Using Machine Learning for Non-Functional Requirements Classification: A Practical Study

Daniel Abella C. M. de Souza
Federal University of Campina Grande (UFCG))
Campina Grande, Paraiba - Brazil
daniel.abella@virtus.ufcg.edu.br

Danyllo Albuquerque
Federal University of Campina Grande (UFCG)
Campina Grande, Paraiba - Brazil
danyllo@copin.ufcg.edu.br

Emanuel Dantas Filho
Federal Institute of Paraiba (IFPB)
Campina Grande, Paraiba - Brazil
emanuel.filho@ifpb.edu.br

Mirko Perkusich
Federal University of Campina Grande (UFCG)
Campina Grande, Paraiba - Brazil
mirko@embedded.ufcg.edu.br

Angelo Perkusich
Federal University of Campina Grande (UFCG)
Campina Grande, Paraiba - Brazil
perkusich@virtus.ufcg.edu.br

ABSTRACT

Non-Functional Requirements (NFR) are used to describe a set of software quality attributes such as reliability, maintainability, and performance. Since the functional and non-functional requirements are mixed together in software documentation, it requires a lot of effort to distinguish them. This study proposed automatic NFR classification by using machine learning classification techniques. An empirical study with three machine learning algorithms was applied to classify NFR automatically. Precision, recall, F1-score, and accuracy were calculated for the classification results through all techniques. The results showed that the SGD SVM classifier achieves the best results where precision, recall, F1-score, and accuracy reported were 0.66, 0.61, and 0.61.

KEYWORDS

Non-Functional Requirements, Classification, Machine Learning, Text Analysis.

1 INTRODUCTION

Functional Requirements (FR) specify software behavior listed by stakeholders, whereas, Non-Functional Requirements (NFR) describe how the software system will provide the means to perform functional tasks. Software requirements specification (SRS) is the most important artifact, which is composed of FR and NFR. Since both types of requirements are mixed within the SRS, it becomes hard to detect and extract manually. Moreover, it is reported that neglecting NFR leads to increased production cost or even project failure [3].

In this context, early NFR detection is essential because it enables system-level constraints to be considered and incorporated into early architectural designs [8]. However, classifying requirements manually is not feasible in terms of required time, budget, and accuracy. To overcome these limitations, various approaches have been presented to classify software NFR into different categories. These approaches were based on retrieval-based [3], linguistic knowledge-based [4], and CNN-based [6]. Although several machine learning techniques were employed for automatic NFR classification tasks, there is no extensive comparison among machine learning feature extraction and classification techniques. Feature selection is an important task in machine learning to increase classification performance [9]. However, investigation of NFR classification accuracy combining different feature extraction techniques such as Bag of Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF). with various classification algorithms named Multinomial Naive Bayes (MNB), Gaussian Naive Bayes (GNB), Bernoulli Naive Bayes (BNB), K-nearest Neighbors (KNN), Support Vector Machines (SVM), Stochastic Gradient Descent (SGD SVM) and Decision Trees (Dtree) a significant research directions.

This paper proposed an approach to the automatic NFR classification technique combining feature extraction with three machine learning algorithms. The PROMISE [1] software requirement dataset was used to investigate the classification accuracy of considered techniques with several measures. The requirements dataset has been pre-processed to remove unnecessary characters, which will be converted to features using the TF-IDF technique. This technique was chosen due to its performance compared to another technique [2], and it has been widely used in other NFR classification studies [10][3]. Next, BNB, KNN, SVM, and SGD algorithms were applied to train the classifier. Finally, the classification performance of each method has been compared using different NFR classifying results.

The remainder of this paper is organized as follows; Sections II and III illustrate the proposed method and result analysis, respectively. Finally, sections IV and V expose the main limitations of this study and conclude this paper with future research directions.

2 STUDY CONFIGURATION

This study seeks to find the best pair of feature extraction and machine learning algorithms aiming at software requirement classification. This process considers NB, KNN, SVM, and D-tree machine learning algorithms and TF-IDF feature extraction techniques. The whole process of this study is divided into the following four steps and depicted in Figure 1. The input of this method was the PROMISE requirement specification. The details regarding the steps are elaborated in what follows.

Data Processing. Initially, we seek to clean and preprocess the dataset used in this study. This task was composed of four sub-steps:

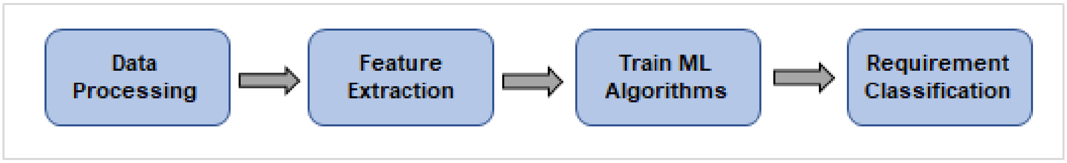

Figure 1: Method Overview.

(1) *Removing Special Characters*: Special characters and symbols, typically non-alphanumeric, introduce extraneous noise into the experimental dataset. To enhance data quality, regular expressions were employed to systematically remove these special characters.
(2) *Applying Case-Folding*: In English, words can appear in either upper-case or lower-case forms while conveying similar meanings. To ensure consistent treatment of cases, a case-folding procedure was implemented.
(3) *Excluding Stop Words*: Certain words, such as "a," "an," and "the," have minimal impact on the semantic content of software requirements. For this study, a commonly used natural language processing stop words corpus from the Natural Language Toolkit (NLTK) was employed to exclude these terms.
(4) *Tokenization*: In this sub-step, longer blocks of text are divided into smaller units, referred to as tokens. In our case, software requirements were tokenized into sentences, which were subsequently segmented into individual words.

Feature Extraction. This step transforms a list of essential words into a feature set used by a machine learning (ML) classifier. The textual dataset consists of documents that represent words, sentences, or paragraphs of free text, and those are inherently unstructured. Extracting meaningful features from the textual dataset is essential to building text classification by the ML model. In this step, textual data was converted from text into some numeric representations to be understood by ML algorithms.

Initially, this research went through the standard steps for text pre-processing. TF-IDF supported the feature extraction process from the pre-processed text. This technique scores associated with each term present in a given requirement. The TF-IDF term weighting technique assigns a higher score to rare words and a lower one to words occurring frequently across all requirements. After cleaning the requirements strings, this technique was applied to extract textual features to train the ML classifiers.

Train ML Classifiers. The TF-IDF was used to extract textual features, which act as the input of ML algorithms of training classifiers. The experimental tasks involved four ML algorithms (i.e., BNB, KNN, SVM, and SGD). These algorithms were applied for requirement classification to compare the effectiveness and performance.

Requirement Classification. Each ML classifier will output for a given requirement whether it belongs (or not) to a specific category. For instance, to requirement classification according to the category “Maintainability” (M), the classifier should return the list of requirements for which it received a positive answer.

Also, the other documents will be classified accordingly. The combination of textual feature extraction methods and four machine learning algorithms have been applied in this software requirements classification study. The textual dataset was converted into vector representations to be fed as input in ML algorithms.

3 RESULTS AND DISCUSSION

The experimental study utilized a requirement corpus, and the evaluated results are presented in this section. Precision, recall, and the F1-score were measured to evaluate how well the model learned to classify software requirements. The labeled data was used for training, and unlabeled text strings were used for testing. The experimental classification results of the NFR dataset for different machine-learning algorithms are shown in Table II and Table III.

3.1 Dataset

The PROMISE software requirement dataset was used for the experimental tasks involved in this study [1]. This dataset was chosen to facilitate the replication of this study by independent researchers. Additionally, this dataset is widely used for other studies related to software requirement classification. The PROMISE dataset contains a set of labeled FR and NFR. The dataset comprises about 625 requirement sentences, with 255 functional and 370 non-functional requirements. The NFR is labeled with eleven categories (i.e., availability, legal, look and feel, maintainability, operational, performance, scalability, security, usability, fault tolerance, and portability).

3.2 Experimental Environment

The first and second authors conducted the experimental tasks in an environment featuring an Intel Core i7 processor with 32GB of RAM, operating on the Windows platform. The computing environment was configured with Google Colab in a notebook setting, and we employed a suite of essential packages, including Pandas, NumPy, NLTK, scikit-learn (sklearn), and Matplotlib, for tasks such as data loading, preprocessing, and results evaluation.

3.3 Results

The dataset, meticulously prepared for this study, was employed for both training and testing to assess the classification accuracy of machine learning classifiers. The results of the experimental NFR classification, using TF-IDF feature extraction in conjunction with four distinct ML algorithms (KNN, SVM, SGD SVM, and DTree), are presented comprehensively in Table 1.

In Table 1, a comprehensive summary of the experimental outcomes is provided. These outcomes were achieved through the utilization of the TF-IDF feature extraction technique, considering

Table 1: Precision, Recall and F1-score results comparison.

	TF-IDF Feature Extraction								
	word level			n-gram			Character level		
	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score
KNN	0.70	0.60	0.64	0.74	0.52	0.57	0.71	0.63	0.64
SVM	0.68	0.58	0.62	0.45	0.24	0.27	0.72	0.57	0.62
SGD	0.70	0.68	0.68	0.77	0.65	0.67	0.77	0.68	0.69
DTree	0.25	0.23	0.21	0.14	0.11	0.09	0.13	0.19	0.16

character and word levels, as well as n-grams. The table showcases the weighted average precision, recall, and F1-score for the classification results, offering valuable insights into the performance of the models.

Our analysis reveals that TF-IDF feature extraction at the character level outperforms the other methods considered. Specifically, the highest precision, recall, and F1 score were achieved using the SGD SVM technique across all feature extraction methods. Notably, SGD SVM exhibited its superior performance when paired with TF-IDF (character level), yielding precision, recall, and F1-score values of 0.77, 0.68, and 0.69, respectively. Conversely, the combination of DTree with TF-IDF (N-gram) demonstrated the least favorable results, with precision, recall, and F1-score reaching only 0.14, 0.11, and 0.09, respectively.

Table 2 provides a summary of the weighted average scores across the various cases, with SGD once again emerging as the top-performing method. Specifically, SGD achieved precision, recall, F1-score, and accuracy scores of 0.74, 0.67, 0.68, and 0.83, respectively. In contrast, KNN and SVM exhibited moderate performance, with KNN attaining precision, recall, F1-score, and accuracy values of 0.72, 0.58, 0.62, and 0.78, respectively, while SVM reported 0.62, 0.47, 0.50, and 0.73 for precision, recall, F1-score, and accuracy, respectively.

Table 2: Precision, Recall and F1-score average.

	Precision	**Recall**	**F1-Score**	**Accuracy**
KNN	0.72	0.58	0.62	0.78
SVM	0.62	0.47	0.50	0.73
SGD	0.74	0.67	0.68	0.83
DTree	0.17	0.18	0.15	0.48

Based on the empirical findings, it is evident that the combination of SGD and TF-IDF feature extraction at the character level consistently outperforms other machine learning and feature extraction combinations in the context of NFR classification. It is important to note that this study does not encompass FR classification, focusing exclusively on the assessment of NFR classification.

4 THREATS AND LIMITATION

In this section, we discuss the limitations and potential threats to the validity of our work, following the guidelines outlined in [7], and elaborate on how these threats were partially mitigated.

Limitations refer to issues that our approach cannot currently address. We used the PROMISE dataset, which came pre-labeled with FR and NFR classifications. The correctness of such labeling cannot be guaranteed, and any inaccuracies could potentially affect the outcomes of this study.

Construct Validity focuses on whether theoretical constructs are accurately interpreted and measured. One threat to construct validity in this study concerns the correct classification of sentences used in our experiments. To ensure consistent and accurate classification of various NFR types, we adopted the definitions of NFR types as outlined in the ISO 25010 standard [5].

Internal Validity is concerned with the study's design and whether results logically follow from the data. A potential threat to internal validity in our study is the risk of machine learning overfitting during testing [10]. To mitigate this risk, we implemented 5-fold cross-validation in our experiments, enhancing the robustness of our results.

External Validity pertains to the extent to which our findings can be generalized to other settings. A specific threat here relates to the absence of certain NFR types in the experimental dataset. To improve external validity, we plan to conduct a large-scale empirical study in the future, expanding the scope of NFR types considered.

Reliability concerns whether the study would yield consistent results if replicated by other researchers. In this study, we assume that employing the TF-IDF technique for feature extraction is the optimal choice for our purposes. However, we cannot guarantee that alternative techniques, such as Bag of Words (BoW) or Chi-Squared, would yield superior results. This remains an avenue for further exploration in future research endeavors.

5 FINAL REMARKS

This study presents an automated approach for the classification of Non-Functional Requirements (NFR) using machine learning techniques. We conducted experiments using the well-established PROMISE NFR dataset, which categorizes requirements into distinct categories. Following feature extraction through the TF-IDF technique, we employed four machine learning classifier algorithms—BNB, KNN, SVM, and SGD—to classify requirements into predefined categories.

Among the classification algorithms, SGD demonstrated superior performance compared to the others considered in this study. This particular machine learning classifier achieved notable average precision, recall, F1-score, and accuracy values of 0.74, 0.67, 0.68, and 0.83, respectively. In contrast, the decision tree algorithm exhibited the least favorable results, reporting precision, recall, F1-score, and accuracy scores of 0.17, 0.18, 0.15, and 0.48, respectively. However, SVM and KNN attained moderate scores, surpassing DTree but falling short of the SGD method. Furthermore, the TF-IDF feature extraction technique at the character level consistently demonstrated higher average scores. This empirical investigation highlights the effectiveness of TF-IDF paired with the SGD algorithm for achieving optimal results in NFR classification.

Looking ahead, this paper's future research direction encompasses the incorporation of additional classification algorithms, such as Multinomial Naive Bayes (MNB), Gaussian Naive Bayes (GNB), and Bernoulli Naive Bayes (BNB), as well as the exploration of alternative feature selection techniques like "Bag of Words (BoW)" and "Chi-Squared". Additionally, replicating these experimental tasks in a real-world setting holds the potential to validate the practical utility of our approach in the context of NFR classification.

REFERENCES

[1] G Boetticher. 2007. The PROMISE repository of empirical software engineering data. *http://promisedata. org/repository* (2007).

[2] Andrea Bommert, Xudong Sun, Bernd Bischl, Jörg Rahnenführer, and Michel Lang. 2020. Benchmark for filter methods for feature selection in high-dimensional classification data. *Computational Statistics & Data Analysis* 143 (2020), 106839.

[3] Jane Cleland-Huang, Raffaella Settimi, Xuchang Zou, and Peter Solc. 2006. The detection and classification of non-functional requirements with application to early aspects. In *14th IEEE International Requirements Engineering Conference (RE'06)*. IEEE, 39–48.

[4] Ishrar Hussain, Leila Kosseim, and Olga Ormandjieva. 2008. Using linguistic knowledge to classify non-functional requirements in SRS documents. In *International Conference on Application of Natural Language to Information Systems*. Springer, 287–298.

[5] ISO Iso. 2011. Iec25010: 2011 systems and software engineering–systems and software quality requirements and evaluation (square)–system and software quality models. *International Organization for Standardization* 34, 2910 (2011), 208.

[6] Raul Navarro-Almanza, Reyes Juarez-Ramirez, and Guillermo Licea. 2017. Towards supporting software engineering using deep learning: A case of software requirements classification. In *2017 5th International Conference in Software Engineering Research and Innovation (CONISOFT)*. IEEE, 116–120.

[7] Forrest Shull, Janice Singer, and Dag IK Sjøberg. 2007. *Guide to advanced empirical software engineering*. Springer.

[8] John Slankas and Laurie Williams. 2013. Automated extraction of non-functional requirements in available documentation. In *2013 1st International workshop on natural language analysis in software engineering (NaturaLiSE)*. IEEE, 9–16.

[9] Bing Xue, Mengjie Zhang, Will N Browne, and Xin Yao. 2015. A survey on evolutionary computation approaches to feature selection. *IEEE Transactions on Evolutionary Computation* 20, 4 (2015), 606–626.

[10] Hui Yang and Peng Liang. 2015. Identification and Classification of Requirements from App User Reviews.. In *SEKE*. 7–12.