An Empirical Study on Classification of Non-Functional Requirements

Wen Zhang, Ye Yang, Qing Wang, Fengdi Shu
Institute of Software, Chinese Academy of Sciences
Beijing 100190, P.R.China
{zhangwen,ye,wq, fdshu}@itechs.iscas.ac.cn

Abstract—The classification of NFRs brings about the benefits that NFRs with respect to the same type in the system can be considered and implemented aggregates by developers, and as a result be verified by quality assurance assigned for the type. This paper conducts an empirical study on using text mining techniques to classify NFRs automatically. Three kinds of index terms, which are at different levels of linguistics semantics, as N-grams, individual words, and multi-word expressions (MWE), are used in representation of NFRs. Then, SVM (Support Vector Machine) with linear kernel is used as the classifier. We collected a data set from PROMISE web site for experimentation in this empirical study. The experiments show that index term as individual words with Boolean weighting outperforms the other two index terms. When MWEs are used to enhance representation of individual words, there is no significant improvement on classification performance. Automatic classification produces better performance on categories of large sizes than that on categories of small sizes. It can be drawn from the experimental results that for automatic classification of NFRs, individual words are the best index terms in text representation of short NFRs’ description and we should collect as many as possible NFRs of software system.

Keywords—Non-Functional Requirements, Automatic Classification, Support Vector Machine.

I. INTRODUCTION

Non-functional requirements (NFRs) describe the expected qualities of a software system such as usability, security and look and feel of user interface. These qualities do impact on the architectural design of the software system [1] and satisfaction of stakeholders relevant with the software system [2] [3] [4]. NFRs are put forward by stakeholders with no expertise in software engineering and the number of NFRs is often very large. Moreover, NFRs are scattered over the functional requirements and the types of the NFRs are unknown because they are written in natural language. These characteristics of NFRs make its classification a labor-intensive and time-consuming job without the adoption of automatic techniques. Nevertheless, software developers cannot ignore NFRs because they are decisive on the success of the software system.

Compared with functional requirement, NFRs tend to be properties of a system as whole, and hence cannot be verified for individual components or modules [19] [23]. This makes the classification of NFRs necessary in development because, on the one hand, we need to handle NFRs in a different way from functional requirements and, different types of NFRs should be handled by developers with different expertise.

For instance, in a security critical, mission critical, or economy vital software systems, formal method with model checking of the correctness of specification on system security is used to determine whether or not the ongoing behavior holds of the system’s structure. However, a serious (and obvious) drawback of model checking is the state explosion problem because the size of the global state graph is (at least) exponential in the size of the program text [21]. Thus, an exigent demand from system developers is to identify the security-related NFRs from the requirement document and express them using formal specifications, and then to conduct verification of the satisfaction of system’s properties on those security specifications.

On the other hand, the classification of NFRs benefits an overall decision on the systems’ satisfaction on each category of NFRs and the progress of system development developers have made. Functional requirements can be measured as either satisfied or not satisfied, but NFRs cannot merely measured by a linear scale as degree of satisfaction [25]. In system testing, for example, we can customize our test strategy based on the classification of NFRs and thus system behaviors of NFRs were reported directly to the project manager.

Most, if not all, projects have resource limitations and time constraints, with different requirements having different concerns of software systems. Even after the requirements are elicited and collected, they are still just an unorganized set of data. As such, it is necessary to categorize the requirements into different types so as to match problems and solutions in separate domain of discourse, especially for large-scale software projects with hundreds or even thousands of requirements.

In human resource allocation and optimization [22], different developers possess different expertise in handling various aspects of software development. Different tasks in development may need different expertise and capability from the developers. Thus, a match of developers and tasks is at the core of the success of software development. The identification of different types of NFRs results in the formation of different types of development tasks. Consequently, those tasks are assigned to developers aggregates according to their expertise and capability level. For instance, the NFRs with “performance” type and the NFRs with “maintainability” type should be dealt with by developers with different expertise.
We forward NFRs of the type "look and feel" to UI (User Interface) design experts of the system so that the satisfaction of these NFRs can be ensured throughout the whole system.

Methods for the elicitation of NFRs include questionnaires, checklists and templates for inquiring stakeholders concerning quality issues [24]. Basically, NFRs are made of textual sentences whose contents concern the expected qualities of a software system. For instance, "The application shall match the color of the schema set forth by Department of Homeland Security. LF" is a typical NFR record fetched out from the data set and it comprises two parts: textual description and NFR type as "LF". The description of NFR is very simple and easy to understand. It does not need professional knowledge of engineering aspects. Thus, text mining, which combines both natural language processing (NLP) and statistical machine learning, can be used for the task of automatic classification of NFRs.

Although there is substantial literature on NFR classification [2] [4] [5] [17], we find two problems. First, most methods are purely intuitive and derived without theoretical support or mathematical model. Also, some methods are extremely labor-intensive and time-consuming, and others are qualitative, without quantitative analysis. Second, the index terms used in existing automatic classification of NFRs are keywords extracted directly from requirements without feature selection. This would cause huge dimensionality of NFR vectors and consequently bring about huge computation when quantitative methods are used. Most importance, we are uncertain that other index terms than keywords are more appropriate for automatic classification of NFRs.

The questions devised for this empirical study are: 1) among existing indexing methods, which one is the best performance for automatic classification of NFRs, and 2) is it possible to produce better performance with SVM than those derived in previous work?

The remainder of this paper is organized as follows. Section 2 describes the research approach employed in this paper. Section 3 conducts experiments of using SVM and different index terms to classify NFRs. Section 4 present the related work and Section 5 concludes this paper.

II. RESEARCH APPROACH

The research approach adopted in the empirical study is shown in Figure 1 to automate the classification of NFRs. First, NFRs' textual description are represented using different types of index terms as N-grams, individual words, and MWEs, respectively. Then, we transfer NFR textual description into numerical vectors in different feature space determined by different types of index terms. Second, support vector machine (SVM), which is a popular classifier in machine learning [6] [9], is used to classify NFR vectors. Finally, performance of each classification is evaluated.

A. Index Terms

The requirement data set we collected from the PROMISE web site (http://promisedata.org/repository) and it contains 625 records of both functional and non-functional requirements.

N-gram Representation. N-gram is proposed in text mining to categorize documents with textual errors such as spelling and grammatical errors, and it has been proved as an effective technique to handle these kinds of errors [7]. An N-gram is an N-character contiguous fragment of a long string. Since every string is decomposed into small fragments, any errors that are present in words only affect a limited number of those fragments. If we measure the similarity of two strings based on their N-grams, we will find that their similarity is immune to most textual errors. We used bi-gram (344 2-grams) and tri-gram (1,295 3-grams) for representation of NFRs, respectively.

Word Representation. The method of using individual words of a text content to represent the text can be traced to Salton et al [10]. We follow this idea to use words in NFRs for representation. First, we eliminated stop words from NFRs' description 1. Second, word stemming 2 was conducted to map word variants to the same stem. We set the minimum length of a stem as 2. That is, only the stems which have more than 2 characters will be accepted as stems of words. Thus, 1,127 individual word stems are produced from the data set.

MWE Representation. We used the method proposed by Justeson and Katz [12] for MWE extraction from NFRs. The basic idea behind this method is that a MWE should include 2 to 6 individual words and it should occur in a collection of documents more than twice. Nevertheless, the part of speeches of individual words of a MWE should meet the regular expression described in formula 1.

\[
((A \mid N)^\ast \mid (A \mid N)^\ast (NP)^\ast (A \mid N)^\ast )N
\]  

(1)

Here, A denotes an adjective, N denotes a noun and P denotes a preposition. Using the method adopted from [11], we extracted 93 MWEs from the data set.

1We obtain the stop words from http://ftp.uspto.gov/patft/help/stopword.htm
2we use Porter stemming algorithm that is available at http://tartarus.org/martin/PorterStemmer/
B. Feature Selection

In this paper, we adopt information gain (IG) [8], which is a classic method for feature selection in machine learning, to select informative index terms for automatic classification of NFRs. IG is defined as the expected reduction in entropy caused by partitioning NFRs according to a given term. The formula of IG is presented in Equation 2 and the formula of entropy is depicted in Equation 3.

\[
Gain(S, A) = Entropy(S) - \sum_{v \in Value(A)} \left( \frac{|S_v|}{|S|} \cdot Entropy(S_v) \right)
\]

(2)

\[
Entropy(S) = \sum_{i=1}^{c} -p_i \log_2 p_i
\]

(3)

Here, \( S \) is the collection of the types of all NFRs such as PE (performance) and US (usability). \( Value(A) \) is a set of all possible values of index term \( A \). \( S_v \) is a subset for which \( A \) value \( v \), \( c \) is the number of categories of all NFRs, and \( p_i \) is the proportion of the NFRs that belong to category \( i \). To observe the performances of textual features on automatic classification dynamically, the different feature sets of textual features are constructed at different removal percentages of low IG value terms (see also removal ratio in Section 3.2).

C. SVM Classifier

The classifier used for automatic classification is support vector machine (SVM) [6] [9], that is introduced in statistical machine learning. We selected this method in this paper because Gokyer et al [9] used it to transfer NFRs to architectural concerns and preferred effectiveness was achieved. In this paper, the linear kernel \((u*v)\) is used for SVM training because it is superior to non-linear kernels for classifying textual contents validated by prior research [8] [13].

B. Classification Process

There are actually two kinds of work involved in representing textual contents: indexing and term weighting. Indexing is the job of assigning index terms for textual contents and term weighting is job of assigning weights to terms, to measure the importance of index terms in textual documents. We employ Boolean values as term weights for the index terms for the reason that most NFRs in the data set are very short and contain less than 20 index terms so they do not need complex weighting schemes such as those mentioned for document representation [15].

IG is employed to change the percentages of index terms used for representation. The removal ratio is predefined to remove the index terms with small entropy. For instance, if we set the removal ratio to 0.1 for representation with individual words, then a percentage of 90% of individual words with smaller entropy will be eliminated from the feature set and we only use the remaining 10% of individual words for the representation of NFRs. The purpose of varying different percentages of index terms for representation is that we want to observe the robustness of classification performances when the set of index terms become smaller and smaller. This is especially important for deciding which type of index terms should be used for representing NFRs when computation capacity is not enough to support large dimension of vectors in training classifier. In representation with MWEs, we use all the 1,127 individual words and a proportion, which is defined by the removal ratio, of MWEs as the index terms.

The experiments in this paper are carried out with 10-fold cross-validation technique. In each experiment, we divide the whole data set (for both positive and negative classes) into 10 subsets. The 9 of 10 subsets are used for training and the remaining one subset is used for testing. We repeat the experiment 10 times and the performance of the classification is measured as average precision and recall [2] of the 10 repetitions.
C. Experimental Results

The outcome of our experiments shows that the precisions of all the automatic classification tasks are significantly higher than those of the classification approach proposed by Cleland-Huang et al [2]. Yet, the recalls of the automatic classifications in this paper are not comparable to the classification proposed by Cleland-Huang et al. We do not list the precisions and recalls produced in our experiments due to space limitation. (Readers who have an interest in the precisions and recalls are welcome to ask the authors for more details).

We conjecture that the high recalls of the results of Cleland-Huang et al [2] can be attributed to the small size (within the range between 10 and 20) of index terms they employed in their experiments, which is much smaller than the sizes of index terms (within the range between 100 and 1,000) used in our experiments. When small size of index terms is used, larger number of NFRs is classified as relevant NFRs of the category. Consequently, more irrelevant NFRs will be misclassified as relevant. Considering an extreme case of classifying relevant NFRs with "Usability", if all the NFRs are regarded as relevant with "Usability", then the recall of automatic classification will be 100%. However, this classification result may be of less help for automating the task of classification. Thus, we argue that for automatic classification of NFRs, precision should be given more importance if recall is acceptable.

The F-measure [16] described in Equation 4 combines both precision and recall for performance evaluation. We used F-measure as the indicator for performance evaluation. In general, the larger the F-measure is, the better the classification result is. Here, for the purpose of comparison, we make out the F-measures of Cleland-Huang et al [2] as the baseline performance in the experiments.

\[
F\text{-}measure = \frac{2 \times precision \times recall}{precision + recall} \tag{4}
\]

Figures 2-4 show the experimental results of classifying the assigned three categories "Usability", "Security", and "Look And Feel" at different removal ratios using four types of index terms (2-gram, 3-gram, word and MWE) to represent NFRs.

First, it can be seen that the representation with individual words has the best performance measured by F-measure nearly on all three categories. In some cases, representation with terminologies improves the precision of automatic classification of representation with individual words (when the removal ratio arrives at 0.9). That is, even if most index terms are removed from the feature set, their performances do not decline drastically. However, in most cases, representation with terminologies is not able to improve the performances of automatic NFR classification compared with representation with individual words, even worse than that of 3-gram representation. Both representation with individual words and terminologies produce very robust classification.

This outcome implies that for automatic classification of NFRs, representation with individual words is sufficient to produce a desirable performance. This outcome is very different from the experimental results produced by Zhang et al.
They argued that MWEs are superior to individual words for classifying news documents automatically. We explain that the lengths of NFRs in the data set are much shorter (20 individual words on average) than those of Reuters-21578 news documents (80 individual words on average) so the semantics inherent in textual contents of NFRs is not so much important as those inherent in news documents.

Second, for representation with N-grams, representation with 3-grams outperforms that with 2-grams. This outcome can be attributed to that the number of 3-grams is much larger than that of 2-grams (see Section 2.1). Moreover, the robustness of the 3-gram representation is better than the 2-gram representation because the magnitudes of variations of F-measures of the 3-gram representation are smaller than those produced by the 2-gram representation.

Thirdly, for classification on different categories, the performance on "Security" is as almost the same as that on "Usability" but outperforms that on "Look And Feel". The number of NFRs in category "Security" (66) is approximately equal to that in category "Usability" (67), which is much larger than the number of "Look And Feel" (38). We explain that automatic classification would produce more favorable performance on categories those having large sizes than those having small sizes.

Based on the experimental results, the answer for question 1 devised for this case study in Section 1 should be that to date, keywords are the most effective index terms for automatic classification of NFRs and for question 2, our answer is that the machine learning technique, at least SVM, can improve the classification significantly.

IV. RELATED WORK

NFR has attracted much interest of researchers from software engineering field. Much work has been invested in managing NFRs. Tran and Chung [20] proposed a prototype tool for explicit representation of NFRs. They aim to consider NFRs in a more goal-oriented perspective and argue that NFRs have much influence on the design of the solution as well as reinforce engineering process. In order to formulate implicit relationships of NFRs, ontology and graphic visualization are used in their tool to express softgoals and their interdependencies explicitly.

Cleland-Huang et al [4] introduce a goal-centric approach to managing the impact of change upon the NFRs of a software system. They partition NFRs into different softgoals of a system and construct a softgoal interdependency graph (SIG) to trace both direct and indirect impacts of software changes on NFRs. Probabilistic network model is employed to enable the traceability of impacts. However, the job of constructing SIG is labor-intensive and time-consuming because of the lack of automatic approaches. If we can classify NFRs into different softgoals automatically, it would be much easier to identify the relations between subgoals and softgoals. That is, human workload involved in constructing SIG will be reduced to a great extent.

In another work, Cleland-Huang, et al.[2] proposed an information retrieval method to discover and identify NFRs from system specification. Their basic assumption is that different types of NFRs are characterized by distinct keywords (index terms) that can be learned from documents of that type. However, their method of selecting index terms is ad-hoc and does not consider any linguistic properties of those index terms. Moreover, the classifier used in their method, which is based on the additive weights of index terms on a given NFR type, is quite simple without any theoretical ground.

Rosenhainer [14] proposed aspect mining to identify cross-cutting concerns in requirements specifications. Two techniques are suggested to be used for aspect mining: identification through inspection and identification supported by information retrieval (IR). The former one is manual and the later one is semi-automatic. Rosenhainer argued that IR-based technique is more promising than manual method and their experimental results on interactions between functional and non-functional requirements have validated this argument. This work encourages our study in this paper to use IR techniques for automatic classification of NFRs.

Casamayor et al [24] employed naïve Bayes and EM (Expectation Maximization) algorithm as a semi-supervised learning approach to classify non-functional requirements in textual specifications. They used the same data set as used in this paper and reported that their algorithm produced an average accuracy above 70%. In fact, the performance of SVM in our experiments is better than theirs because we exclude those categories of small number of data points. That is, unbalanced distribution of data points is purposely alleviated in our experiments. Moreover, our conjecture is also validated by their experimental results that those category of large number of data points such as "usability", "security" and "Look And Feel".

V. CONCLUDING REMARKS

NFR is crucial to the success of a software system as it describes necessary qualities of system to avoid devastating effects and system failure [17]. In this empirical study, vector space model and machine learning technique are employed to classify NFRs automatically. We used different index terms to transfer NFRs into numeric vectors and examined their performances on automatic classification of NFRs. The machine learning classifier we adopted in this paper is SVM with linear kernel, which is widely recommended as a promising classifier for text mining. Information gain for feature selection and SVM for automatic classification is introduced. We conducted experiments using the data set collected from PROMISE data set.

The experimental results show that individual words, when used as the index terms, have the best performance in classifying NFRs automatically. We noticed that, in most cases, the more samples in a category in data set, the better performance the automatic classifier will produce on the category. This outcome illustrated that the number of NFRs in the data set is an important factor for automating the classification of NFRs.
This inference suggests that we need to collect NFRs as many as possible if automatic classification is desired. This work can be applied in at least two aspects in software engineering currently. First, it can be used to identify NFRs in requirement specification. Usually, customers, testers, and stakeholders of a software system will mix their desirable qualities in a specification. Whereas functional requirements describe what the system needs to do, NFRs describe constraints on the solution space and capture a broad spectrum of properties, such as usability and security [18]. Because the solutions of functional requirements and NFRs are different, or the time to consider these two kinds of requirement in system design is different, we must differentiate these two kinds of requirements. Second, different NFRs are often handled by different designers and developers with different background knowledge in architecture, it is crucial to classify the NFRs into different categories so that the NFRs in the same category can be processed as a comprehensive requirement in system design.

ACKNOWLEDGMENTS

This work is supported by the National Natural Science Foundation of China under Grant Nos. 90718042, 60873072, 61073044, and 60903050; the National Science and Technology Major Project; the National Basic Research Program under Grant No. 2007CB310802; the Scientific Research Foundation for the Returned Overseas Chinese Scholars, State Education Ministry.

REFERENCES