Requirement Acquisition from Social Q&A Sites

Ming Xiao(✉), Gang Yin, Tao Wang, Cheng Yang, and Mengwen Chen

National Key Laboratory for Parallel and Distributed Processing, College of Computer Science, National University of Defense Technology, Changsha, Hunan Province, China
xiaoming-7@qq.com, {jack.nudt,taowang2005}@nudt.edu.cn

Abstract. Social Q&A sites have changed the way of knowledge sharing in software communities. Comparing to the traditional mail-list, bug/change repositories, software forums and software marketplaces, users and developers are more active in social Q&A sites, and social Q&A sites are more open and free. The feedbacks from users have much potential valuable information, such as feature requests, bugs or sentiment, but there also exists lots of noise especially for social Q&A sites. How to mine the useful information from the feedbacks in social Q&A sites has become a problem. This paper focuses on the feature requirements in requirement acquisition, which can be used to assist software development. We propose an effective approach, which combines Support Vector Machine (SVM) with requirement dictionary to find the questions about feature requests from the posts in social Q&A sites. We evaluate the approach on available dataset, and compare it to the other different approaches. The results show that the automatically requirement acquisition through improved SVM approach is useful and can significantly decreases the manual effort.

Keywords: Requirement acquisition · Feature requests · Q&A sites

1 Introduction

In software development, requirement acquisition is very important, and the software can be successful depends on whether it meets the need of users. So user feedbacks are valuable for software development. With the development of Internet technologies, more and more software developments rely on Internet communities to gain the feedbacks or to communicate with users. More and more users also join in the communities to involve in the software development process, especially provide valuable suggestions for software developments. For example, in App Store users deliver the feedback after they use an app, and the information is useful for developers because it can assist developers to know what users hope the app to be and what features should be added in next versions.

There are many kinds of software communities, such as software forums, mail-list, and software marketplace. These years have witnessed the rapid development of social Q&A sites because they can provide better environments for

L. Liu and M. Aoyama (Eds.): APRES 2015, CCIS 558, pp. 64–74, 2015.
DOI: 10.1007/978-3-662-48634-4_5

users to express their views or exchange their opinions. As the biggest programming Q&A site, Stack Overflow (www.stackoverflow.com) has achieved a great success in these years and is gaining more and more focuses. This can be a good description of the advantages of social Q&A sites. Bogdan Vasilescu [1] finds that the users in mail-list are migrating to Stack Exchange, a network of 145 communities that are created and run by experts and enthusiasts who are passionate about a specific topic [2]. Stack Exchange builds libraries of high-quality questions and answers focusing on each community's area of expertise, such as Stack Overflow (www.stackoverflow.com) for programming problems, WordPress (wordpress.stackexchange.com) for WordPress development, Android Enthusiasts (android.stackexchange.com) for enthusiasts and power users of the Android operating system, Unix&Linux (unix.stackexchange.com) for users of Linux, FreeBSD and other Unix-like operating systems. So the feedbacks in these Q&A sites have large numbers of valuable information for software development, and the number of questions is huge and increases quickly. But the questions are about many topics such as bugs, feature requests or even complaints. In this paper we focus on feature requests which are new features or re-design of existing features, and we hope to extract the questions about feature requests from the whole questions in social Q&A sites to assist software development.

In short, this paper makes the following contributions:

- Firstly focus on the requirement acquisition for software development from social Q&A sites which have become the most active communities;
- Based on the previous research, implement the approach based on linguistic rules to extract the valuable questions about feature requests;
- Propose a requirement acquisition approach based on Support Vector Machine (SVM) and improve the process of feature selecting based on TF-IDF and requirement dictionary about feature requirements;
- Evaluate the approach based on SVM and the approach based on linguistic rules, and the results show that our approach can get better performance and decrease the manual effort significantly.

2 Related Work

It seems that there are many similar studies on requirement acquisition. C. Iacob and R. Harrison [3] designed a MARA (Mobile App Requirement Analyzing) prototype, which can be used to automatically index the mobile app feature requests from online views. They got a set of 237 linguistic rules and used them to refer the other reviews, and they ranked the reviews based on the frequency and length of feature requests. M. Harman, Y. Jia, and Y. Zhang [4] focused on feature requests information from the text of app store descriptions. L. V. Galvis Carreño and K. Winbladh [5] alleviated the process of developer getting information for the next version of software through automatic topics extraction.

To our best knowledge, the most similar work to ours is AR-Miner [6], which facilitated mobile app developers to discover the most "informative" user reviews from a large and rapidly increasing pool of user reviews. But our work focus on social Q&A sites and the questions posted by users to extract the valuable questions about feature requests to assistant the software development.

3 Approach

In this section, we describe our approach based on SVM and the approach based on linguistic rules in detail. Firstly, we introduce how we handle the natural language text from social Q&A sites. Then we introduce the SVM and how to identify the features in SVM. At last, we present the approach based on linguistic rules in detail.

3.1 Preprocessing

Natural language text is unstructured, and we have to preprocess the text to get a structured data that the computer can recognize and compute. The general method to preprocess the natural lounge is VSM (Vector Space Model), which can translates the natural language text into vector space. In our approach, we also process the text with VSM, and the steps are listed as follows:

step 1 Get the texts and delete the useless tags such as HTML tags. The only valuable information in our method is pure text that users have posted;
step 2 Lowercase the words, because we needn't consider the letter case;
step 3 Tokenize the text to words and these words will be used to build a vector to represent the text;
step 4 Remove the punctuations, which are useless, so we need to remove them automatically;
step 5 Remove the stop words, which often are used in English expression but contain little information, and there is no need to keep these words;
step 6 Stem the words to root words, because the words in text have different tenses, but we needn't consider the difference and should regard them as the same;
step 7 Remove the words that are used less than 3 times.

After upper processing we have mapped the natural language text to vector space, and the vector will be used to compute with the computer.

3.2 Constructing Classifier Based on SVM

Our core work is to construct a classification machine which can automatically classify the documents into two categories, of which one is about feature requests and the other is about non-feature requests. We choose Support Vector Machines (SVM), which is widely used in text classification and fits our problem. SVM is a supervised learning model in machine learning field. In supervised learning, we need the training data that consists of a set of training examples. A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples [7]. In our approach, we input the training data that consist of training examples labeled as feature (F) or non-feature (NF) to SVM to train a classifier. After that, when a new question is inputted in the trained SVM model, the model will return the result about whether this question is about feature request or not. Fig. 1 depicts the overview of SVM approach in detail.

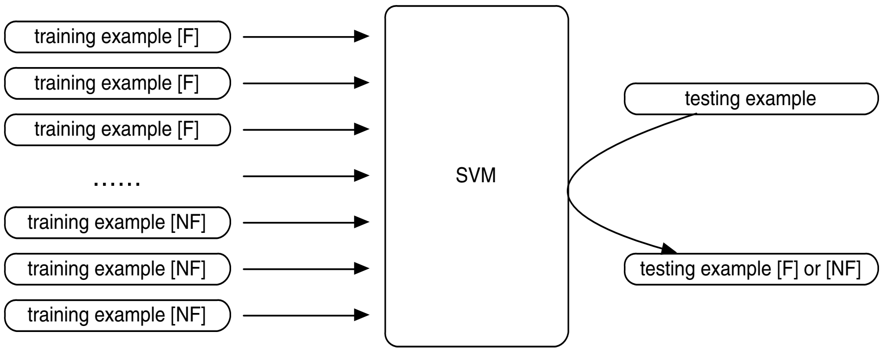

Fig. 1. Overview of SVM approach

The core idea of SVM is not complicated. SVM constructs a hyper-plane to separate the two kinds of training examples. In our experiment, we are lucky to use a successful and widely used SVM implementation named LIBSVM [8]. LIBSVM is a library for Support Vector Machines (SVMs) [9]. It is developed to help users to easily apply SVM to their applications by professor Chang from national Taiwan University.

3.3 Feature Selection in SVM

Focusing on the questions in social Q&A sites, we regard the question as a document. The weight of every word in a document is not the same. It is obvious that a word is used in a document frequently and is rarely used in other documents, and the weight of the word should be higher. On the contrary, a word is used in document rarely but is often used in other documents, and the weight of the word should be lower. So we should evaluate the importance of words and compute the weight of every word. Term Frequency-Inverse Document Frequency (TF-IDF) is widely used to compute the weight of words in documents. In which, Term Frequency (TF) means the frequency a word appears in a document and Inverse Document Frequency (IDF) means the reciprocal of document frequency. The specific definition of TF-IDF is as follow:

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}} \quad (1)$$

$$idf_i = log(\frac{|D|}{|j : t_i \in d_j|}) \tag{2}$$

$$tfidf_{i,j} = tf_{i,j} \times idf_j \tag{3}$$

Where $n_{i,j}$ means how many times a word i is used in document j, $|D|$means the number of all the documents and $|j : t_i \in d_j|$ means how many documents are consist of word i. For word i in document j, we synthesize the TF and IDF together to evaluate the weight of a word.

But in requirement acquisition, we not only simply distinguish the documents but also judge a document whether is about feature requests or not. In the experiments we find that there are some specific words that are used to depict feature requests with high probability, such as “add”, which is often used to express that someone wishes the software to add a specific feature or “allow” may mean that the software should allow users to do something. The regulation is obvious and C. Iacob and R. Harrison [3] also used some specific linguistic rules, which users often use, to find the feature requests from user comments. But these key words are usually used in many documents but may appear only once in a document, so the TF-IDF value may be very low, but we think these words can depict a document about feature requests well. So we borrow the idea from MARA [3] and build a requirement dictionary, which contains the words that are usually used to express feature requests. The partial key words which are often used to represent feature requests are listed in Table 1.

Table 1. Keywords about Feature Requests

add, allow, complaint, could, hope, improvement, instead, lacks, maybe, missing, must, need, please, prefer, request, should, suggest, want, will, wish, would, tag, allow, hope, improve, want, auto, let, filter...

We want to increase the weight of these key words because they can represent the documents we want well. In our approach, we simply duplicate these words in a document three times. For example, if “add” is used in a document, we extend the frequency of the word three times and there will be three “add” in the document. After the process, the IDF value will not change but the TF value can increase obviously.

3.4 Feature Acquisition Based on Linguistic Rules

To compare with the SVM approach, we implement a prototype based on MARA (Mobile App Review Analyzer) [3] that was proposed to retrieve and analyze mobile app feature requests from online reviews through a set of linguistic rules. We want to know whether the approach based on linguistic rules can fit the problem of our experiment or not. In our experiment, we choose 200 questions which are tagged "feature-request" by users to define a set of linguistic rules, and some illustrative examples are depicted in Table 2. In our experiment of linguistic rules, we only consider the titles of the questions because of the complexity of the bodies of the questions. After defining the linguistic rules which are usually used to express feature requests, we use these linguistic rules to judge whether the new question is about feature requests or not. The manual defining of the linguistic rules is time consuming and labor intensive. At last we define a set of 32 linguistic rules which are usually used to express feature requests.

Table 2. Linguistic Rules Elicited from 200 Training Examples

Linguistic rule	**Example contexts**
add/change ... to ...	add ability to cancel flags; change color of inline links in SO;
let's/let us ...	Let us bring an end to the "robo-reviewer" war: Phase 1-2; Let's improve the How to Ask page(s);
filter/improve ...	Filter 10k tools by a tag; Improve the "404 Not Found" error page for deleted questions;
can/should we ...	Can we have a delete review queue?; Should we be able to flag suggested edits?
sth. should ...	"Similar Questions" search should take the tags into account; Room owners should be allowed to accept ¡20 rep users to talk in a room;
don't ...	Don't tell me to review a declined flag on a deleted post; Don't allow suggested edits to be "finished" while someone has clicked "improve";

4 Experiment Setting

4.1 Dataset

To support our experiment, we dumped the dataset from meta.stackoverflow.com which is a user feedback site for Stack Overflow. We imported the XML format dataset to the database, and we analyze it and get some valuable things. The detail is showed in Table 3. As the table depicts, in the dataset, there are 5770 questions. In the whole questions, there are 909 questions being tagged "feature-request" and 4861 questions being without "feature-request" tag. The time period of the dataset is from 2008.11.26 to 2014.9.14. But we are surprised that the most of questions are posted in 2014, so we boldly believe that people are involving in the social Q&A sites quickly now.

Even though there are 909 questions being tagged "feature-request" in all 5770 questions, we still want to explore how many questions are about feature requests in the whole questions without "feature-request" tag. We randomly choose 500 questions from the untagged questions and tag the questions manually. To eliminate the divergence of personality and make the tag process incorrect, we apply a method that two authors tag the questions separately and then

Table 3. Statistics of Experiment Dataset

#questions	#"feature-request"	#no "feature-request"	time period
5770	909	4861	2008.11.26-2014.9.14

filter the questions that were tagged the same tags by both authors. Because both authors are not professional developers for Stack overflow, it is not absolutely correct in tagging questions, but the obvious questions can be tagged correctly.

In the process of tagging, we find that many questions are about features, bugs and even complaints. The result of tagging is showed in Table 4, and we can find that many questions without "feature-request" tags are also about feature requests. In all 500 questions, author 1 and author 5 labeled 181 and 121 questions about feature requests separately, the other 319 and 379 questions are not about feature requests. We integrate the result of two authors do and get 106 questions which both regard as feature requests and 304 questions which are not regarded as questions about feature requests by both authors. We spent much time doing the process of label questions manually, and it is absolutely hard to find the feature requests from the useless questions. So it is significant to propose an approach to tag the questions automatically, and our work is valuable and will assistant the developers to find the feature requests from large numbers of questions in social Q&A sites.

Table 4. Result of Tagging Questions Manually

	#questions	#about feature	#about non-feature
author1	500	181	319
author5	500	121	379
author1&author5	410	106	304

4.2 Performance Metrics

In this section, we focus on the performance metrics that will be used in our evaluation. We consider precision, recall and F-measure to evaluate the result of experiments. Precision is the ratio between returned results that are correct and the total number of returned results. Recall is the ratio between the returned results that are actual feature requests and the total number of feature requests in the input. F-measure considers both the precision and the recall, and can be interpreted as a weighted average of the precision and recall. F-measure is defined as follows in detail:

$$F = 2 \times \frac{precision \times recall}{precision + recall} \tag{4}$$

The precision, recall and F-measure can depict the performance of one approach for requirement acquisition well. For precision, recall and F-measure, the values of them are the bigger the better.

5 Results and Discussion

In this section, we introduce the experiment results in detail, comparing the performance of the approach based on linguistic rules, the approach based on pure SVM and the approach based on SVM with requirement dictionary. Then we consider the scale of testing data and test the performance in different approaches with the increasing numbers of testing data.

5.1 Comparison Among Three Approaches

In this experiment, we also choose the same 200 questions tagged "feature-request" by users and 200 questions which are labeled as "non-feature" by both two authors manually, and the whole 400 questions will be used as training examples to train the SVM model. The only difference of the two SVM experiments is whether considering the requirement dictionary about requirement acquisition. In SVM with requirement dictionary, we use the dictionary to correct the TF-IDF values of each word in documents. It is different from the previous approach based on linguistic rules, because we not only consider the titles but also the bodies of questions to train the SVM model. Our work is focusing on finding the questions about feature requests automatically and accurately.

Table 5. Result of the Experiments

	linguistic rules	**pure SVM**	**SVM with requirement dictionary**
Precision	61%	68.8%	72%
Recall	55%	75%	77%
F-measure	57.8%	71.77%	74.42%

The results of experiment is showed in Table 5 in detail. Focusing on 200 questions as testing examples, which contain 100 questions about feature requests and 100 questions are not feature requests. The results of different approaches are different, and the precision of the linguistic approach is 61%, but it can achieve 68.8% and 72% for pure SVM and SVM with requirement dictionary. The recalls among them are 55%, 75% and 77%. The F-measures for three different approaches are 57.8%, 71.77% and 74.42%. From the results we can see that the SVM approach can achieve better performance than the approach based on linguistic rules, and the improved SVM with requirement dictionary can also improve the performance of feature acquisition. The results fit our expectation. The approach based on linguistic rules is easy to understand. But depended on limited training data, it is hard to define enough linguistic rules that can fit the texts posted by users. Because users express their requirements in different types, and the process of defining linguistic rules manually is hard and time consuming. As Table 5 shows, the approach based on SVM is effective, SVM can depend on the words of every text and automatically construct a classifier based on the limited training data, but we can find that there is not much difference in the performance between pure SVM and SVM with requirement dictionary. The result is because the dictionary we build has only 74 words, it is obviously not enough to sum up the words usually used to express requirements.

5.2 The Result in Different Scales of Testing Data

Using the three different approaches in requirement acquisition, we want to know the performance with the increasing scales of testing data. Here we only consider the recall of different approaches because of the hard work in tagging questions, it also means that how many questions about feature requests can be extracted through different approaches. We separately choose 200, 300, 400, 500, 600 and 700 questions which are tagged "feature-request" by users of meta.stackoverflow.com, and test the data through 32 linguistic rules, pure SVM model and SVM model with requirement dictionary. The results is showed in Fig. 2.

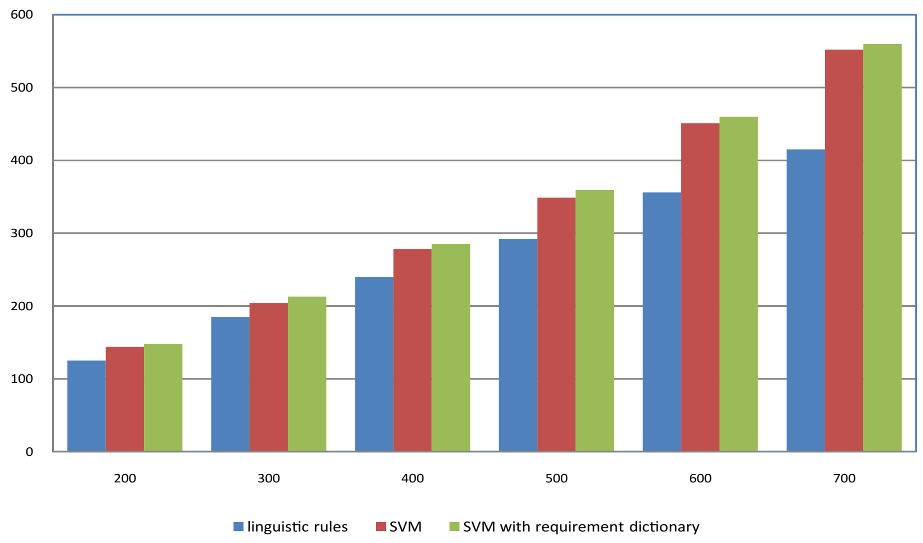

Fig. 2. Result in different scales of testing data

As Fig. 2 shows, with the increasing number of testing data, the number of questions which are extracted by three different approaches are also increasing. And we can find that the performance among three approaches remains the same trend. The performance of the approach based on linguistic rules decreases with the increasing numbers of testing data obviously, because the small number of linguistic rules can't match the more and more users' expressions.

6 Threats to Validity

The experiment results show that SVM with requirement dictionary can achieve a better result than approach based on linguistic rules and pure SVM, but there are still some threats. Firstly, the authors of this paper are not professional software developers, the manual tagged questions may be not always true for real developers. Secondly, in our experiment we choose the questions tagged "feature-request" by users as the training examples about feature requests, and this is based on the hypothesis that the users post their feature requests accurately. At last, in our experiment we increase the weight of words in documents through three times its number of occurrences, but this strategy is not perfect. So there is still a lot of work needs to be done for a better approach to mine the feature requests for assisting the software developers.

7 Conclusion

User feedback is important for software development. With the development of Internet communities, users can express their opinions through many ways, such as forums, mail-list, and software marketplace. These years have focused on the development of Stack Overflow that is a famous programming Q&A sites, and the social Q&A has become more and more popular because of its better environment. Based on the new way to gather users' feedbacks, more and more Q&A sites have been established, there are also some Q&A sites for software. But the social Q&A sites are open and free, a lot of information is invaluable, so to find a method to extract the valuable information is important. In this paper, we propose an approach based on SVM with requirement dictionary about feature acquisition. Comparing the new approach with pure SVM and the approach based on linguistic rules, we can find the approach based on SVM with requirement dictionary about feature acquisition is better than pure SVM and linguistic method. In the future, we hope to find the latent topics about requirements from users. We hope to establish a better approach to extract the most important information for continuous software development.

Acknowledgments. In this paper, the research was sponsored by the National Natural Science Foundation of China (Grant No.61432020 and No.61472430).

References

1. Vasilescu, B., Serebrenik, A., Devanbu, P., et al.: How social Q&A sites are changing knowledge sharing in open source software communities. In: Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing, pp. 342–354. ACM (2014)
2. http://stackexchange.com/tour
3. Iacob, C., Harrison, R.: Retrieving and analyzing mobile apps feature requests from online reviews. In: Proceedings of the 10th Working Conference on Mining Software Repositories, pp. 41–44 (2013)

4. Harman, M., Jia, Y., Zhang, Y.: App store mining and analysis: Msr for app stores. In: Proceedings of the 9th Working Conference on Mining Software Repositories, pp. 108–111 (2012)
5. Galvis Carreño, L.V., Winbladh, K.: Analysis of user comments: an approach for software requirements evolution. In: Proceedings of the 35th International Conference on Software Engineering, pp. 582–591 (2013)
6. Chen, N., Lin, J., Hoi, S. C., Xiao, X., Zhang, B.: AR-miner: mining informative reviews for developers from mobile app marketplace. In: Proceedings of the 36th International Conference on Software Engineering, Hyderabad, India, 31 May–07June 2014
7. https://en.wikipedia.org/wiki/Supervised_learning
8. LIBSVM: a library for support vector machines
9. http://www.csie.ntu.edu.tw/cjlin/libsvm/
10. Noll, J.: Requirements acquisition in open source development: Firefox 2.0. Open Source Development, Communities and Quality, pp. 69–79. Springer, US (2008)
11. Bajaj K, Pattabiraman K, Mesbah A.: Mining questions asked by web developers. In: Proceedings of the 11th Working Conference on Mining Software Repositories. ACM, pp. 112–121 (2014)
12. Somprasertsri, G., Lalitrojwong, P.: Mining Feature-Opinion in Online Customer Reviews for Opinion Summarization. J. UCS **16**(6), 938–955 (2010)
13. Cleland-Huang, J., Dumitru, H., Duan, C., et al.: Automated support for managing feature requests in open forums. Communications of the ACM **52**(10), 68–74 (2009)
14. Hu, M., Liu, B.: Mining and summarizing customer reviews. In: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 168–177. ACM (2004)