TECHNIUM

Technium Vol. 3, Issue 11 pp.32-39 (2021)
ISSN: 2668-778X
www.techniumscience.com

Supporting Classification of Software Requirements system Using Intelligent Technologies Algorithms

Ashraf Abdulmunim Abdulmajeed [1], Younis S. Younis [2]

[1]Department of Software, College of Computer Sciences and Mathematics, University of Mosul. Mosul / Iraq.
[2]Ministry of Education, Nineveh Education Directorate, Mosul / Iraq.
[1] Email: ashraf_althanoon@uomosul.edu.iq, [2] E-mail: fajirnet1@yahoo.com

Abstract. The important first stage in the life cycle of a program is gathering and analysing requirements for creating or developing a system. The classification of program needs is a crucial step that will be used later in the design and implementation phases. The classification process may be done manually, which takes a lot of time, effort, and money, or it can be done automatically using intelligent approaches, which takes a lot less time, effort, and money. Building a system that supports the needs classification process automatically is a crucial part of software development. The goal of this research is to look into the many automatic classification approaches that are currently available. To assist researchers and software developers in selecting the suitable requirement categorization approach, those requirements were divided into functional and non-functional requirements. since natural language is full of ambiguity and is not well defined, and has no regular structure, it is considered somewhat variable. This paper presents machine requirement classification where system development requirements are categorized into functional and non-functional requirements by using two machine learning approaches. During this research paper, MATLAB 2020a was used, as well as the study's results indicate When applying Multinomial Naive Bayes technology, the model achieves the highest accuracy of 95.55 %,93.09 % sensitivity, and 96.48 % precision, However, when using Logist Regression, the suggested model has a classification accuracy of 91.23 %,91.54 % sensitivity, and 94.32 % precision.

Keywords. Requirement Classification, Functional Requirements, On-Functional Requirements, Feature Extraction, Text Normalization.

1. Introduction

The building or developing of any software system requires five stages (analysis, design, implementation, testing, and maintenance). The analysis stage is related to requirements analysis, i.e., translation of system needs, and this stage is also called requirements engineering. This stage is the basis for building software engineering. These analyzed requirements define the user's expectations of the software to be built. The requirements analysis and classification stage are the first and crucial stage in building software applications [1].

During software development we note that software requirements consist of different types, as an example, During the requirements gathering phase, the requirements can be classified as functional (explicit features or functions of the product) or non-functional requirements (criteria of risk or quality that are implicit in the product), and through the analysis of requirements we note The

requirements may serve different purposes, such as showing security weaknesses in the product, as well as measuring the necessities of scalability and adapting to any future environment, to evaluating the general appearance and appearance of the product[2].
This research looks at how two different machine learning methods, Logist Regression (LR) and Multinomial Naive Bayes (MNB), might help enhance classifying requirements. The chosen machine learning algorithms are used in this research utilizing several stemming approaches as well as certain preprocessing strategies. Automatic categorization is used to divide requirements into functional and non-functional categories.

This paper is divided into six sections: The background and related work on the classification requirements are covered in Section Two. In Section Three, the research technique is thoroughly described. In Section Four, we'll look at how to classify software requirements. In Section Five, the findings are described. In Section Six, the conclusion is presented, after that, there's an acknowledgment and a list of references.

2. Background and related works

In this section, we'll look into various articles that deal with needs analysis, such as surveys, classification, and summarization.
One of these studies is [3], in which the authors offer a method for partially prioritizing software needs. To build a prioritized list of criteria, they used several clustering data mining algorithms. The results of this study's trials revealed that applying automated algorithms to evaluate software needs was beneficial.
Kur Tanovic et al. [4] classified needs into Functional Requirements, Non-Functional Requirements, and categories of non-functional requirements using the Support Vector Machines technique. The authors employed the PROMISE repository, which is known for having an uneven set of functional and non-functional criteria. User comments on Amazon items were incorporated into the primary data set to help balance the database.
Another study [5] uses Support Vector Machine to offer an automated approach for requirement categorization. The authors used two Mercedes-Benz vehicle specs to test the machine learning techniques. The findings of this study demonstrated that automated requirement categorization is significant and provides a satisfactory degree of accuracy.
Jindal et al. [6] used a single machine learning method to undertake an automated analysis of multiple software needs specifications from the PROMISE repository and introduce binary categorization into several sorts of security requirements categories (decision tree). Tokenization, stemming, and stop word removal were utilized as pre-processing techniques, while vectorization was conducted using TF-IDF.
To classify NFRs automatically, a semi-supervised automated technique was used [7]. The authors of this study aimed to aid requirement analyzers in the categorization of requirements. To begin, the requirement analysis team selected and categorized a collection of requirements that would serve as the system's training set. After that, the machine was able to identify the remaining unclassified needs. Using a common set of requirement papers, the authors employed some supervised machine learning methods. [8] proposes a framework for automated categorization of requirements. The authors presented a framework for requirement categorization in both top-down (top-down method) and bottom-up (bottom-up approach) (bottom-up approach). The top-down approach proceeds from corporate goals to SRS, with minor details in between. The bottom-up approach progresses from a detailed SRS to business objectives. A list of classed needs and a dictionary of frequently repeated key phrases in each classification are the framework's outputs. Another future framework, according to the authors, may be used to automate the last categorization phase. The second framework generates a list of needs that have been categorized.

Technium Vol. 3, Issue 11 pp.32-39 (2021)
ISSN: 2668-778X
www.techniumscience.com

3. Methodology

This section describes the method proposed by the researcher to classify the requirements during the software development stage. Requirement's analysis and classify process contains five stages: Gather requirements (Dataset), preprocessing, transformation, classification, and testing in figure 1 explains the block diagram including the suggested work's approach.

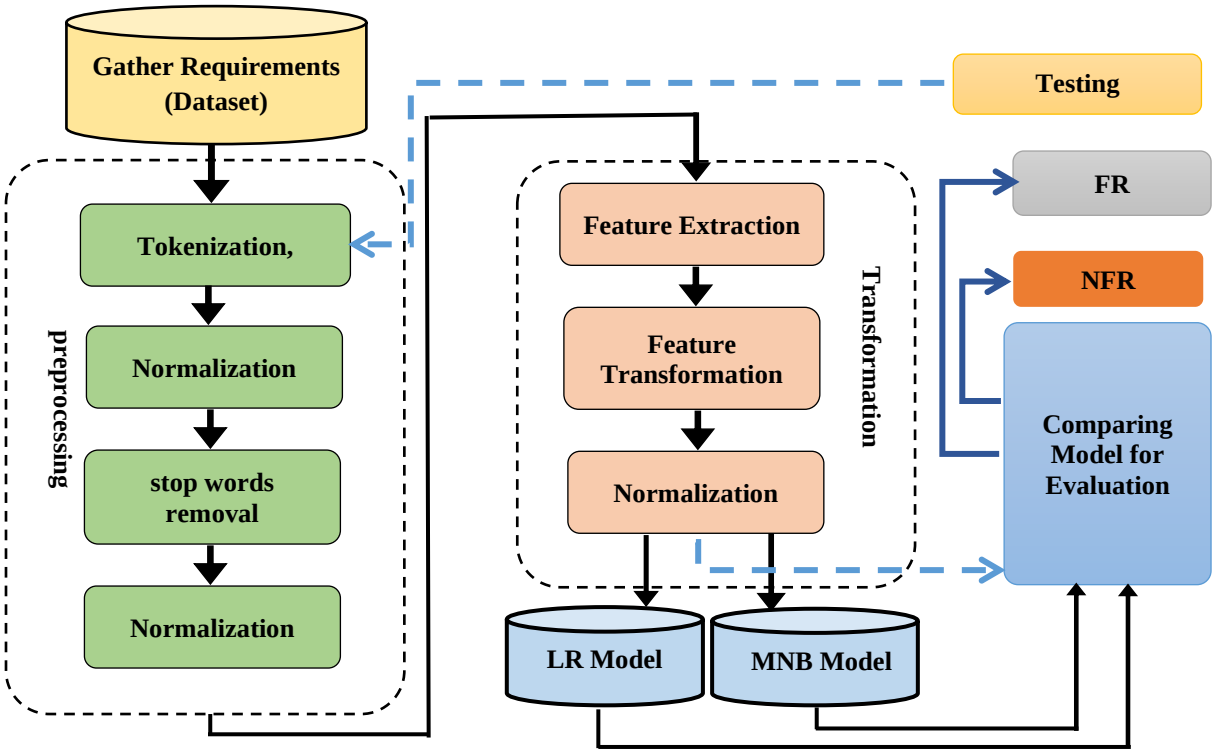

Figure 1 explains the block diagram including the suggested work's approach.

1.1. Gather Requirements

The data set that was used in this study contains 625 requirements [8] collected from multiple resources to build a dataset to be analyzed. A dataset was categorized as functional or non-functional throughout the data-gathering phase. There are 255 functional requirements and 370 non-functional requirements in the dataset. In this work, 70% of the dataset is used to train the models, and 30% of the dataset is used to test the models. The two categories are described explained below*:*

A- A Functional Requirement: is a demand for a consequence of behavior that must be given by a system function and might include specific product behaviors, features, and use cases. Functional requirements are divided into two categories: high-level requirements (HLR) and low-level requirements (LLR) [9].

B- Non-Functional Requirements: According to Davis et al [10], a Non-Functional Requirement of a system is described as "anything that defines the non-behavioral features of a system, encapsulating the qualities and restrictions under which a system must function.". The non-functional requirements are labelled with different letters according to NFR types: A=Availability, O= Operational, L = Legal, LF = Look and feel, US = Usability, SC = Scalability, SE = Security, FT = Fault tolerance, MN = Maintainability, PE = Performance , and PO = Portability. The dataset is collected from different projects. The dataset has three attributes (Project ID, Requirement Text, Class) see Table 1 [11].

TECHNIUM
Technium Vol. 3, Issue 11 pp.32-39 (2021)
ISSN: 2668-778X
www.techniumscience.com

Table 1. Shows the dataset example.

Project ID	Requirement Text	Class
1	"The system shall allow modification of the display"	F
2	"The product shall be easy for a realtor to learn"	US
3	"The product shall be able to support multiple remote users"	SC
4	"The product shall run on the existing hardware for all environments"	O

1.2. Pre-Processing

In this phase, every requirement in the dataset is tokenized, normalized, and stop words in requirements are deleted during this step. Real-world natural language documents (or corpora) are frequently cluttered with errors, noise, complicated sentence patterns, and diction variance The second step is after gathering the requirements step in This research is concentrated on the pre-processing step (Table 2 shows an example of the pre-processing step). It generally consists of four steps:

A- Tokenization. Tokenization is the initial stage in the Pre-Processing, according to the researcher's desire, this requires separating a text into a list of tokens, which can be phrases or individual words [12].

B- Normalization. The process of Normalization text into a single, uniform form is known as normalization. Text is normalized by putting common letters in the same form, removing repetitive words, and removing repeated letters within the same word [13].

C- Stop Words Removal. Stop words (e.g., "a," "the," and "to") are relatively common words that have little to no significance in corpus processing [14]. Stop words can sometimes be removed from a document to decrease noise and enhance NLP performance.

D- Stemming. The practice of extracting the roots or stems of words is known as stemming. The root of a term is the component of it that can no longer be studied. The stem is made up of root with a few alterations [15].

Table 2 shows the pre-processing step.

Text	**Ashraf needs to build a system to determine the weather, this system works works and performs well on a daillllllly basis and without errors.**
Tokenization	**(Ashraf), (needs), (to), (build), (build), (a) (system), (to), (determine), (the), (weather), (this), (weather), (system), (works), (works), (and), (performs),(well), (on), (daillllllly),(basis), (and),(without),(errors).**
Normalization	**(Ashraf), (needs), (to), (build),(a),(system), (to), (determine), (the), (weather), (this), (weather), (system), (works),(and), (performs), (well),(on), (daily), (basis), (and), (without),(errors).**
Stop Words Removal	**(needs), (build), (system), (determine), (weather), (system), (works), (performs), (well), (daily), (without), (errors).**
Stemming	**(need), (build), (system), (determine), (weather), (system), (work), (perform), (well), (daily), (without), (error).**

1.3. Transformation

The third step of the proposed method for classifying requirements includes text transformation which is consisted of (feature extraction, feature transformation, and feature selection):

A- Feature extraction. After pre-processing step, This phase is required so that we may extract features from data that can be utilized to train the machine learning model [16] using the information retrieved from the documents.
B- Feature transformation. In this step, a choice is made to select the most important useful features that we can use in training the proposed model [17]. In this study, every word is treated as a separate character in this study. Every requirement text is broken into a collection of tokens, each token being one word, and each word (token) being a feature.
C- Feature selection. The most important feature selection phase and useful features to be used in model training, where the feature selection goes through a filtering phase to exclude some features that are not so important [18].

The next step is a trained model, in this step, to categorize requirements from the previous stage, two machine learning algorithms are used. The last step is the testing and evaluation of the trained model.

4. Classifying Software Requirements

In this section, the proposed technique for Automatic classifying is used to divide requirements into functional and non-functional categories. The technique is an automatic technique that uses two machine learning techniques (Logistic Regression (LR) and Multinomial Naive Bayes). Logistic Regression (LR) and Multinomial Naive Bayes, to help enhance classifying systems. The chosen machine learning algorithms are used in this research utilizing several stemming approaches as well as certain preprocessing strategies. In the proposed approach consists of multiple phases (Gather requirements phase, data pre-processing, data transformation, classification, and testing).
The dataset(requirements) used in this work includes 625 requirements collected from multiple resources to build a dataset to be analyzed. There are 255 functional requirements and 370 non-functional requirements in the dataset. In this work, 70% of the dataset is used to train the models, and 30% of the dataset is used to test the models. All content elements in the proposed approach were by using standard preprocessing steps such as tokenization, normalization, stop words removal, and stemming.
Following that, the requirements are tokenized into a collection of tokens. As previously stated, requirements are normalized (Table 2). A Multi-Stop word is used as a stop words handler in stop words removal. A list of stop words is created, and if a word appears in this list, it is identified as a stop word and eliminated from the required text. Table 2 is an example of a stop word. The next step is Data Transformation. Every required text is broken into a series of tokens using a word tokenizer. each token is one word, each word (token) is a feature. One of the (machine learning) methods is used in this stage. Two machine learning techniques are utilized in this research to experiment with and test the proposed model ("Logist Regression" and "Multinomial Naive Bayes").
The models are trained using the specified characteristics

5. Results

In this part, we provide the experiment's findings and explore their ramifications in this section. The MNB beats the LR in the requirement classification job using the suggested methodologies. We also compute Accuracy, Sensitivity, and Precision, which are derived as follows [19]:

$$\text{Accuracy} = \frac{TP+TN}{TP+FP+TN+FN} \quad (1)$$

$$\text{Sensitivity} = \frac{TP}{TP+FN} \quad (2)$$

$$\text{Precisions} = \frac{TP}{TP+FP} \quad (3)$$

In the testing phase, accuracy is used to assess the classification algorithm's accuracy and capacity to classify categories. As may be seen in (Eq 1).
The proportion of predicted accurate labels to the total number of is defined as precision (Eq 2), while the proportion of predicted correct labels to the total number is defined as sensitivity (Eq 3). TP, TN, FP, and FN are used to calculate these metrics. Functional requirements that are correctly classified as functional are referred to as TP (True Positive). Non-functional needs that are accurately categorized as non-functional requirements are referred to as TN (True Negative). Non-functional needs that are wrongly identified as functional requirements are referred to as FP (False Positive). Functional needs that are wrongly labeled as non-functional requirements are referred to as FN (False Negative).

Table 2. Results of the requirement classification Classifier's performance evaluation

Type of technologies used	Performance Evaluation Metric		
	Accuracy	Sensitivity	precision
MNB	95.55 %	93.09 %	96.48 %
LR	91.23 %	91.54 %	94.32 %

The suggested methodologies determine Accuracy, Sensitivity, and Precision, as shown in table 3 above. Through the results we obtained from this research paper, it is evident to us that MNB outperforms with high accuracy compared to LR. The main contribution of this paper is the application of two different technologies for classifying the requirements and the summarization of these techniques which help the other researchers as well as software engineers in both research and practical work When building or developing software.
“The experimental studies were implemented using the MATLAB 2020a, all applications were run on a laptop Lenovo, CPU: Intel Core i3-70204 CPU @2.3 GHZ. (RAM: 8 GB/ 512 GB SSD /Windows 10 pro- 64-bit)”

6. Conclusions

In this study, the MNB outperforms LR in the requirement classification task considering the accuracy, Sensitivity, and Precision measurement. The main contribution of this paper is the application of two different technologies for classifying the requirements and the summarization of these techniques which help the other researchers as well as software engineers in both research and practical work When building or developing software.
The research concludes that the enhancement in the pre-processing task steps has a huge effect on the result of automatic requirement classification.
In the future, we may be able to classify requirements using a SVM or a CNN. We believe this will provide tremendous assistance to software engineers.

Acknowledgment

"The authors would like to thank the University of Mosul / College of Computer Science and Mathematics for their facilities, which have helped to enhance the quality of this work".

References

[1] Y. Bassil, "A Simulation Model for the Waterfall Software Development Life Cycle," International Journal of Engineering & Technology (iJET), vol. 2, no. 5, pp. 2049-3444, 2012.

[2] H. Femmer, J. Mund, and D. Méndez Fernández, "It's the Activities,Stupid!: A New Perspective on RE Quality," in 2nd International Workshop on Requirements Engineering and Testing (RET'15), 2015.

[3] Z. S. H. Abad and G. Ruhe, "Using real options to manage Technical Debt in Requirements Engineering," in Requirements Engineering Conference (RE), 2015 IEEE 23rd International, Ottawa, ON, Canada, 2015.

[4] Kurtanovi´c, Z.; Maalej, W. Automatically classifying functional and non-functional requirements using supervised machine learning. In Proceedings of the 2017 IEEE 25th International Requirements Engineering Conference (RE), Lisbon, Portugal, 4–8 September 2017; pp. 490–495.

[5] E. Knauss and D. Ott, "Automatic requirement categorization of large natural language specifications at mercedes-benz for review improvements," in in Proceedings of the 19th International Conference on Requirements Engineering: Foundation for Software Quality, ser. REFSQ'13. Berlin, Heidelberg: Springer-Verlag, 2013.

[6] Jindal, R.; Malhotra, R.; Jain, A. Automated classification of security requirements. In Proceedings of the 2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI),Jaipur, India, 21–24 September 2016; pp. 2027–2033.

[7] A. Casamayor, D. Godoy and M. Campo, "Identification of non-functional requirements in textual specifications: A semi-supervised learning approach," ELSEVIER, vol. 52, no. 4, pp. 436-445, 2010.

[8] T. K. R. P. D. Menzies, "The Promise Repository of Empirical Software Engineering Data," Department of Computer Science, North Carolina State University, 2016.

[9] Pohl, K.; Rupp, C. Requirements Engineering Fundamentals; O'Reilly Media, Inc.: Sevastopol, CA, USA, 2011.

[10] Davis, Alan M.: Software Requirements: Objects, Functions, and States. Prentice- Hall, Inc., USA, 1993, ISBN 013805763X. 4.

[11] Kurtanovi´c, Z.; Maalej, W. Automatically classifying functional and non-functional requirements using supervised machine learning. In Proceedings of the 2017 IEEE 25th International Requirements Engineering Conference (RE), Lisbon, Portugal, 4–8 September 2017; pp. 490–495.

[12] N. Indurkhya and F. J. Damerau. Handbook of Natural Language Processing. Chapman & Hall/CRC, 2nd edition, 2010.

[13] P. Krishnamurthy, P. P. Talukdar, N. Sridhar, A. Ramakrishnan and K. Bali, "Hindi Text Normalization," in Fifth International Conference on Knowledge Based Computer Systems (KBCS), Hyderabad, India, 2004

[14] C. D. Manning, P. Raghavan, and H. Sch• utze. Introduction to Information Retrieval. Cambridge University Press, New York, NY, USA, 2008.

[15] N. Indurkhya and F. J. Damerau. Handbook of Natural Language Processing. Chapman & Hall/CRC, 2nd edition, 2010.

[16] S. Sarawagi , "Information Extraction," Foundations and Trends in Databases, vol. 1, no. 3, p. 261–377, 2007.

[17] L. M. Rose, N. Matragkas, D. S. Kolovos and R. F. Paige, "A feature model for model-to-text transformation languages," in Modeling in Software Engineering (MISE), 2012 ICSE Workshop on, Zurich, Switzerland, 2012.

[18] Géron, A. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow; O'Reilly Media, Inc.:Sevastopol, CA, USA, 2019.

[19] Abdulmajeed, Ashraf Abdulmunim, Saleem, Nada Nimat, "Intelligent tool for detecting Covid-19 using convolutional neural network based on both CT and x-ray lung images", AIP publishing, AIP Conference Proceedings 2404,2nd international conference on engineering and science, 2021.