Automatically Classifying Non-functional Requirements with Feature Extraction and Supervised Machine Learning Techniques: A Research Preview

Mahtab EzzatiKarami(✉) and Nazim H. Madhavji

The University of Western Ontario, London, ON N65B7, Canada
mezzati@uwo.ca

Abstract. Context and Motivation: In large projects, extracting the relevant NFR-information as per the stakeholder's responsibility and needs can be time-consuming and challenging. **Question/Problem:** Classification of NFRs is one way to mitigate this problem. However, because of the size and complexity of the SRS, the manual classification of NFRs is considered time-consuming, labour-intensive, and error-prone. An automated solution is needed that provides a reliable and efficient classification of NFRs. **Principal ideas/results:** Using natural language processing and supervised machine learning (SML) algorithms, we investigate feature extraction techniques (i.e., POS-tagging based, BoW, and TF-IDF) to assess their efficacy in automated classification, in conjunction with the SML algorithms (such as: SVM, SGD SVM, LR, DT, Bagging DT, Extra Tree, RF, GNB, MNB, and BNB). **Contribution:** The proposed combinations: (i) SVM with TF-IDF, (ii) LR with POS and BoW, and (iii) MNB with BoW, all achieve precision and recall values greater than 0.85, and process execution time of less than 0.1 s. Comparison with related work is favourable as is preliminary validation using an industry dataset.

Keywords: Non-functional requirements · Classification · Supervised Machine Learning · Feature extraction

1 Introduction

Non-functional requirements (NFRs) describe desirable quality attributes (e.g., performance, reliability, and availability) of a software system. In a software requirements specification (SRS) document, the functional requirements and NFRs are often mixed together (perhaps categorised under domain or application-specific headers). In large projects, understanding and extracting the relevant NFR-information as per the stakeholder's responsibility and needs can be time-consuming and challenging, due to the size, complexity, and lack of familiarity with the SRS.

For example, in one large project [1], there were approx. 600 regulatory requirements buried amongst 12,000 requirements spread over a thousand pages of a contract. In such

F. Dalpiaz and P. Spoletini (Eds.): REFSQ 2021, LNCS 12685, pp. 71–78, 2021.
https://doi.org/10.1007/978-3-030-73128-1_5

cases, manually identifying the needed NFRs for project-work is effortful and error-prone [1]. Other researchers [2] have also shown that classifying NFRs into different types can aid stakeholders' project concerns [2].

In our research, we investigated how accurately we can classify NFRs automatically into various types; in particular, *usability*, *security*, *performance*, and *operational* requirements – which are the top four NFR-types in the PROMISE dataset. The goal of our research is to provide stakeholders with a reliable and efficient solution for extracting specific NFRs from SRS documents using natural language processing techniques combined with supervised machine learning (SML) algorithms. We investigate feature extraction techniques (i.e., POS-tagging based, BoW, and TF-IDF) to assess their efficacy in automated classification, in conjunction with the SML algorithms (such as: SVM, SGD SVM, LR, DT, Bagging DT, Extra Tree, RF, GNB, MNB, and BNB).

The resultant combinations: (i) SVM with TF-IDF, (ii) LR with POS and BoW, and (iii) MNB with BoW, all achieve precision and recall values greater than 0.85 and process execution times of less than 0.1 s; meaning, the right NFRs would be rendered, and quickly, to the stakeholder concerned. Comparison with related work [2, 4] is favourable as is the classification of NFRs in an industry dataset.

2 Related Work

We describe and analyse several previous attempts at classifying NFRs using machine learning techniques. In 2006, Cleland-Huang et al. [2] classified 370 NFRs from 15 SRS documents developed by graduate students. Their used certain keywords as indicator terms to distinguish different types of NFRs, trained these, and then classified NFRs from given documents according to the occurrence of the indicator terms. Their model yielded a classification recall of approx. 0.8 (considered high) and precision of 0.12 (low) due to a high rate of false positives.

In 2011, Zhang et al. [3] conducted an empirical study on classifying NFRs using SVM and the PROMISE dataset. Three kinds of index terms, at different levels of linguistic semantics (N-grams, individual words, and multi-word expressions (MWE)) were used in the representation of NFRs. Their results show that index terms as individual words with Boolean weighting outperform N-grams and MWEs; and using MWEs does not enhance the representation of individual words significantly. Also, they observed that automatic classification results in better performance on categories of large sizes than of small sizes. They conclude that individual words are the best index terms in text representation of short NFR descriptions. In comparison to the study by Cleland-Huang et al. [2], they report higher precision but lower recall values.

In 2017, Lu et al. [4] classified app user reviews into four types of NFRs (i.e., reliability, usability, portability, and performance), functional requirements, and others. Their approach combines four classification techniques (BOW, TF-IDF, CHI2 and AUR-BOW) with three machine learning algorithms (Naïve Bayes (NB), J48, and Bagging) to classify the reviews. The combination of AUR-BOW with Bagging achieved the best result (precision: 0.71 and recall: 0.72). An interesting observation made was that automatic classification using an imbalanced dataset performs poorly when the numeracy of certain types of NFRs is low.

2.1 Analysis

Some opportunities motivated us to conduct our research:

- Most of the related works have used the PROMISE dataset, which has at least two issues: (i) it is imbalanced, and (ii) the number of NFRs is not numerous (370 NFRs). For our investigation, we combined two datasets (PROMISE and PURE (with 160 NFRs)) totalling more than 500 NFRs. In addition, we separately apply (or preliminarily validate) our techniques on an industry dataset (of approx. 260 NFRs).
- We cover a number of techniques in combinations: three feature extraction techniques (BOW, TF-IDF, and POS-tagging) and 10 supervised machine learning techniques (SVM, SGD SVM, DT, Extra Tree, Bagging Tree, LR, RF, GNB, BNB, and MNB).

3 Research Investigation

As mentioned in Sect. 1, our investigation focuses on four NFR types: usability, performance, security, and operational NFRs. The key research question we ask is:

Which combination of feature extraction techniques (BOW, TF-IDF, and POS-tagging) *and SML algorithms* (SVM, SGD SVM, DT, Extra Tree, Bagging Tree, LR, RF, GNB, BNB, and MNB) *yields the highest precision and recall values*?

The higher the precision value, more the proportion of the identified NFRs that are relevant to the inquiring stakeholder. The higher the recall value, less the relevant NFRs that are missed (or not identified) from the dataset.

In a practical setting, the best combination of feature extraction technique and the SML algorithm could be a basis for a real-time interactive tool that serves the needs of various kinds of stakeholders: analysts, architects, programmers, testers, product managers, domain experts and more. They all need to know the details of the NFRs relevant to their concern, in a given project, from time to time. For example: What are the reliability needs of the system and have I addressed them all in the design of the system's architecture? Is my choice of algorithms to code appropriate for the performance requirements of this device? Does the envisaged core quality of the system give us a competitive advantage?

3.1 Datasets

We use two datasets: NFR PROMISE dataset and the PURE dataset[1]. The PROMISE dataset consists of 625 requirements (255 functional, 370 NFRs). There are 11 groups of NFRs but because the number of instances of a few types is low, we chose the top four types for classification: usability (62), performance (48), security (58), and operational (61). We also used the PURE dataset (which focuses on public requirements documents). It consists of 296 requirements (136 functional and 160 NFR). There are 16 groups of NFRs in this dataset and the top three types are usability (54), performance (18), and security (17).

[1] https://zenodo.org/record/1414117#.X7wfOWhKiUk.

We combine these two datasets in a sequence to form the training set. The effect on precision and recall of mixing the two datasets differently (e.g., interleaved NFRs) is not empirically tested though it seems that the SML algorithms are agnostic about the ordering of the NFRs in the dataset.

Given a dataset of all requirements, we first identify functional requirements and, separately, NFRs. For this, we use the distinguishing criteria from the established literature [5]. We then further classify the NFRs into the four (for PROMISE) and three (for PURE) predominant types. By stripping off extraneous metadata, we then convert the datasets into standard CSV files. They include the columns: "Requirement Description", "F/NF" for functional or NFR, and "Subtype" of NFRs. Table 1 depicts example NFRs from the PROMISE and PURE datasets.

Table 1. Example NFRs from the PROMISE and PURE datasets

Label	Requirement text	Dataset
Availability	"Aside from server failure the software product shall achieve 99.9% up time"	PROMISE
Safety	"The system will do periodic backups through a live internet connection"	PURE

3.2 Research Methodology

We want to determine the best combination pair of the (i) feature extraction and (ii) supervised machine learning, techniques. We overview this process in this Preview paper; for further details see (the first author's thesis): https://ir.lib.uwo.ca.

Step 1: Data Preprocessing
In this step, we first parse the CSV file into a DataFrame for the convenience of processing with Python, with each row in the DataFrame representing a single requirement sample. This involves: (i) parsing the given csv file, (ii) using NLTK library to remove case distinctions, (iii) tokenization and punctuation removal, (iv) stop word removal (v) part of speech tagging, and (vi) stemming and lemmatization.

Step 2: Feature Extraction
We now want to extract features as input to our classification algorithm:

- Requirements are converted into numeric vectors using the BoW in [6].
- TF-IDF scores associated with each term present in a given requirement is used in this classification framework.
- We adopted a feature list proposed by Hussain et al. [7] with which they attained 95% accuracy in the binary classification of FRs and NFRs. (However, note that they did not classify NFRs into sub-types) Thus, adopting the NFR-characteristics from [5], we added a number of syntactic features (e.g., # Adjectives, #Adverbs, and #Nouns) and coded keyword features as part of speech groups.

At this point, three feature sets are prepared. For each of them we have xtrain (data frame with requirement features) and ytrain (data frame that includes target values): (i) For the classification of functional/NFR, ytrain contains 1 (for functional) or 0 (for NFR); and (ii) For multiclass classification of NFRs, ytrain contains 1 for usability, 2 for security, 3 for performance, and 4 for operational types. The output of this step is the three extracted feature sets that are input to machine learning algorithms of training classifiers.

Step 3: Training Classifiers
Here, we investigate the performance of the 10 mentioned classifiers using stratified 10-fold cross-validation technique [2]. Each time the dataset is divided into 10 subsets, nine are used for training and the remaining one is used for testing. We repeat the process 10 times and the performance of the classification is measured as average precision and recall of the 10 repetitions complemented by their execution time.

Step 4: Classifying Requirements
Each of the classifiers trained in Step 3 is used to predict for each requirement whether it belongs to usability, performance, security, or operational.

4 Preliminary Evaluation

This section describes the results of the described investigations. In Sect. 4.1, we give precision and recall to evaluate how well the model learnt to classify non-functional requirements. Comparison with related work is also described. In Sect. 4.2, we show the performance of the model using an industry dataset. Results of the experiments with this dataset are shown as well.

4.1 Preliminary Analysis

Table 2 gives an overview of the results of multi-class classification of NFRs. The average of POS, BoW, and TF-IDF shows that SVM achieved the best results with recall of 0.88 and precision of 0.89. LR, Extra Tree, MNB and SGD- SVM with recall values of 0.86, 0.85, 0.85, and 0.84 (resp.) performed well too. All classifiers achieved recall values above 0.8 except DT, Bagging Tree, and BNB.

Among all the combinations of feature extraction techniques and machine learning algorithms, SVM with TF-IDF (recall: 0.9, precision: 0.92), LR with POS and BOW (recall:0.90, precision: 0.87), and MNB with BOW (recall: 0.90, precision: 0.88) achieved the best results. However, further empirical work is needed to assess the root-causes of the 70-odd misclassified NFRs and how to improve the performance.

Comparison with related work, e.g. [2, 4], is generally favourable (except DT (0.74) and Bagging Tree (0.75) vs. [2]: 0.76). Our best case SVM/TF-IDF vs. Related work are depicted in Table 3.

Table 2. Results of NFR classification

Algorithm	POS			BOW			TF-IDF			Averege		
	Precision	Recall	Time	Precision	Recall	Time	Precision	Recall	Time	Precision	Recall	Time
SVM	0.89	0.87	0.13	0.88	0.87	0.1	0.92	0.9	0.08	0.89	0.88	0.1
SGD SVM	0.85	0.83	0.25	0.88	0.86	0.27	0.84	0.82	0.3	0.86	0.84	0.27
LR	0.9	0.87	0.09	0.90	0.87	0.9	0.9	0.85	0.09	0.9	0.86	0.09
DT	0.78	0.76	0.14	0.77	0.74	0.16	0.76	0.72	0.17	0.77	0.74	0.16
Extra tree	0.88	0.85	1.6	0.88	0.84	1.72	0.9	0.86	1.8	0.89	0.85	1.7
Bagging tree	0.81	0.76	0.87	0.8	0.76	0.94	0.81	0.74	1.03	0.81	0.75	0.94
RF	0.82	0.78	0.2	0.85	0.81	0.22	0.84	0.8	0.21	0.84	0.8	0.21
GNB	0.81	0.81	0.08	0.82	0.82	0.1	0.8	0.79	0.1	0.81	0.81	0.09
MNB	0.88	0.86	0.04	0.9	0.88	0.05	0.89	0.83	0.05	0.89	0.85	0.04
BNB	0.85	0.77	0.08	0.87	0.75	0.1	0.88	0.78	0.11	0.87	0.77	0.09

Table 3. Best case comparison with related work

	Precision	Recall	Execution time
Our	0.92	0.90	0.08s
Reference [2]	0.24	0.76	Not specified
Reference [4]	0.71	0.72	Not specified

4.2 Preliminary Validation

Table 4 shows recall values ≥0.89 and precision values ≥0.92 of the three classifiers using an industry dataset (262 NFRs): SVM, LR, and MNB, for multi-class classification of the four mentioned NFR types. An example NFR from this dataset is:

"The xxx shall perform a failover between an active xxx node and its redundant xxx node partner in no more than 5 seconds."

Table 4. Multi-class NFR classification of the industry dataset

Algorithm	POS			BOW			TF-IDF		
	Precision	Recall	Time	Precision	Recall	Time	Precision	Recall	Time
SVM	0.93	0.91	0.09	0.96	0.95	0.06	0.96	0.96	0.09
LR	0.93	0.92	0.03	0.96	0.94	0.04	0.93	0.90	0.04
MNB	0.92	0.90	0.02	0.94	0.9	0.03	0.92	0.89	0.02

5 Conclusion and Future Work

We used three feature extraction techniques: BoW, TF-IDF, and POS-tagging, combined with 10 supervised machine learning algorithms SVM, SGD SVM, LR, DT, Extra Tree, Bagging Tree, RF, GNB, MNB, and BNB for classifying four NFR types (usability, performance, security, and operational NFRs). Also, we used a combination of PROMISE and PURE datasets (see Sect. 3.1) as our training set. SVM with TF-IDF, LR with POS and BOW, and MNB with BOW achieved the best results (see Table 2) with recall value over 0.90 and precision value over 0.87.

Comparison with related work is favourable (see Sect. 4.1). We also preliminarily validated our results, using an industry dataset (see Table 4), showing recall values $\geq$0.89 and precision values $\geq$0.92. These values suggest that the right NFRs would be rendered, and quickly, to the stakeholders concerned.

In this research, we used a multi-class classifier for the specified NFR types. For future work, we intend to investigate whether developing binary classifiers for each of these NFR types could possibly improve the performance of the classification task at hand.

Also, in our research thus far, we have considered product quality attributes such as usability, performance, security, etc. Apart from extending the coverage of the product quality attributes (e.g., reliability, privacy, and security), a further step in providing a concern-based NFR classification tool for stakeholders is to consider *process* quality aspects such as implementation risk, effort, and cost, which are the bedrock of software projects in industry. For example, recognising risky requirements can help in prioritising these requirements over less risky ones in early decision-making in the development process using the Spiral model of development [8].

Acknowledgments. This work is supported, in part, by Natural Science and Engineering Research Council (NSERC) of Canada.

References

1. Nekvi, I., Madhavji, N.H.: Impediments to requirements-compliance in contractual systems engineering projects: a case study. ACM Trans. Manage. Inf. Syst. **5**(3), 15, 1–35 (2014)
2. Cleland-Huang, J., Settimi, R., Zou, X., Solc, P.: Automated classification of non-functional requirements. Requirements Eng. **12**(2), 103–120 (2007). https://doi.org/10.1007/s00766-007-0045-1
3. Zhang, W., Yang, Y., Wang, Q., Shu, F.: An empirical study on classification of non-functional requirements. In: Proceedings of the 23rd International Conference on Software Engineering and Knowledge Engineering (SEKE), pp. 190–195 (2011)
4. Lu, M., Liang, P.: Automatic classification of non-functional requirements from augmented app user reviews. In: Proceedings of the 21st Int. Conference on Evaluation and Assessment in Software Engineering, June 2017, pp. 344–353 (2017)
5. Sommerville, I., Sawyer, P.: Requirements Engineering: A Good Practice Guide. Wiley, New York (1997)
6. Slankas, J., Williams, L.: Automated extraction of non-functional requirements in avail able documentation. In: Proceedings IEEE 1st Workshop NaturaLiSE, pp. 9–16 (2013)

7. Hussain, I., Kosseim, L., Ormandjieva, O.: Using linguistic knowledge to classify non-functional requirements in SRS documents. In: Kapetanios, E., Sugumaran, V., Spiliopoulou, M. (eds.) NLDB 2008. LNCS, vol. 5039, pp. 287–298. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-69858-6_28
8. Boehm, B.W.: A spiral model of software development and enhancement. IEEE Comput. **21**(5), 61–72 (1988)