA are more than 0.70. This indicates that dependency relation-based extraction rules are effective for identifying correct features and opinions. However, the precision values of SRPA decrease obviously with the increase of noisy data, from 0.60 to 0.40, while our precision values decrease from 0.71 to 0.53. In addition to NLP problems, the causes are as follows. First, there are many ordinary nouns and adjectives irrelevant to user opinions in the high noisy data subsets. Due to unconstrained dependency relations, the words match the extraction rules so as to be extracted incorrectly. Second, the expressions of user opinions are flexible and diverse in the reviews. The descriptions of the same feature do not usually have a unified term. Thus, noisy feature pruning in SRPA does not produce obvious effect, as it relies on the frequency of terms. Moreover, SRPA does not still provide a method for noisy opinion word pruning. To address these problems, P-SRPA employs a heuristic method to distinguish opinion words from ordinary adjectives and then filters incorrect features through the SVM classifier.

Fig. 3 (c) shows P-SRPA archives better average F-score than that of SRPA. We can draw the conclusion that P-SRPA is useful to provide user feedback for requirements evolution analysis even if dealing with large amounts of noisy review data.

4.2 Requirements Analysis for KIS 2011 Improvement

Evaluation of Generated Report.
We organize a human subject study to determine the usefulness of the generated report for system improvement. In the study, five participants are the developers that work in the computer security companys. All of them have more than three years of experience in software development. The participants are required to use KIS 2011 and read its SRS for developing a set of candidate requirements changes. As the SRS of KIS 2011 is not open to users, we create it according to understanding of the application and other information, such as product introduction, user manual and AV-Test results. The SRS contains 21 functional statements and 14 quality statements that are appreciable to users. We compare the generated report and participants' results to determine whether our approach can discover the candidate requirements changes that are ignored by developers.

Generating Candidate Requirements Changes of KIS 2011.
We carried out the following steps for generating the report on the candidate requirements changes of KIS 2011. We first utilized k-means++ algorithm with k=80 to classify the extracted features and then obtained 23 system aspects after merging duplicated feature categories. Table 1 shows 15 systems aspects that contain the features frequently mentioned in the reviews.

Second, we chose a week as the time yield and used the Granger causality test to analyze whether the time series of the utility of each system aspect shown in Table 1 were useful in forecasting the time series of the sales rank of KIS 2011. As the exact sales of KIS 2011 are not available, we modify the utility-oriented GCM in Section 3.3 by replacing the software sales with the software sales rank that is publicly available information in most Web platforms. As the prior research in marketing experimentally observed that the distribution of sales in terms of associated sales rank has a Power distribution [17], we select the principal system aspects whose utilities have the negative correlations with the sales rank as shown in Table 2. These system aspects are related to the requirements of KIS future releases. Note that the system aspects of *scan*, *safe run*, and *parental control* have not the causal influences on the sales rank, although they receive many user opinions in the reviews. Such system aspects do not have to be changed from the economic perspective, since the software revenue is not significantly enhanced by improving their utilities.

Table 1. Results of categorizing features

Perspective	Aspect	Feature
Overall		Software, product, application, package, program, suit, version, Internet security, Kaspersky, Kaspersky internet security, KIS
Function	Protection	Computer protection, Virus protection, Malware protection, Protection
	Antivirus	Antivirus, Anti-virus, Anti virus
	Firewall	Two-way firewall, Personal firewall, Firewall
	Scan	Scan, Computer scan, System scan, Virus scan, Malware scan, Scanning
	Safe run	Safe run, Safe desktop, Safe mode
	Parental control	Parental control
	Installation	Installation, Installation process, Installing
	Update	Update, Upgrade
Quality	Performance	Performance, Speed, Time, Slowdown
	Resource	Resource utilization, System resource, Resource, Memory, CPU, Footprint
	Reliability	Reliability, Bug, Crash, Reboot
	Usability	Usability, Easy to use, Use friendly
	User interface	User interface, Window, Setting, Look
	Flexibility	Flexibility, Configuration, Customization

Table 2. Causal relationships of KIS 2011

Function				
Protection	Antivirus	Firewall	Installation	Update
-0.738*	-0.625*	-0.226**	-0.558*	-0.387**

Quality					
Performance	Resource	Reliability	Usability	User interface	Flexibility
-0.815*	-0.764*	-0.317*	-0.832*	-0.629*	-0.315**

*p=0.05; **p=0.1

Table 3. Statistics of Utilities of System Aspects of KIS 2011

	Function		Quality				
	Installation	Update	Performance	Resource	Reliability	Usability	User inter-face
$EU(a)/SU(a)$	3.79/0.96	3.75/1.29	3.44/1.24	3.50/1.02	2.83/0.58	3.71/0.95	3.85/0.97

Finally, we generated the report on the candidate requirements changes through observing the overall user satisfaction $OS(R)$ of KIS 2011 (4.08) and the statistics ($EU(a)$ and $SU(a)$) of the utility of each system aspect. We used the strategy in section 4.4 (λ=1.36) to find the system aspects relevant to the candidate requirements changes of KIS 2011 that are shown in Table 3. The categories of potential/necessary requirements changes are distinguished through yellow/red colors. Through the above analysis, we checked the SRS of KIS 2011 and revised the associated statements with the system aspects in Table 3 to generate the report shown in Table 4.

Table 4. Candidate requirements changes of KIS 2011

ID	Statement	Aspect	Type
1	Automatic configuration during installation	Installation	#necessary
2	Update databases and application modules more smoothly	Update	#potential
3	Make less impact on the computer in daily use.	Performance	#necessary
4	Reduce the resource footprint when performing the user's task	resource	#necessary
5	Fix bugs and issues that are causing crashing and rebooting	Reliability	#necessary
6	Improve usability	Usability	#potential
7	Make user interface more intuitive and reduce needless pop-up messages	User interface	#potential

Analysis and Discussion of Requirements Changes.
Fig. 4 indicates the requirements changes designed by the participants, which contains 4 functional and 3 quality aspects of KIS 2011. Only quality aspects are consistent with those in our report whereas functional aspects are completely different.

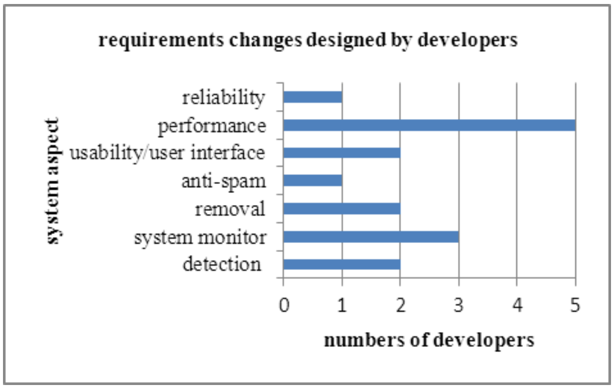


Fig. 4. Requirements changes designed by developers

We investigated the decision-making processes of participants. They stated that the changed functional requirements were designed based on the evaluation criteria for Internet security technology and personal experience. For functional requirements, developers paid more attention to key and special features while users were more concerned with the features closely related to user habits. As KIS 2011 is one of best sellers in the Internet security domain, its main functional aspects implicitly meet mass user desires in the open market. The improvement of such aspects cannot have significant impact on software revenue only when they have serious issues and bugs. In terms of KIS 2011, however, automatic remote *installation* and regular online *update* often fail in different contexts of use in the real world. The improvement of those aspects can promote the purchase of most users. Developers often ignore the requirements changes in these aspects as they may not be regarded as important modules of the software from the perspective of technology.

We showed the participants the report generated by our approach and told them that changes in the functional aspects of *installation* and *update* could bring about significant growth in the software revenue. Three participants admitted to lose sight of these aspects, and the other two participants tried to revise them. Thereby, our approach can discover the requirements changes that developers sometimes overlook.

From Fig. 4, we find that the participants have obvious disagreements in the quality aspects of *reliability* and *usability*. Few participants determined the changes in them. The other participants stated that it was hard to well implement *reliability* and *usability* because their user preferences were diverse. Considering huge development costs, the participants gave up their improvement. However, the report generated by our approach shows that changes in such quality aspects are necessary from the economic perspective. Therefore, it is a trade-off between economic factors and technical levels for requirement evolution. Our approach suggests the requirements changes that are more economically valuable. Further, the analysts are required to make decision on how to meet user interests through a certain technical level as far as possible.

5 Related Work

Several researchers have developed techniques for eliciting requirements from online user feedback. Gebauer et al found the factors significantly related to overall user evaluation through the content analysis of online user reviews and then resulted in user requirements of mobile devices [5]. Lee et al. gathered customer's opinions from social network service to facilitate requirements elicitation [6]. These approaches capture changing requirements without limited range of users and insufficient expressions. However, they rely more on the manual content analysis of user opinions.

Cleland-Huang et al utilized a classification algorithm to detect non-functional requirements from stakeholder comments [19]. Hao et al. adopted machine learning techniques to extract the aspects of service quality from Web reviews for conducting automatic service quality evaluation [3]. Carreño et al. adapted topic modeling techniques to deal with available user feedback of mobile applications for extracting new/changed requirements for next versions [4]. Our previous research compared the changes in user satisfaction before and after software evolution to provide instructive information for designing future systems [18]. Although the existing approaches validate that automated techniques can efficiently explore user feedback for requirements evolution, they lack the deep analysis about how valuable user feedback of different system aspects are for determining changes in requirements.

In our research, opinion mining techniques make it possible to automatically elicit requirements from huge volume of user feedback data. Common approaches generally fall into two categories. One category is to identify user opinions through grammatical structures [7, 20-22]. Such approaches have good performance for mining fine-gained features and related opinion words. However, the completeness of extraction rules/ templates and domain knowledge have obvious impact on the accuracy of algorithms.

The other category is to use topic modeling techniques for simultaneously extracting and grouping user opinions [23-25]. These approaches are governed by how often feature terms and opinion words co-occur in different context. As expressions of user opinions are diverse, topic models are appropriate for mining coarse-gained features. A protential problem is that the extracted features may be not meaningful.

6 Conclusions

This paper has presented a novel approach for requirements evolution from the economic perspective. We explore a broad spectrum of online reviews and combine the techniques of opinion mining, machine learning, and text clustering with a utility-oriented econometric model to find system aspects significantly related to software sales for revising the requirements. A case study in the Internet security domain was carried out to show that our opinion mining method achieved good recall and precision in large amounts of noisy review data. Moreover, our approach supported analysts by suggesting the requirements changes that were more economically valuable. Therefore, it is useful to improve existing approaches for requirements evolution in understanding user demands in an open market.

Future work will refine our opinion mining method for improving the accuracy and efficiency of automated user feedback acquisition in the big data era. Furthermore, we will evaluate our approach using a broader data set from different domains.

Acknowledgments. The work in this paper was partially fund by National Natural Science Foundation of China under Grant No. 61170087 and State Key Laboratory of Software Development Environment of China under Grant No. SKLSDE-2012ZX-13.

References

1. Zowghi, D., Offen, R.: A Logical Framework for Modeling and Reasoning about the Evolution of Requirements. In: 3rd IEEE International Symposium on Requirements Engineering, pp. 247–257. IEEE Computer Society (1997)
2. Godfrey, M.W., German, D.M.: The Past, Present, and Future of Software Evolution. In: 2008 Frontiers of Software Maintenance, FoSM 2008, pp. 129–138 (2008)
3. Hao, J., Li, S., Chen, Z.: Extracting Service Aspects from Web Reviews. In: Wang, F.L., Gong, Z., Luo, X., Lei, J. (eds.) WISM 2010. LNCS, vol. 6318, pp. 320–327. Springer, Heidelberg (2010)
4. Galvis, L.V., Winbladh, K.: Analysis of User Comments: An Approach for Software Requirements Evolution. In: 35th International Conference on Software Engineering, pp. 582–591. IEEE Press, New York (2013)
5. Gebauer, J., Tang, Y., Baimai, C.: User Requirements of Mobile Technology: Results from A Content Analysis of User Reviews. Inf. Syst. E-Bus. Manage 6, 361–384 (2008)
6. Lee, Y., Kim, N., Kim, D., Lee, D., In, H.P.: Customer Requirements Elicitation based on Social Network Service. KSII Trans. on Internet and Information Systems 5(10), 1733–1750 (2011)

7. Qiu, G., Liu, B., Bu, J., Chen, C.: Opinion Word Expansion and Target Extraction through Double Propagation. Comput. Linguist. 37(1), 9–27 (2011)
8. Cortes, C., Vapnik, V.: Support-Vector Networks. Mach. Learn. 20(3), 273–297 (1995)
9. Kotler, P.: Marketing Management: Analysis, Planning, Implementation, and Control. Prentice Hall College Div. (1999)
10. Gabrilovich, E., Markovitch, S.: Computing Semantic Relatedness Using Wikipedia-based Explicit Semantic Analysis. In: 20th Int'l Joint Conf. on Artificial Intelligence, pp. 1606–1611. Morgan Kaufmann Publishers Inc. (2007)
11. Qiu, G., Liu, B., Bu, J., Chen, C.: Expanding Domain Sentiment Lexicon through Double Propagation. In: 21st Int'l Joint Conf. on Artificial Intelligence, pp. 1199–1204. Morgan Kaufmann Publishers Inc. (2009)
12. Pagano, D., Brügge, B.: User Involvement in Software Evolution Practice: A Case Study. In: 2013 Int'l Conf. on Software Engineering, pp. 953–962. IEEE Press, New York (2013)
13. Chevalier, J.A., Mayzlin, D.: The Effect of Word of Mouth on Sales: Online Book Reviews. Journal of Marketing Research 43(3), 345–354 (2006)
14. Cantone, I., Marucci, L., Iorio, F., Ricci, M.A., Belcastro, V., Bansal, M., Santini, S., di Bernardo, M., di Bernardo, D., Cosma, M.P.: A Yeast Synthetic Network for In Vivo Assessment of Reverse-Engineering and Modeling Approaches. Cell 137(1), 172–181 (2009)
15. Granger, C.W.J.: Testing for Causality: A Personal Viewpoint. Journal of Economic Dy-namics and Control 2, 329–352 (1980)
16. Jiang, W., Zhang, L., Dai, Y., Jiang, J., Wang, G.: Analyzing Helpfulness of Online Reviews for User Requirements Elicitation. Chinese Journal of Computers 36(1), 119–131 (2013)
17. Goolsbee, A., Chevalier, J.: Measuring Prices and Price Competition Online: Amazon.com and Barnesand Noble.com. Quantitative Marketing and Economics 1, 203–222 (2003)
18. Lew, P., Olsina, L., Becker, P., Zhang, L.: An Integrated Strategy to Systematically Understand and Manage Quality in Use for Web Applications. Requirements Engineering 17(4), 299–330 (2012)
19. Cleland-Huang, J., Settimi, R., Xuchang, Z., Solc, P.: The Detection and Classification of Non-functional Requirements with Application to Early Aspects. In: 14th IEEE Int'l Requirements Engineering Conf., pp. 39–48. IEEE CS (2006)
20. Hu, M., Liu, B.: Mining and Summarizing Customer Reviews. In: Tenth ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining, pp. 168–177. ACM, New York (2004)
21. Popescu, A., Etzioni, O.: Extracting Product Features and Opinions from Reviews. In: Conf. on Human Language Tech. and Empirical Methods in Natural Language Processing, pp. 339–346. ACL (2005)
22. Wu, Y., Zhang, Q., Huang, X., Wu, L.: Phrase Dependency Parsing for Opinion Mining. In: Conf. on Empirical Methods in Natural Language Processing, pp. 1533–1541. ACL (2009)
23. Mei, Q., Ling, X., Wondra, M., Su, H., Zhai, C.: Topic Sentiment Mixture: Modeling Facets and Opinions in Weblogs. In: 16th Int'l Conf. on World Wide Web, pp. 171–180. ACM, New York (2007)
24. Zhao, W., Jiang, J., Yan, H., Li, X.: Jointly Modeling Aspects and Opinions with A Max-Ent-LDA Hybrid. In: Conf. on Empirical Methods in Natural Language Processing, pp. 56–65. ACL (2010)
25. Jo, Y., Oh, A.H.: Aspect and Sentiment Unification Model for Online Review Analysis. In: 4th ACM Int'l Conf. on Web Search and Data Mining, pp. 815–824. ACM (2011)