For User-Driven Software Evolution: Requirements Elicitation Derived from Mining Online Reviews

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

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

Abstract. Online reviews that manifest user feedback have become an available resource for eliciting requirements to design future releases. However, due to complex and diverse opinion expressions, it is challenging to utilize automated analysis for deriving constructive feedback from these reviews. What's more, determining important changes in requirements based on user feedback is also challenging. To address these two problems, this paper proposes a systematic approach for transforming online reviews to evolutionary requirements. According to the characteristics of reviews, we first adapt opinion mining techniques to automatically extract opinion expressions about common software features. To provide meaningful feedback, we then present an optimized method of clustering opinion expressions in terms of a macro network topology. Based on this feedback, we finally combine user satisfaction analysis with the inherent economic attributes associated with the software's revenue to determine evolutionary requirements. Experimental results show that our approach achieves good performance for obtaining constructive feedback even with large amounts of review data, and furthermore discovers the evolutionary requirements that tend to be ignored by developers from a technology perspective.

Keywords: Software evolution, requirements elicitation, online reviews, opinion mining, user satisfaction analysis.

1 Introduction

Successful software systems are always able to evolve as stakeholder requirements and environments where the deployed systems operate [1]. In today's competitive market, meeting changing user demands[1] is a critical driving factor of software evolution. The software system should adapt to the social environment where users form opinions based on their experience with it [3]. Therefore, systematically and effectively eliciting evolutionary requirements is critical for the software to adapt and improve.

[1] In economics, demand is an economic principle that describes a consumer's desire, willingness and ability to pay a price for a specific good or service.

User feedback provides useful information that can help to improve software quality and identify missing features [4]. However, with software delivered via the Internet, the scopes and types of users are uncertain before delivering software systems, since there are differences of space and time between users and developers. Fortunately, user generated content is becoming mainstream in web platforms, and consumers are willing to publish online reviews to express their opinions about software systems. These reviews that manifest user demands in real contexts of use have become an available feedback resource for eliciting requirements to design future software releases. Moreover, the reviews that come from large quantities of disparate users contain abundant data and its expressions [5]. For example, the Apple App Store has more than five hundred million active registered users and billions of online reviews.

Current software engineering research and practice favor requirements elicitation derived from online reviews. Several approaches have been developed, regarding the techniques and processes to consolidate, analyze and determine requirements in accordance with online feedback [6-7]. However, these approaches mostly rely on manual content analysis and as such, are not efficient for dealing with large amounts of online reviews in order to shorten time-to-market. Obviously, automated techniques, such as text mining, information retrieval, and machine learning can be effective tools for identifying software features and associated opinions mentioned in user comments [8-9]. However, due to complex and diverse opinion expressions, it is challenging to utilize automated analysis for accurately deriving constructive feedback from the reviews of software systems. What's more, assisting developers with determining evolutionary requirements based on user feedback is also challenging.

In this paper, we present a systematic approach for the transformation of online reviews to evolutionary requirements. For the first problem of automated text analysis we analyze the characteristics of online software reviews and then adapt the syntactic relation-based propagation approach (SRPA) to automatically identify opinion expressions about common software features. In order to provide meaningful feedback, we present a method S-GN for clustering opinion expressions from a network perspective, which uses the Girvan-Newman algorithm with the Explicit Semantic Analysis similarity. To address the second problem of assisting developers, we consider an economic impact to analyze user satisfaction based on this feedback and then determine important changes for requirements.

The contributions of our research are as follows:

- SRPA+, an expanded version of SRPA, is more suitable for mining complete opinion expressions from software reviews.
- The proposed method S-GN optimizes the clusters of opinion expressions by considering a macro network topology.
- We combine user satisfaction analysis with the inherent economic attributes associated with the software revenue to determine the evolutionary requirements.
- We show experimentally that our approach achieves good performance for deriving constructive feedback even with large amounts of review data, and furthermore discovers the evolutionary requirements that tend to be ignored by developers.

The remainder of this paper is organized as follows. Section 2 elaborates our approach. Section 3 describes the experiment and analyzes the results. Section 4 introduces related work and Section 5 discusses conclusions and future work.

2 Systematic Approach of Requirements Elicitation

In our approach, we first extract opinions about software features from large amounts of online reviews and determine whether opinions are positive or negative. Then, we categorize opinions and select corresponding review sentences to represent the feedback on software features. Finally, we generate a document of evolutionary requirements from an economic perspective.

2.1 Extracting Targets and Sentiment Words

A user opinion mentioned in a review is defined as a target-sentiment pair. The target is a topic on the software feature that is a prominent or distinctive user-visible aspect, quality, or characteristic of the system [10]. The sentiment is the user evaluation of its target. Sentiment words and targets are often related syntactically and their relations can be modeled using the dependency grammar [11]. There have been many methods used to model sentiment words, targets and their dependency relationships [21-23]. We adapt the syntactic relation-based propagation approach (SRPA) due to its modeling naturally for the opinion mining task. SRPA extracts targets and sentiment words iteratively using known and extracted words through the identification of syntactic relations [12]. The bootstrapping process starts with a seed sentiment lexicon.

The key of SRPA is the propagation rules based on syntactic relations. As targets and sentiment words are usually nouns/noun phrases and adjectives respectively [13], SRPA only defines the relations between nouns and adjectives, between adjectives, and between nouns. However, software systems have the dynamic features of computing and processing. Users may comment on the software's behavior and impact on the application environment using opinion expressions that depend on the relations between verbs and adverbs. In addition, sentiment verbs such as *love* can also express opinions. Accordingly, we define new propagation rules based on Stanford POS tagger[2] and syntactic parser[3], as shown in Table 1.

Table 1. New propagation rules

ID	Description	Output	Example
$R1_1$	$S \rightarrow S\text{-}Dep \rightarrow T$ *s.t.* $S \in \{S\}$, $POS(T) \in \{VB\}$, $S\text{-}Dep \in \{MR\}$	$t=T$	The software updates quickly.
$R1_2$	$S \rightarrow S\text{-}Dep \rightarrow T$ *s.t.* $T \in \{T\}$, $POS(S) \in \{RB\}$, $S\text{-}Dep \in \{MR\}$	$s=S$	This software operates well with Firefox.
R2	$T_{i(j)} \rightarrow T_{i(j)}\text{-}Dep \rightarrow T_{j(i)}$ *s.t.* $T_{j(i)} \in \{T\}$, $POS(T_{i(j)}) \in \{VB\}$, $T_{i(j)}\text{-}Dep \in \{CONJ\}$	$t=T_{i(j)}$	It is easy to download and update the software.
R3	$S_{i(j)} \rightarrow S_{i(j)}\text{-}Dep \rightarrow S_{j(i)}$ *s.t.* $S_{j(i)} \in \{S\}$, $POS(S_{i(j)}) \in \{RB\}$, $S_{i(j)}\text{-}Dep \in \{CONJ\}$	$s=S_{i(j)}$	Norton runs smoothly and quietly.
R4	$S \rightarrow S\text{-}Dep \rightarrow T$ *s.t.* $S \in \{S\}$, $POS(T) \in \{NN\}$, $S\text{-}Dep \in \{OBJ\}$	$t=T$	Installation destabilizes the correct operation.
R5	$S \rightarrow S\text{-}Dep \rightarrow T$ *s.t.* $S \in \{S\}$, $POS(T) \in \{VB\}$, $S\text{-}Dep \in \{COMP\}$	$t=T$	I love to use Kaspersky.

**S* (or *T*) is the sentiment word (or target). {*S*} (or {*T*}) is the set of known sentiment words (or the set of targets). *POS*(*S*) (or *POS*(*T*)) represents the POS tag of the sentiment word (or target). {*RB*}: *RB*, *RBR*, *RBS*; {*VB*}: *VB*, *VBD*, *VBG*, *VBN*, *VBP*, *VBZ*; {*NN*}: *NN*, *NNS*, *NNP*. *S-Dep* (or *T-Dep*) represents the dependency relation of the word *S* (or *T*). {*MR*}: *advmod*; {*CONJ*}: *conj*; {*OBJ*}: *dobj*, *pobj*; {*COMP*}: *xcomp*.

2.2 Identifying Complete Expressions of Opinions

Since the targets extracted in the previous step are individual words, we need to identify the complete target expressions. Generally, a sentence clause contains only one target-sentiment pair unless there are conjunctions [12]. However, in the sentence

[2] http://nlp.stanford.edu/software/tagger.shtml

[3] http://nlp.stanford.edu/software/lex-parser.shtml

Kaspersky didn't update to a correct version quickly, there are two target-sentiment expressions, namely *Kaspersky update quickly* and *a correct version*. Therefore, we further merge the opinion expressions on the same software feature, and remove the noisy ones caused by unconstrained propagation rules.

The target words consist of nouns and verbs. For noun target expressions, we exploit the phrase-structure trees to identify the noun phrases that contain the extracted target words. There are two cases of verb target expressions. If the target words are verbs with direct objects or prepositional objects, the verb phrases in the phrase-structure trees serve as the target expressions. Note that the parser doesn't distinguish prepositional objects and adverb prepositional phrases. We check the compound phrases of verbs and prepositions through the online dictionary[4]. If the target words are verbs without objects, the verbs and their subjects compose the target expressions.

Based on the identified target expressions, we employ the following two rules to merge the opinion expressions in the same sentence. **Rule 1**: if the opinion expressions share some words, they must describe the same software feature. **Rule 2**: if the words in opinion expressions have direct dependency relations, they are likely to represent the same software feature.

We prune noisy opinion expressions according to word frequency. The sentiment words, negations, and stop words are removed from the merged opinion expressions and then the scores of processed expressions are calculated as follows:

$$O-Score(e_i)=\alpha*\frac{W_{e_i}}{W_E}+\beta*\frac{N_{e_i}}{N_E}+\gamma*\frac{F_{e_i}}{F_E} \tag{1}$$

where e_i is a processed expression; E is the set of processed expressions; W_x is the number of words in x; N_{ei} is the number of processed expressions that contain the words in e_i; N_E is the number of processed expressions in E; F_x is the number of frequent words in x; α, β, and γ are weights. For a sentence, we only choose the opinion expession with the highest score unless there are conjunctions.

2.3 Assigning Sentiment Polarities to Opinion Expressions

We propose a two-stage method for assigning polarities to opinions. In the first stage, the polarities for the newly extracted sentiment words are inferred through the rules in [12]. The extracted words are assigned with the same polarities as the known words in the current review. The polarity changes when there are an odd number of negations or contrary words between the extracted word and the known word. The polarity value of the sentiment word that has either no or multiple polarities is computed as the sum of polarity values of known sentiment words in the current review.

In the second stage, the polarity score of a complete opinion expression is estimated by the following ordered weighted averaging (OWA) aggregation [14] of the contained sentiment words.

$$p=\sum_{i=1}^{m}\frac{w_{s_i}}{N_{s_i}}s_i \tag{2}$$

[4] http://thesaurus.com/

where s_i is the polarity of a sentiment word in the opinion expression; N_{si} is the number of sentiment words that have the same type with s_i; w_{si} is the OWA weight of s_i. The types of sentiment words consis of verbs (*v*), adjectives (*adj*), and adverbs (*adv*). As verbs are the core of sentences whereas adjectives and adverbs are modifiers, the types of sentiment words are ranked as (*v*, *adv*, *adj*) from big to small according to their importance. The value of w_{si} is concerned with the position of the type of s_i. The OWA operators aggregate both the polarities of sentiment words and the importance of their types. If the polarity score of an opinion expression is greater than 0, its polarity is positive, and otherwise negative. The polarity changes if there are an odd number of negations associated with sentiment words in the opinion expression.

2.4 Organizing Opinions into Structured Feedback

In order to provide meaningful feedback, we first group similar opinion expressions about software features. Each category represents a unique overall, functional or quality requirement. Then we produced the structured feedback classified by software features, including several corresponding sentences.

The opinion expressions and the semantic associations between them construct an undirected graph $G = (V, E)$, where V is the set of vertices for the opinion expressions and E is the set of edges for the semantic links between any two vertices. There is a semantic link between two opinion expressions if their semantic similarity is greater than a certain threshold λ. We rely on the Explicit Semantic Analysis (ESA) algorithm to compute the semantic similarity between opinion expressions. ESA first builds an inverted index for all Wikipedia concepts, then represents any text as a weighted vector of Wikipedia concepts by retrieving its terms from the inverted index and finally assesses the relatedness in concept space using conventional metrics [15].

Based on the graph G, we adopt the Girvan-Newman (GN) algorithm to cluster the opinion expressions. The GN algorithm is a typical method for detecting network communities. Without the prior information for determining the centers of communities, the GN algorithm constructs communities by progressively removing edges from the original graph [16]. Each community represents a feature category.

For describing the feedback of each feature category, we select 3-5 sentences in which the opinion expressions are nearest to the center of the category. Finally, we manually label the names of feature categories in accordance with the corresponding sentences and then merge the feature categories with the same name.

2.5 Generating Document of Evolutionary Requirements

The feedback for each feature category implies an overall, function or quality requirement of the software system. However, it is critical to assess the priority and importance of feedback for software evolution. Because user satisfaction is the best indicator of a company's future profits [17], it is the direct driving factor of software evolution. From an economic perspective, our decision to determine evolutionary requirements depends on the user satisfaction indexes evaluated by the feedback for software features. The satisfaction score of a software feature is defined as follows:

$$S-Score(f)=\frac{1}{\left(n_p+n_n\right)}\left(\sum_{i=1}^{n_p}P_i\left(f\right)+E_d*\sum_{i=1}^{n_n}N_i\left(f\right)\right) \tag{3}$$

where n_p and n_n are the numbers of positive and negative opinion expressions about the software feature *f*; $P_i(f)$ and $N_i(f)$ are the positive and negative polarity value of an opinion expression about the software feature *f*; E_d is the price elasticity of demand, which is a measure used in economics to show the responsiveness, or elasticity, of the quantity demanded of a good or service to a change in its price[5]. Formulated as [18], E_d indicates the substitutability and importance of the software in customer purchases. Introducing E_d to user satisfaction evaluation emphasizes user acceptance of the technological level in the market environment. If E_d is greater, even a few negative opinions may result in so massive loss of users to reduce the software revenue.

According to Equation (3), if the user satisfaction score of a software feature is greater than 0, the corresponding requirement is reusable or added, and changed otherwise. We compute the user satisfaction score for high frequent features mentioned in the reviews and then manually generate the document of evolutionary requirements in the light of those review sentences for each software features. Those evolutionary requirements that drive more economic gain are prioritized systematically.

3 Experiment

To demonstrate the practicality of our approach in eliciting evolutionary requirements even with large amounts of online reviews, we first introduce the data sets and settings. Then, we evaluate the opinion mining techniques including the identification and classification of opinions. Finally, we analyze the usefulness of the evolutionary requirements document for developers.

3.1 Data Sets and Settings

As can be seen in Table 2, we used two data sets of online reviews: the packaged software of Kaspersky Internet Security 2011 3-Users (KIS 2011) from Amazon.com and the mobile application of TuneIn Radio Pro V3.6 (TuneIn 3.6) from the Apple App Store. For each testing data set, we manually labeled the potential software features, opinions and their polarities mentioned in the reviews, and then classified the review sentences according to the semantics of related opinions.

Table 2. Statistics of the data sets of online reviews

Data set	#Reviews	#Sentences	#Words	#Sentence per review	#Words per sentence
KIS 2011	380	3392	52682	8.9	15.5
TuneIn 3.6	461	1211	12711	2.6	10.5

The Stanford POS Tagger and parser are respectively used to tag and parse the data sets. The seed sentiment lexicon is provided by Liu's sentiment words[6]. The Wikipedia

[5] `http://en.wikipedia.org/wiki/Price_elasticity_of_demand`
[6] `http://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html`

version on Sep. 1, 2011 is adopted for computing the ESA semantic similarity. α=0.5, β=0.3, γ=0.2, and λ=0.4 are experimentally set for our approach.

As the exact quantity demanded for determining E_d is not available, we modified the evaluation model of user satisfaction in Section 2.5 by replacing E_d with the price elasticity of sales rank E_r. The sales rank implies the demand for a software product/ application relative to other competitors. Since it is observed that the association between sales rank and demand approximately conforms to a Pareto distribution, the formula of E_r is similar with $-E_d$.

3.2 Evaluation of Opinion Identification

As shown in Table 3, our method for opinion identification, SRPA+, outperforms SRPA in all conditions. Significantly higher recall, especially for KIS 2011 indicates that the new propagation rules work well. The improvement in precision implies that merging opinion expressions avoids one-sided identification. Note that the results of TuneIn 3.6 produced by SRPA+ are basically lower than those of KIS 2011. This is because the decreased performance of natural language processing techniques for dealing with more phrases and incomplete sentences in the reviews of TuneIn 3.6.

Table 3. The comparison results of opinion identification

Data set	Recall		Precision		F-score	
	SRPA	SRPA+	SRPA	SRPA+	SRPA	SRPA+
KIS 2011	0.67	0.83	0.74	0.78	0.70	0.80
TuneIn 3.6	0.69	0.81	0.71	0.73	0.70	0.77

Table 4 illustrates the results of polarity assignment using SRPA+. Clearly, the good recall reveals that the propagation performs well in discovering new sentiment words. We can observe that the precision of opinion expressions is significantly higher than that of new sentiment words. There are two main reasons for relatively worse performance of new sentiment words in precision. First, the review data sets often have errors of spelling and grammatical structure or non-standard sentences so that automatic tagging and parsing don't always work correctly. Second, the propagation rules have only the constraints of POS tags so that more ordinary words are introduced with the increase of the review data sets. In spite of this, our methods of merging and pruning opinion expressions reduce the effect of noisy sentiment words.

Table 4. The results of polarity assignment

Data set	Recall			Precision			F-score		
	New words	Words	Expressions	New words	Words	Expressions	New words	Words	Expressions
KIS 2011	0.71	0.73	0.76	0.62	0.67	0.72	0.66	0.70	0.73
TuneIn 3.6	0.68	0.71	0.72	0.63	0.65	0.70	0.65	0.68	0.71

3.3 Evaluation of Opinion Classification

The proposed method for opinion classification is called S-GN. The baseline methods include J-Kmeans and S-Kmeans. Both are k-means clustering algorithms based on the Jaccard coefficient and the ESA similarity respectively. S-GN produces k clusters automatically. The inputs of J-Kmeans and S-Kmeans are set as the same number of clusters produced by S-GN.

Table 5 shows the results of opinion classification in recall, precision and F-score. S-GN and S-Kmeans significantly outperform J-Kmeans in all conditions. S-GN has better results than S-Kmeans especially in precision. Such results imply that the ESA similarity is better than the Jaccard coefficient. The Jaccard coefficient is essentially a keyword-based similarity measurement. As ESA enhances the semantic representations of texts using expanded Wikipedia concepts, it alleviates the clusters of duplicated categories. In addition, the k-means algorithm requires a priori number of clusters. It is difficult to optimize *k* seeds for avoiding poor clusters. The GN algorithm produces the optimized number of clusters considering the global network topology so that it reduces the clusters containing mixed categories.

Table 5. The comparison results of opinion classification

Data set	Recall			Precision			F-score		
	J-Kmeans	S-Kmeans	S-GN	J-Kmeans	S-Kmeans	S-GN	J-Kmeans	S-Kmeans	S-GN
KIS 2011	0.58	0.68	0.71	0.63	0.72	0.76	0.60	0.70	0.73
TuneIn 3.6	0.55	0.66	0.67	0.60	0.71	0.74	0.57	0.68	0.70

3.4 Evaluation of Generated Evolutionary Requirements Document

We organized a human subjective study to evaluate the usefulness of the generated evolutionary requirements document. 50 participants that have over three years experience in software development were required to report the evolutionary requirements for TuneIn 3.6. We compared the generated document with participants' reports to validate that our approach can discover requirements that were ignored by developers.

In the first stage, the participants make decisions about requirements evolution based on their experience with TuneIn 3.6. Table 6 indicates the common results designed by developers, with which more than 30% of the participants agreed. Table 7 shows the results generated by our approach. For functional requirements, developers paid more attention to key, special and value-added features while users were more concerned with the features relative to user habits. As TuneIn 3.6 is one of the best sellers in the *music* category, its main functional features are implicitly desirable by the mass market. Such features should be improved or changed only when there are significant issues and bugs, for example "*pause*". However, developers have difficulty predicting user preferences for these features in the real world, such as "*favorites*", "*schedule*" and "*streaming*". In terms of quality requirements, developers assessed the objective features for Internet Radio applications as "*connection*", "*reliability*", and "*efficiency*" whereas users revealed subjective features like "*operability*". Consequently, user feedback can assist in determining evolutionary requirements that developers sometimes overlook.

Table 6. Evolutionary requirements of TuneIn 3.6 designed by developers

ID	Type	Requirements	Feature	#Participants
1	functionality	Fix issues that are causing playback or stream errors	record	15
2		Fix issue that is causing skipping to the current radio after pausing	pause	21
3	quality	Make connection smoother and more stable	connection	32
4		Fix issues that are causing crashing, freezing, and restarting	reliability	48
5		Improve the speed for connecting stations and favorites in personal account	efficiency	26

Table 7. Evolutionary requirements of TuneIn 3.6 generated by our approach

ID	Type	Requirements	Feature	Frequency	E-score
1	functionality	Improve reliability during pausing and the quality after pausing	pause	17	-0.263
2		Keep the schedule of stations	schedule	16	-0.789
3		Keep the local favorites	favorites	19	-0.413
4		Provide high quality streaming and allow users to choose streaming quality	streaming	39	-0.193
5	quality	Work in the background	operability	16	-0.678
6		Make connection smoother and more stable	connection	18	-0.392
7		Reduce the crash, freezing, reboot and skipping during listening to radios	reliability	35	-0.432

In the second stage, we provided the structured feedback derived from mining the reviews of TuneIn 3.6 for the participants to revise their presented evolutionary requirements. The most common strategy used by 76% participants is to decide the priority of feedback on each software feature by the frequency of its occurrence and its user satisfaction. The features revised by the participants contains "*user interface*" in addition to those in Table 7. The ordinary satisfaction score and economic satisfaction score of the feature are -0.0714 and 0.0489. Intuitively, it is difficult to determine evolving "*user interface*" because its ordinary satisfaction is only slightly negative. However, the economic satisfaction indicates the acceptance of a feature compared with the overall technical level in the market. Although the negative ordinary satisfaction score implies that "*user interface*" implemented in TuneIn 3.6 is lower than common user expectation with it, the positive economic satisfaction score suggests that the same is true for other competitors in the market. Even if users are not satisfied with TuneIn 3.6, they have no other better alternatives. In other words, improving "*user interface*" cannot significantly have positive impact on the revenue of TuneIn 3.6. Thereby, our approach argues even though a feature of the software product or application has poor ordinary satisfaction, it does not have to be changed until its economic satisfaction is negative. Determining evolutionary requirements based on economic satisfaction manifests that software evolution does not blindly pursue user interests, but rather balances the interests of users and developers.

In addition, "*user interface*" is a feature relative to specific contexts of use. As TuneIn 3.6 is an application for the public, the requirements for "*user interface*" are diverse in terms of specific user habits. To reduce the potential risk, this type of features should only be improved when it has significantly low user satisfaction.

4 Related Work

In other research, there are techniques developed for eliciting requirements from online user feedback. Gebauer et al. use content analysis of user reviews to identify functional and non-functional requirements of mobile devices through finding the factors that are significantly related to overall user evaluation [6]. Lee et al. elicit customer requirements using their opinions gathered from social network services [7]. Such works capture changing requirements without limited range of users and insufficient expressions. However, these approaches mostly rely on manual content analysis.

Cleland-Huang et al. adopt a classification algorithm to detect non-functional requirements (NFRs) from freeform documents including stakeholder comments [19]. One problem is that limited documents hinder identifying changing NFRs in a timely manner. Hao et al. utilize machine learning techniques to extract the aspects of service quality from Web reviews for conducting automatic service quality evaluation [8]. Carreño et al. adapt information retrieval techniques including topic modeling for exploring the rich user comments of mobile applications to extract new/changed requirements for future releases [9]. Li et al. compare the changes in user satisfaction before and after software evolution to provide instructive information for designing future systems [20]. Although these approaches initially access the validity of using automated techniques to discover software requirements, they lack the deep analysis about how user feedback influences changes in requirements.

Our research is inspired by opinion mining techniques that make it possible to automatically elicit requirements from huge volumes of user feedback data. The mainstream approaches are divided into two categories. One is to identify opinions based on word co-occurrence and grammatical structures [21-23]. Such approaches have good performance for extracting fine-grained features as well as opinions. However, the integrity of extraction rules/templates and domain knowledge have an obvious impact on their accuracy. The other one is to identify and group opinion pairs using topic modeling [24-26]. While it is not hard for topic models to find those very general and frequent features from a large document collection, it is not easy to find those locally frequent but globally not so frequent features [2].

5 Conclusions

This paper presented a novel approach for eliciting evolutionary requirements through analysis of online review data by integrating various techniques of SRPA+, S-GN and user satisfaction analysis including economic factors. To conduct our research, we first accessed a broad spectrum of review data with complex and diverse opinion expressions and then evaluated the performance of automated techniques for consolidating, analyzing and structuring feedback information. Furthermore, the proposed method of user satisfaction analysis assisted developers with finding a set of evolutionary requirements associated with the software revenue. We reported a human subjective study with fifty developers, evaluating the usefulness of the evolutionary requirements document generated by our approach. As a result, the generated document could help developers understand why and what to evolve for future software releases. In particular, they were led to focus on the improvements in specific functions and quality in use that they had previously ignored.

Future work will refine our opinion mining method to improve the performance of automated requirements elicitation in the big data era. In addition, we will further 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. Nuseibeh, B., Easterbrook, S.M.: Requirements Engineering: A Roadmap. In: 2000 Conf. on The Future of Software Engeering, pp. 35–46. ACM, New York (2000)

2. Liu, B.: Sentiment Analysis and Opinion Mining. Morgan & Claypool Publishers (2012)
3. 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)
4. 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)
5. Vasa, R., Hoon, L., Mouzakis, K., Noguchi, A.: A Preliminary Analysis of Mobile App User Reviews. In: 24th Conf. on Australian Computer-Human Interaction, pp. 241–244. ACM, New York (2012)
6. 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)
7. 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)
8. 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)
9. Galvis, L.V., Winbladh, K.: Analysis of User Comments: An Approach for Software Requirements Evolution. In: 35th International Conference on Software Engeering, pp. 582–591. IEEE Press, New York (2013)
10. Kang, K.C., Cohen, S.G., Hess, J.A., Novak, W.E., Peterson, A.S.: Feature-Oriented Domain Analysis (FODA) Feasibility Study. Carnegie-Mellon University Software Engeering Institute (1990)
11. Tesniere, L.: Élements de Syntaxe Structurale: Préf. de Jean Fourquet. C. Klincksieck (1959)
12. Qiu, G., Liu, B., Bu, J., Chen, C.: Sentiment Word Expansion and Target Extraction through Double Propagation. Comput. Linguist. 37, 9–27 (2011)
13. 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)
14. Yager, R.R.: On Ordered Weighted Averaging Aggregation Operators in Multicriteria Decision Making. IEEE Trans. on Systems, Man and Cybernetics 18(1), 183–190 (1988)
15. 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)
16. Girvan, M., Newman, M.E.: Community Structure in Social and Biological Networks. The National Academy of Sciences 99(12), 7821–7826 (2002)
17. Kotler, P.: Marketing Management: Analysis, Planning, Implementation, and Control. Prentice Hall College Div. (1999)
18. Michael, P., Melanie, P., Kent, M.: Economics. Addison-Wesley, Harlow (2002)
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 Engeering Conf., pp. 39–48. IEEE CS (2006)
20. Li, H., Zhang, L., Zhang, L., Shen, J.: A User Satisfaction Analysis Approach for Software Evolution. In: Int'l. Conf. on Progress in Informatics and Computing, pp. 1093–1097. IEEE Press, New York (2010)
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. Ding, X., Liu, B., Yu, P.S.: A Holistic Lexicon-based Approach to Opinion Mining. In: Int'l Conf. on Web Search and Web Data Mining, pp. 231–240. ACM (2008)
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: Fourth ACM Int'l Conf. on Web Search and Data Mining, pp. 815–824. ACM (2011)
26. 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)