Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Identifying Non-functional Requirements from Unconstrained Documents using Natural Language Processing and Machine Learning Approaches

QAIS A. SHREDA[1], ABUALSOUD A. HANANI[2]

[1]Software Engineering master program, Birzeit University, Palestine (e-mail: qays.shreda@gmail.com)

[2]Electrical and Computer Engineering, Birzeit University, Palestine (e-mail: ahanani@birzeit.edu)

Corresponding author: Abualsoud A. Hanani (e-mail: ahanani@ birzeit.edu).

ABSTRACT Requirements engineering is the first phase in software development life cycle and it also plays one of the most important and critical roles. Requirement document mainly contains both functional requirements and non-functional requirements. Non-functional requirements are significant to describe the properties and constraints of the system. Early identification of Non-functional requirement has direct impact on the system architecture and initial design decision. Practically, non-functional requirements are extracted manually from the document. This makes it tedious, time-consuming task and prone to various errors. In this paper, we propose an automatic approach to identify and classify non-functional requirements using semantic and syntactic analysis with machine learning approaches from unconstrained documents. We used A dataset of public requirements documents (PURE) that consists of 79 unconstrained requirements documents in different forms. In our approach, features were extracted from the requirement sentences using four different natural language processing methods including statistical and state-of-the-art semantic analysis presented by Google word2vec and bidirectional encoder representations from transformers models. The adopted approach can efficiently classify non-functional requirements with an accuracy between 84% and 87% using statistical vectorization method and 88% to 92% using word embedding semantic methods. Furthermore, by fusing different models trained on different features, the accuracy improves by 2.4% compared with the best individual classifier.

INDEX TERMS Software requirement, Machine learning, Natural language processing.

I. INTRODUCTION

REQUIREMENTS Engineering (RE) can be defined as "a set of activities for exploring, evaluating, documenting, consolidating, revising and adapting the objectives, capabilities, qualities, constraints and assumptions that the system-to-be should meet based on problems raised by the system-as-is and opportunities provided by new technologies" [1]. It is documented in a form called requirements document (RD) to be suitable for communication, analysis and subsequent implementation. RE is a crucial phase at the beginning of software development life cycle (SDLC). It is one of the most important and critical role in systems and software projects, and all subsequent stages depend on them.

Requirements document is commonly written by a software analyst in cooperation with customer's experts in natural language text for ease of communication within a community stakeholders. Several contractors who can bid for the contract. Furthermore it may also contain diagrams or screenshots. The decision to represent requirements in natural language has pertinent interpretation. First, natural language is understandable and accepted by most people. Second, the ultimate purpose of the project is to produce a system satisfies the user requirements. [1].

The input of requirements documentation phase are a bunch of agreed statements of different types: general objectives, system requirements, software requirements, environmental assumptions, relevant domain properties and concept definitions. They are elicited through various activities such as one to one or group interview, workshops, questionnaires, use-cases and prototyping. The output of the specification and documentation phase is the first version of the requirements document [34].

There are a wide range of techniques that are usually used for requirements specification and documentation. Unconstrained document is one of these techniques. It is prose in natural language without specific rules. This technique has several advantages: there are no limitations in expressiveness on what can be specified in natural language. Furthermore, free text in natural language can be understood by all parties, and no special training is required. On the downside, unconstrained requirements document prose in natural language that is prone to several defect ,such as notably ambiguities, noises, remorse, immeasurably statements and opacity. Other technique, disciplined documentation in structured natural language, where the requirements engineer follow local rules on how statements should be written in natural language, or global rules on how the requirements document should be organized [2].

Typically, software requirements are classified into two types of requirements: functional requirements (FR) and non-Functional requirements (NFR). This classification helps to understanding the common characteristics of different types of needs. NFR are very significant to describe the properties and constraints of the system.

Since the early days of software engineering NFR have been existed. The numbers of their categories are estimated to be more than 150 categories. IEEE-Std 830-1993 identified only 13 NFR from 252 NFR types. The most five NFR commonly used in several domains are: reliability, performance, security, maintainability, and usability. In this study, we focus on those five NFR, in addition to availability that most of projects types probably need.

The importance of RE is enormous to develop an effective software and reduce software errors of software development. For example, system design and architecture must carefully consider constraints and NFR which directly affect on initial design in the early stages. Undetected ignored requirements until a later stage in software development life cycle will be very costly and greatly affect on customer satisfaction [35]. The arising errors have been caused by incorrect requirements have become a significant problem in software development. Problems caused by requirements errors typically make up 25 % to 70 % of total software errors in the USA in 2012 [3].

Early identification of NFR is very important in the evaluation of alternatives architectural and design decisions. System architects needs NFR to determine constraints as: security, reliability, performance, scalability, availability etc, in order to design the system architecture. In contrast of FR, NFR are significantly difficult to handle changes due to absent or missing NFR [2].

Requirements document are written in natural language and can be processed as any text document. It contains paragraphs, sentences, and words. It has much in common with natural language documents challenges, such as semantic and syntactic ambiguity, synonymy, coherence, and personality intention and style. These challenges promote us to apply the state-of-the-art Natural language processing (NLP) techniques such as bidirectional encoder representations from transformers (BERT) and word2vec embedding models. These models are the most modern effective approaches that have a capability to capture the embed and context of the requirement sentences or documents including semantic and syntactic meaning.

Requirements analysis are considered as one of the most common problematic activities in SDLC. Practically, requirements are extracted manually from the requirement document. This makes it a tedious task, prone to various errors, requires a lot of effort and time consuming. Where each requirements document need to be read, analysed, and classified manually. Furthermore, the major problems of identification the requirements are concentrated in NFR, where identification of FR are relatively easier than NFR. This is because the user's recitation for NFR is often unclear, ambiguous or hidden in functional requirements. Permanently, NFR that are identified by requirements engineers and users manually based on their experience.

In this study, we propose an automated approach to identify and classify NFR from unconstrained requirement documents. We use syntactic and semantic analysis to extract features from requirement sentences. Machine learning approaches are used to classify the requirement sentences into five NFR categories.

Online website was developed in order to ease the process of manual labeling the requirement sentences that had been extracted from requirement documents by a group of experts in software engineering. Furthermore, NLP techniques were used to represent the requirement sentences features syntactically and semantically in numeric forms. This task is a prerequisite process for automatic classification. Two main types of NLP techniques were adopted to extract features from requirement sentences. The first one is statistical vectorization methods such as term frequency (TF) and term frequency inverse document frequency (TF-IDF). The second is word embedding methods include Google Word2Vec and BERT, which are common distributed semantic models for better word representation based on big data. These methods allow words with similar semantic meaning to have similar representation. And to classify the requirement sentences we use four different ML approaches, Naive Base (NB), Support Vector Machines (SVM), Logistic Regression (LR), and Convolutional Neural Network (CNN). In this study we also presents a new method to classify NFR by combined four different NLP techniques into one fusion model to exploit the best of the four extraction techniques together in one model.

The rest of this paper is structured as follows. Section 2 discusses related work in the field of requirements classification. Section 3 provides complete details about the research methodology. Section 4, presents the experiment and results. Chapter5, discuss the experiment results. Finally, section 6 provides conclusion and future works.

II. LITERATURE REVIEW

In the last few years, there have been a lot of interest and studies concern in developing new approaches to classify software requirements automatically. Various learning methods have been used, including rule-based methods, machine learning methods, genetic methods, deep learning and various hybrid approaches.

This section elucidates various recent studies directions observed in requirements classification. Rule-based approaches were an interesting topic to classify requirements in the past decade as they performed well for specific dataset. When the size of requirements documents increased and the software fields have expanded, this issue has become more complicated. Nowadays, researchers shift towards statistical methods using models generated by machine learning algorithms leaving rule-based methodologies out of the focus of modern research. This section is organized in two subsections as follows : Rule-based approaches and machine learning approaches review.

A. RULE-BASED APPROACHES :

Rule-based approaches mainly classify text into organized groups using linguistic analysis. These rules used to construct a model using syntactic elements of a text to identify pertinent categories based on it is content. Each rule consists of pattern and a predicted category. This approach has been demonstrated by many researchers. Sharma et al [4], proposed a pattern based rule approach in order to parsing the requirements based on NLP. They Suppose presence of a certain combination of words and their relationship are unique for each category of NFR . The researchers defined a domain specific language for software requirements to build textual syntactic pattern identification. The contribution of this work confide to small set of complex rules. And the evaluation results came with percentage recall between 60 and 85 % for five categories of NFR.

A similar approach adopted by Xiao et al [5]. They proposed a linguistic analysis model to parse requirement documents with semantic meaning using semantic pattern matching .They performed a study on 115 sentences from 18 different sources, and 25 applications by IBM. The evaluation results were 86.2% accuracy extraction among open source dataset and 87.5% among IBM applications.

Cleland-Huang et al [6], suggested information retrieval approach to identify and classify NFR. They proposed a classification approach depend on training dataset for specify set of keyword “Indicator term” for each NFR categories. In case the terms are identified and weighted, it can be used in classifying the next sentence. Two levels of indicators has been adopted, top-K terms that indicate all NFR types and all-terms indicator" that indicate each type of NFR. A certain threshold has been adopted for each NFR category, and in case the score did not achieve the assigned threshold, requirements are classified as FR. In this research they obtained 79.9% overall recall and 42.5% precision. They also found some types of NFR performed bad recall up to 40%, In contrast, they achieved good results in usability NFR reaching to 80%.

Another research was done by Hussain et al [2] adopted linguistic knowledge to classify NFR in software requirement document . The researches identified 9 groups of keywords: adjective, adverb, model keywords, etc. Where the frequency of each keyword was incorporated as feature in the main feature list and it was ranked using smoothed and non-smoothed probability measure. They set a threshold for each keyword to attribute it to its specific NFR type. With high percentage recall, the research team found this knowledge can help in classification requirements and increase the quality of requirements .

Knauss et al, proposed a statistical approach using Bayesian statistics to identify and classify security requirements [7]. The researchers used this method to calculate the probability that the requirement is security. They achieved good results in cases where the classifier is applied to the requirements from the same source as it was trained with. But when the model tested on different requirements document from other domain, they achieved poor results. This was expected because of syntactic meaning significantly limited in case it is used with other data-sets.

B. MACHINE LEARNING APPROACHES :

Machine learning approaches have recently been gaining with researchers in text classification due to its adaptability and accuracy for automated text mining. In software requirements a significant number of works has adopted machine learning approaches to identify and classify software requirements. Kurtanovic and Maalej developed a supervised machine learning approach based on syntactic and lexical features [8]. They used Amazon software reviews in order to train the model. They adopted SVM and NB algorithms to identify both FR and NFR. This research found that part of speech tags are the most distinctive features includes with the cardinal number feature. The research obtained recall between 70% - 90% without word feature selection, and precision and recall above 70% using only 2% of feature space.

Slankas and Williams conducted a study to aid analysis more effectively for extracting 14 categories of NFR from unconstrained requirements documents [9]. They used 11 requirements documents from "iTrust and PROMISE" dataset. The study aimed to identify the sentence characteristics that affect on classifier performance. Furthermore, they conducted a comparison between 5 various machine learning classifier to determine which is the best performance to identify NFR. The research found that the sentence characteristics such as lemma, stem and stop words had no little performance effect. They also concluded that word vector representation and SVM classifier performed twice effectively compared to NB. Furthermore,they found that k Nearest Neighbor (KNN) classifier with distance matrix had F1 measure score (precision and recall) of 54 %, while NB classifier had only 32 %.

Similar research proposed by Zhang et al, conduct an empirical study to classify NFRs using SVM classifier with three different NLP index include individual words, multi-word and N-gram processing [10]. They found that individual words index outweigh the N-gram and multi-word in text representation for short NFR sentences. They also made recommendations that the more sample in a category in the data set, the better classification performance.

Vectorization method is one of the common methods used in semantic analysis. For NFR classification, Amasaki and Leelaprute, evaluated the effect of vectorization methods on NFR classification [11]. They suggested five vectorization methods: TF-IDF, Word to Vector (W2V) on both CBOW and Skip gram techniques and document to vector (D2V).The researchers used 4 classifiers in order to prevent favor one of vectorization techniques in case they used one classifier. The experiment used most common classifiers in literature: LR, NB and random forests. To perform their experiment they used Tera-PROMISE repository. This data set contain 635 instances include 370 NFR and 255 FR. The researchers adopted only 4 categories to evaluate vectorization method: operational, performance, security and usability. The research team found that both Doc2Vec vectorization method and SCDV achieved higher performance than traditional methods. Furthermore they found that some NFRs is more difficult to identify than others.

While most studies mainly focus on classification performance measure using precision and recall. Laszlos et al, considered time factor as part of measured performance [?]. They selected 12 classifiers such as SVM, NB, linear kernel, KNN, Extra Trees and Linear logistic regression. The research used TERA-PROMIS data-set. And used data-set that consists of 625 requirements sentences. They found that NB was the best classifier based on execution time, and both precision and recall measurements compared with the rest of the classifiers.

Little research performed a multi label classification method to classify requirements documents. Jiang et al, proposed a fuzzy similarity approach with KNN (FSKNN) to classify multi-label sentence classification. In their methodology, a multi-label text classification propose to find the k nearest neighbors from each training patterns [13]. In another research, Ramadhani et al [14], proposed an automation system of identification of non-functional requirements from the requirement sentence-based classification algorithms. The researchers suggested additional semantic factors to be used with the classification algorithms of FSKNN but in single class label using hipernim and synonym based on WordNet library to automatically classify NFR. The research found that the use of semantic factor with FSKNN improves the performance of Hamming-loss by 21.9% and 43.7% for the accuracy.

Convolution Neural Networks, are most commonly applied to analyzing visual imagery and image recognition. Recently, there has been considerable interest in adopting CNN in NLP. Winkler and Vogelsang proposed an approach that use CNN in classifying requirement specification as requirement and information [15]. The research used 89 requirements specification documents to train the model. The researchers found that the data-set was imbalanced. To solve this problem,they used under sampling techniques after the data set have been shuffled. They apply prepossessing techniques includes tokenization, stemming, lemmatizing and stop word removal.Then the requirements sentences were transformed into vectors using random vectorization methods to be compatible with Neural Network inputs. The research achieved precision of 73% and recall of 89%. They also highlighted that this approach include vulnerabilities such as there is no insight for what it learns, Furthermore it is not clear why these results are produced. This problem is common among neural network Society.

In a similar study, Baker et al proposed a fully connected ANN and CNN approaches to classify NFR [16].But in this researcher they used random vectors to represent requirement sentence as input for CNN. To perform their experiment they adopted only five requirement categories: operability , performance, security and usability. The researchers used common data-sets called (PROMISE) that include 1165 NFR cover 10 categories.The design has been performed in 5 steps: data pre-processing, ANN model construction, CNN model construction and evaluation. The evaluation results of this research achieved precision ranging between 82% and 90% and recall within range between 78% and 85% in ANN model. Where in CNN, they achieved precision between 82% and 94%, and recall between 76% and 97% with high F-score equal 92%.

Dekhtyar and Fong also proposed CNN technique to identification requirements [17]. They adopted Naive Bayes over TF-IDF and Word Count techniques as baseline to compare with CNN approach. The research objective was to classify the requirements into FR and NFR categories. To do this they used SecRec dataset which labeled as security and non-security requirements. And they also used additional requirements from other projects. The researchers implemented a CNN multi-layer feed-forward neural networks using python TensorFlow library, this library uses numerical computation based on data flow graph. The research scored 4.74% higher precision compared to TF-IDF, and 10.17% compared to word count.

Among previous related studies review. We hardly found research direction for classifying NFR using Recurrent Neural Network (RNN). Although it's one of the most common model used in text classification. Abdur-Rahman et al, one of the few researchers suggested deep learning approach using RNN [18]. They performed their research design in three steps: data pre-processing involve removing stop words, special characters, lemmatization and tokenization. Step 2: word vectorization : they used Word2Vec model to convert each word to vector (word embedding). Then they trained three different classifiers :RNN, GRU and LSTM models. In this research, they achieved high precision rate equal 0.961, and 0.967 recall. And they found that RNN is an

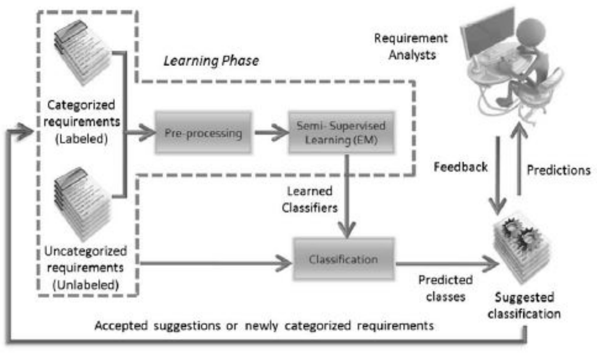

Learning Phase
Requirement Analysts
Categorized requirements (Labeled)
Pre-processing
Semi-Supervised Learning (EM)
Feedback
Predictions
Learned Classifiers
Uncategorized requirements (Unlabeled)
Classification
Predicted classes
Suggested classification
Accepted suggestions or newly categorized requirements

FIGURE 1. Semi-supervised approach for requirement classification [19]

effective approach to classify NFR compared to CNN and GRU approaches.

Automatic NFR classification has well-known limitation because of small number of pre-labeled requirements data-set. This problem is common within researchers who specialize in classifying requirements. One of the researchers who proposed a semi-supervised approach to solve this issue are Casamayor et al [19]. In their research, they reduced the number of labelled requirements using knowledge provided by un-categorized requirements. Through this approach they aimed to reduce the number of instances needed for learning. To achieve this goal, they implemented expectation maximizing strategy based on Bayesian classifiers. Figure 1 describe the proposed scheme they used. Once the initial classifier is ready, it is used to classify other requirements. Where requirements analysis support suggested classification by predict unlabelled NFR. The requirements that have been manually classified are categorized into highly confident requirements. The research concludes that this approach will mitigate the labelling effort by incorporating the manual revision and classification of NFR.

In another aspect, ontology-based adopted to support number approaches to identify and classify requirements engineering. Ontology-based relies heavily on the expressive features of description logic languages. Shah et al proposed a hybrid approach (NFR-Specifier) based on ontology to specify NFR from informal requirements [20]. Their approach architecture starts from pre-processing, ontology formulation, and NFR classification. Ontology formulation model contain generating SRS ontology semi-automatically with rule based approach. They construct a distance similarity measure matrix using word feature (noun,verb, adjective). After that, they group similar requirements into a cluster appointed to specific NFR category. The research concludes that this approach would have a positive impact for requirement engineering.

Two other studies suggested a hybrid approach with ontology formation and NLP to identify and classified requirements. Vlas and Robinson [21] presented an NLP technique aimed to bridge between natural language and formal requirement documents. They used a semi-automated method for discover and classify software requirements from both unconstrained and constrained documents. In their study they develop requirements classifier for natural language (RCNL). RCNL constructs on multi-level ontology, where requirements based lies on upper levels. The lower level are grammar based. RCNL classifier constructed by graphical development tool called "GATE". It contain annotation pattern engine and annotation indexing. Finally, from 61.292 tokens, RCNL recognized 74.3% of those tokens. The rest 25.7% of tokens remain unclassified, This has happened when the classification rule did not correspond with given requirements. The researchers established that RCNL classifier provide an alternative approach, but may be not too much generalized to work with other data-set and needs to be more improved.

In concordance with previous research that classified NFR based on requirements ontology. Rashwan et al [22], proposed an approach adopted SVM to classify requirements sentences into different ontology classes. The researchers annotated manually in total 3064 sentences from PROMISE corpus documents data-set. The sentences were categorized into four main classes: FR, several types of NFR, constraints and others. Documents are prepossessing by tokenizer, splitter, steamer before they are classified using SVM classifier. After that they populate NFR ontology with OWL individuals. This is done through linking the sentences in the requirement documents with the identical classes in the ontology.

While most research proposed supervised ML approach for requirements classification. Few research proposed unsupervised ML approach to handle the same issue. One of them done by Mahmoud and Williams [23]. They suggested a clustering techniques based on systematic analysis to clustering NFR to its various categories, such as security ,usability , reliability and performance. The researcher relied on FR sentences to extract NFR. Where they assumed that FR contain implicitly NFR. for example FR login sentence contains security requirement. The research used wikipedia to find the semantically meaning of the requirements sentences. They adopted three NLP techniques for sentences similarity: Latent semantic analysis (LSA), Co-occurrence and Thesaurus method. These techniques detect the similarity of words semantically. Then the study used partitioning and hierarchical clustering for cohesive words to match class with each NFR word cluster for detecting NFR sentences . In terms of semantic analysis the research found that hierarchical clustering model is more effective than partitioning algorithms. And for similarity semantic analyses. The research found that the encyclopedia "Wikipedia" was more accurate than other methods that relied on dictionaries.

The latest systematic literature review in NFR identification and classification done by Binkhonain and Zhao published in 2019, ELSEVIER journal [24]. They reported systematic review of 24 studies used ML-based approaches for classifying NFR. The objectives of this research lie in three questions: What ML approaches are used in selected studies?, how the algorithms work? and how the ML results have been evaluated. The research found 11 studies used SVM algorithms. While 7 studies used NB algorithms from total of 24 studies. In question two the researcher found across several studies the most NLP techniques used are : stemming, stop words removal, part of speech, tokenization and lematization. In evaluation phase they found that more than 70% of included studies used k-fold cross validation. And for performance measurement techniques that performed to evaluate the results the researchers found that three quarters of studies used precision and recall measurements techniques and 7 of them followed by F-score.

The most important issues that the study concluded is the close collaboration between requirement engineering and ML approaches. Furthermore, the results in the same ML algorithms varies from research to research. Where algorithm performs well in some studies and performed bad in others. Finally ,at the end of this systematic review, the researchers identified three open challenges. First, there is the lack of shared training requirements dataset. Second, no standard definition of NFR, and most of the literature didn't use clear feature identification and selection.

III. RESEARCH METHODOLOGY

The literature review in the previous section highlighted the need for semantic analysis and feature extraction techniques to identify and classify NFR. Furthermore, we highlighted the failures of the rule-based approach to achieve satisfying results with different dataset. This section introduces our methodology that adopted NLP techniques and ML algorithms to identify and classify NFR from unconstrained documents. Figure 2 shows a block diagram that represents our approach employed in this study. In our research we adopted five NFR categories (reliability, performance, security, availability, and usability) that had been identified by IEEE-Std 830-1993 as the most commonly considered NFR in the most domains and software projects.

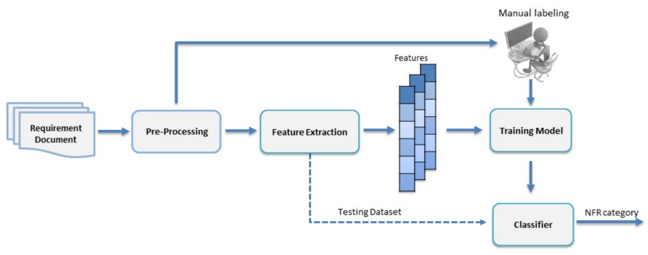


FIGURE 2. Overview of research approach

A. DATASET DESCRIPTION :

In this research we used PURE dataset[1], which contains 79 requirements documents in different forms. It is publicly available on the internet for research use. And described in the article "PURE: A dataset of Public requirements documents" [25]. In this dataset requirements documents had written in natural English language. And it can be used for NLP tasks such as ambiguity detection, identification and requirements categorisation. It contains 34,268 sentences covered multiple domain. The size of documents range from 7 to 288 A4 pages, with an average of 47 pages per document. The construction of the documents was distributed into: structure (S), unconstrained (U) and one statement(O). Most of the documents are combination of unconstrained content and one-statement with about 38% of all documents, and the requirements are represented in one sentence. The documents with uniform formats and the structure documents were 15% of documents. Figure 3 shows the distribution of documents in the PURE dataset.

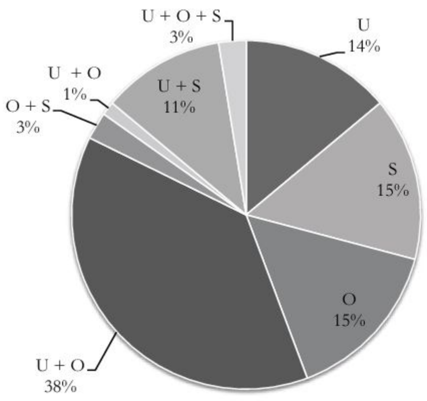


FIGURE 3. PURE dataset distribution [25]

1) PURE dataset manual annotation:

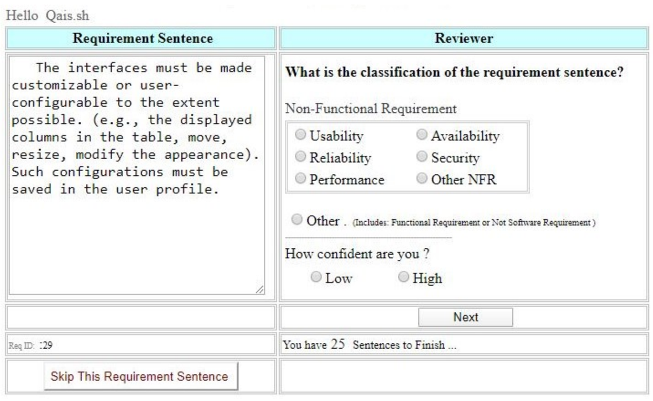
Supervised learning in any ML approach needs a pre-labeled dataset in order to train the models. As mentioned earlier, PURE dataset includes unconstrained requirements documents with unlabeled sentences. In order to conduct supervised learning experiments on this dataset, a sentence-level manual annotation is required. To perform the manual annotation, first of all we extracted the requirements sentences from the documents using a set of common criteria that adopted in extracting sentences from documents. Where the sentences boundary are identified by capital letters and punctuation marks. In order to fulfill this objective, we prepared these documents by parsing all the documents into an XML format. It is worth to mention here that not all the parsed sentences are related to the requirement sentences. But in most cases, in the requirement document, each sentence is talking about one software requirement. Therefore, the annotation (labelling) process performed at sentence level. Then we labeled the extracted sentences manually using a set of procedures. To make it easier and from anywhere accessible, we developed an online website called "Requirements

[1]http://fmt.isti.cnr.it/nlreqdataset/

classifier" [2] for this purpose. We hired a group of experts in software engineering to volunteer in the annotation process. In order to use the online requirement classifier website. First, expert have to register and providing his experience in software engineering by selecting one of three levels based on the years of experience in this field.

Once the volunteers registered on the website. They can login and start annotation process. All of the extracted sentences have been stored in a database, and the system randomly selects a set of sentences for each expert. Each sentence is displayed in a page with a form of options. After the experts read the displayed sentence, they have to decide if the sentence is describing NFR or something else. If the expert finds a sentence which doesn't fit in any of the specified NFR, the other option can be selected. In case the requirement sentence is NFR, the expert has to choose the most appropriate NFR category out of our five target NFR categories:(reliability, availability, usability, performance, and security). As we mentioned before we rely on the NFR definition identified by IEEE-Std 830-1993 [26].

Moreover, the expert has to express his confidence for each sentence by selecting two levels of confidence: low level or high level of confidence for each answer, as its shown in figure 4. We avoided relying on more than two levels of confidence to avoid the neutral choice that most volunteers prefer. Each sentence have to be labeled at least by two different experts to avoid annotation errors. The sub-set of the annotated sentences that have two experts agreed on the same label will be considered in our experiments. To ensure accurate in labeling process, we relied on a sit of criteria to accept each review. First, accepted review should be done at least by two experts. Second, the two experts should have the same answer for each review. If the experts have assigned to two different categories, we choose the review that has the higher confidence. While if the two confidences are equal, the preference is for the reviewer with higher experience. In case there is conflict with same confidence and the reviewers have the same experience, the sentence is excluded.


Hello Qais.sh

Requirement Sentence	Reviewer
The interfaces must be made customizable or user-configurable to the extent possible. (e.g., the displayed columns in the table, move, resize, modify the appearance). Such configurations must be saved in the user profile.	What is the classification of the requirement sentence? Non-Functional Requirement ○ Usability ○ Availability ○ Reliability ○ Security ○ Performance ○ Other NFR ○ Other . (Includes: Functional Requirement or Not Software Requirement) How confident are you ? ○ Low ○ High
	Next
Req ID: :29	You have 25 Sentences to Finish ...
Skip This Requirement Sentence	

FIGURE 4. Requirements manual classification page

[2]http://requirements-bzu.com/

After the annotation task is completed, we got 1846 requirement sentences, labeled by 43 developers and software engineers experts with different levels of experience. The distribution of these requirements categorise are varied as follows : usability : 222, reliability : 62, performance 163, availability : 79, security : 204 and other requirements include functional, constraint and irrelevant sentences: 1119. Figure 5 show the distribution of requirements for 5 types of NFR.

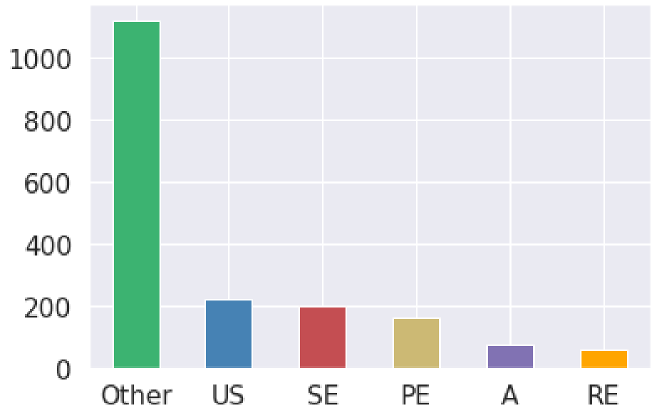


FIGURE 5. PURE dataset

Usually, in any real data-sets, there are always some degrees of imbalance between classes. If the level of imbalance is relatively low there should not be any big impact on ML model performance. In our dataset as shown in the figure 5, there is high degree of imbalance between requirements categories (classes). This issue is common in requirements classification filed. Where the number of NFR sentences always very small compared to FR and other contexts that don't include requirements. Furthermore, the number of NFR categories in the same document are various. This issue led us to use a set of techniques to balance the dataset in order to obtain reliable results from the classification process.

Re-sample technique is one of the most common techniques that is used for balancing text instances.This technique based on both over-sampling for the minority classes and under-sampling for the majority classes. This technique adopted SMOTE strategy which is based on the concept of nearest neighbors to create its synthetic data. SMOTE generates synthetic samples for minority class and introducing synthetic instances. This inherently comes with the issue of creating more of the same data we currently have, without adding any diversity to our dataset. While under sampling techniques achieved by delete percentage number of instances randomly. After we performed balancing task, the dataset has become fairly balanced for each category as its shown in figure 6.

B. THE PROPOSED SYSTEM:

The proposed system consists of three main components, namely: dataset pre-processing, features extraction and ML

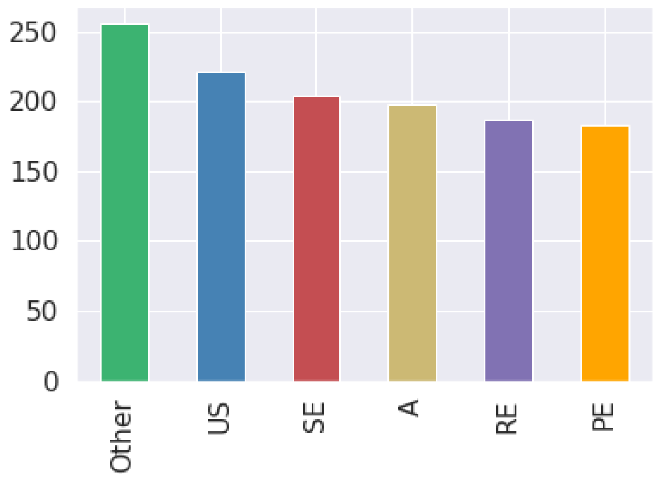


FIGURE 6. PURE dataset after balancing

classification as shown in figure 7. The following subsections describe each of these components:

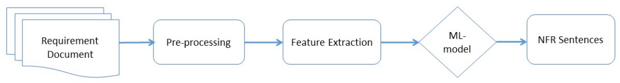


FIGURE 7. System Design:

1) Pre-processing:

Basically, requirements document contains paragraphs, sentences, words, numeric values, punctuation, special character...etc. This document needs to be segmented into smaller tokens for simpler processing and feature extraction. Furthermore, some sentences or paragraphs in these documents are irrelevant to the requirements and need to be excluded from requirement sentences. For this purpose, we performed this task in three steps: tokenization, data cleaning, and normalization, that we described in the following subsequent subsections.

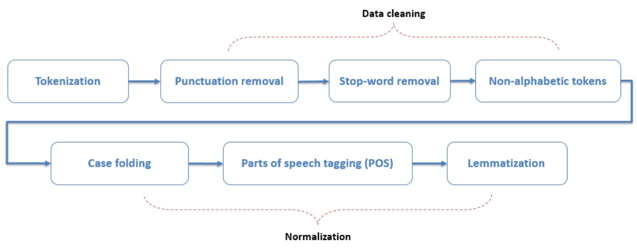


FIGURE 8. Data Pre-processing Tasks

2) Tokenization:

In this process, requirements document is broken up into smaller segments. This process is also called data preparation. The requirements document In this process is broken into paragraphs, and the paragraph into sentences. We relied on a set of criteria to identify the boundary of the sentence involve a capital letter for the start of the sentence and stop marks such as full stop, question mark or an exclamation mark for the end of the sentence. In our experiments, we used sentence tokenization function nltk, which is a python library used to extract English sentences from a document[3]. And each requirement sentence will be chopped up into pieces of terms.

Requirement sentence :

The number of mistakes noted by the students shall be decreased by 50% in the first year.

Tokenized sentence :

['The', 'number', 'of', 'mistakes', 'noted', 'by', 'the', 'students', 'shall', 'be', 'decreased', 'by', '50', '%', 'in', 'the', 'first', 'year', '.']

3) Data cleaning:

Data cleaning is one of the first steps in text pre-processing. It is an important step before the data becomes ready for analysis. In its nature, requirement sentence as most of the natural language texts includes noises that don't provide value in semantic meaning. And in order to achieve better insights and perfect results, it is necessary to have noise-free data.

The output of tokenization process is a set of requirements sentences, which are segmented into tokens. These tokens contain both relevant and irrelevant data such as punctuation, stop words, upper-case, lower-case words, symbols, etc. The objective of data cleaning process is to clean all irrelevant tokens from requirement sentences that may undermine the performance of our model. We accomplished this task in three steps. First step, punctuation removal, in this step all punctuation marks such as stops, question marks, commas, colons, etc are removed from the requirement sentences. Where, the semantic meaning in the text based on the basic words. The second step is stop-word removal, in this step, all high frequency words, such as ('they', 'them', 'their', you, should, from etc) don't add any essential information to the requirement sentence. In our model, we used python library called Natural Language Tool Kit (NLTK). This library contains most of the stop words in English language. The last step of data cleaning task is Non-alphabetic tokens removal that didn't contain useful information.

Tokenized sentence :

['The', 'number', 'of', 'mistakes', 'noted', 'by', 'the', 'students', 'shall', 'be', 'decreased', 'by', '50', '%', 'in', 'the', 'first', 'year', '.']

[3]https://www.nltk.org/api/nltk.tokenize.html

Cleaned sentence :

['number', 'mistakes', 'noted', 'students', 'shall', 'deceased', 'first', 'year']

4) Normalization:

In normalization process, we aimed to convert all the words to a more uniform sequence by transform it to a common base form. In this task, we improve the text modelling and matching. This task is applied on the words level by three steps: case folding, Parts of Speech (POS) tagging and Lemmatization.

Normalized sentence :

['number', 'mistake', 'note', 'student', 'shall', 'decrease', 'first', 'year']

C. FEATURES EXTRACTION (VECTORIZATION) :

The second step in our methodology is to extract representative features from the requirement sentences using a various number of features extraction techniques used in the NLP. In our system we used four vectorization techniques in NLP. Two of them are syntactical based methods: TF and TF-IDF. The other two vectorizatin methods are semantically based methods: Word2Vec[4] and BERT[5]. These methods are the state-of-the-art language representations built on big data of texts corpus.

Using these methods, we transform the requirements sentence into a numerical representation feature in the form of high dimensional vectors which used as input for training machine learning classifiers. This process is also known as vectorization. In this process, requirements sentences properties are extracted in a format supported by machine learning algorithms, and make differences to distinguish it from other requirements categories. This subsection explains in more details how these NLP methods are used to transform the requirements sentences from text to numerical vectors.

1) Term Frequency (TF) :

TF is one of the basic vectorization methods and information retrieval in NLP. It gives indication about the significance of a particular term within the overall requirement documents. In our approach, we use this method to count how many times each word in the requirement sentences appears in all requirement documents and represent it as a vector. To perform TF method, we created words dictionary containing all normalized words in the requirement document. This process also called bag of words (BOW). In this method for each requirement sentence vector was generated in dimension equal to the total number of normalized unique words. The rows corresponds to a requirement sentence and each column represents a unique word. The occurrence number in case the word is exist in the sentence increasing by one. While if the word is not found the feature assigned to Zero. The following example shows how requirement sentence represent as vector using TF method: Req-sentence:The system shall refresh the display every 30 seconds.

After Pre-processing : ['system', 'shall', 'refresh', 'display", 'second']

TABLE 1. Requirement sentence representation in TF

BOW :	access	display	refresh	. . . .	year	shall	system
Vector :	0	1	1	. . . .	0	1	1

Term ordering doesn't be considered in TF method , and the relationship among the words are ignored. This is an obstacle in this method, and it was handled through N-gram technique that improve TF to adopt local ordering when the the vectors generated.

2) Term frequency inverse document frequency (TF-IDF):

In this technique we quantify a word in requirement documents. Weight of each word were computed which signifies of its importance in all requirement documents. This method is widely used in information retrieval in NLP. The weight of the words that occur rarely in the corpus should be scaled up. While, high frequent terms in software document such as 'system', ,'software' , or 'should' need to weight down. Furthermore, removing stop words will also mitigate the effect of frequent the frequent words in the Language that don't add much semantic meaning to the sentence. This methods will improve the basic features that can be extracted from the requirement sentences so that can differentiate between NFR categories. Table 2 shows how the requirement sentence will be represented in TF-IDF:

	access	display	refresh	. . . .	year	shall	system
Vector :	0	0.7512	0.5231	. . . .	0	0.1270	0.3411

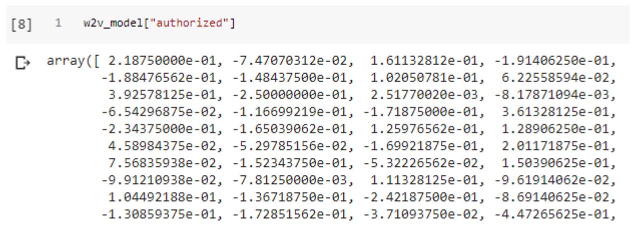
TABLE 2. Requirement sentence representation in TF-IDF

3) Word2Vec :

Word2vec is a common word embedding model provided by Google to improve words representation. This model was trained on nearly 100 billion of words from Google news dataset [27]. In our research, Word2Vec is used to enhance the numeric representation of the words through increase the accuracy of capturing word context from a document in semantic and syntactic words relationship. Each word in the requirement sentences were represented in a vector of 300 dimensions. And each dimension represents one feature encoded from millions of words. The value of each feature in the word representation ranging from zero to one. Figure 9 shows how the word "authorized" is represented in a vector using Word2Vec model. The objective of using this model in this study is to invest the affect of semantic representation for requirement sentences using big data model to achieve high accuracy in NFR classification.

[4]https://code.google.com/archive/p/word2vec/

[5]https://ai.googleblog.com/2018/11/open-sourcing-bert-state-of-art-pre.html



```
[8]  w2v_model["authorized"]

array([ 2.18750000e-01, -7.47070312e-02,  1.61132812e-01, -1.91406250e-01,
       -1.88476562e-01, -1.48437500e-01,  1.02050781e-01,  6.22558594e-02,
        3.92578125e-01, -2.50000000e-01,  2.51770020e-03, -8.17871094e-03,
       -6.54296875e-02, -1.16699219e-01, -1.71875000e-01,  3.61328125e-01,
       -2.34375000e-01, -1.65039062e-01,  1.25976562e-01,  1.28906250e-01,
        4.58984375e-02, -5.29785156e-02, -1.69921875e-01,  2.01171875e-01,
        7.56835938e-02, -1.52343750e-01, -5.32226562e-02,  1.50390625e-01,
       -9.91210938e-02, -7.81250000e-03,  1.11328125e-01, -9.61914062e-02,
        1.04492188e-01, -1.36718750e-01, -2.42187500e-01, -8.69140625e-02,
       -1.30859375e-01, -1.72851562e-01, -3.71093750e-02, -4.47265625e-01,
```

FIGURE 9. Word2Vec vector representation

4) BERT model:

BERT is a text representation technique stands for Bidirectional Encoder Representations from Transformers. BERT is an inflection point in the application of machine learning for NLP and confirmed to be state-of-the-art for a wide range of NLP tasks such semantic analysis and text classification. This breakthrough was the result of Google research in 2018 [28]. BERT is designed to pre-train on two unsupervised tasks, masked language and deep bidirectional representations which have deeper understanding of language on left and right context that overflow single-direction language models. BERT trained on a large volume Wiki Data of 2.5 billion words using two training strategies. Masked LM and Next Sentence Prediction (NSP). In Masked LM fifteen percent of the words in each sentence were replaced with (MASK). Then the model trained to predict the mask words refer to the other context in the trained dataset. While in NSP the model shrink the sentences into two part and trained to predict the second sequence of the sentences. In this study we used BERT model with Masked-LM strategy to represent requirement sentences in semantic numerical vectors. Then, we trained the classifiers on top of the transformer output of the BERT model.

D. ML CLASSIFIERS :

In previous stages of our proposed system, we segmented requirements documents into sentences, then each sentence were converted into a numerical representation in the form of a vector in order to be used by ML models. In this Phase, we built ML models to classify the vectors that represent requirements sentences into our target NFR categories (classes); usability, availability, reliability, security, performance or others. We choose the most common three supervised ML algorithms applied to a similar task: NB, SVM, and LR. NB classifier commonly adopted as baseline in most studies because of its probabilistic model based on the Bayes theorem. While we used SVM and LR due to there results in previous and similar studies in the literature . Furthermore, we used CNN classifier which is considered as the state of the art classifiers that belong to deep learning models. This classifiers enhance features extraction and increase the accuracy of classification compared with the traditional classifiers. In the following subsections we propose how we performed the four ML models that we used in our system:

1) SVM classifier:

SVM is a discriminative classification method which is commonly recognized to be more accurate in NLP as we discussed in the literature review chapter earlier in this thesis [24]. In our system, SVM classifier were used to solve non-linear classification problem using a “kernel trick", which is a method for using a linear classification model to solve a nonlinear problem by projecting the feature vectors of the target classes into a higher dimension in which the classes are linearly separable. Requirement sentences features vectors are mapped to a high dimensional vector space, in which each dimension is linearly separable by decision boundaries which is called (hyperplanes). In our research, we have five classes of NFR and the others class. The traditional SVM classifier is a binary classifier, i.e. can be applied to two classes only. In our case, we have multiple classes (i.e. six classes). To handle this issue one-against-one and one-against-all strategies are used. In order to maximize the margin of the hyperplane, the weight of each feature is minimized using gradient descent algorithm with cost function algorithm.

2) Naive Bayes classifier:

NB classifier is another classifier we adopted in our methodology. This classifier is a probabilistic model based on Bayes theorem. A number of properties in this classifier have prompted us to use it in our NFR classification model. Naive Bayes is one of the most common used supervised ML classifiers [24]. It is widely used to solve NLP classification problem as mentioned in Literature review. NB is demonstrated to be accurate and reliable in natural language classification tasks. Small number of instances is one of the most problematic issues in requirement classification. NB classifier does not require a lot of training data which is one of the issues that led us to choose it in our research.NB classifier needs numerical features as input. In our model. Requirement sentences were transformed into vectors using feature extraction techniques (TF, TF-IDF, W2vec, and BERT). After that, we use Bayes theorem to find the probability of each requirement sentence to which category belongs using our training dataset.

3) Logistic regression classifier :

Logistic regression deals with discrete classes using the natural logarithm. It transforms its output using the logistic sigmoid function to return a probability value which can then be mapped to two or more discrete classes [29]. We used NLP techniques to represent requirement sentences in suitable form. In case of TF and TF-IDF, each requirement sentence is represented into one vector. This makes it suitable as input for LR. In the case of W2V and BERT model requirement sentence is represented in multi dimension vectors, one vector for each word in the sentence. Thus, we have to convert the multiple vectors to one vector for each sentence using mean value for all vectors represented from requirement sentence.

4) Convolution Neural Network (CNN):

CNN algorithm is commonly applied for analyzing image classification. CNN takes an input image as 3 dimensional array based on the image resolution . The height and the width of the image represented 2 dimensions of the array. While the third dimension is the color of the pixel (RGB). In our model we apply CNN model to identify and classify requirement sentences. The sentences are segmented into words. Each word is converted to vectors using the four feature extraction techniques. The TF and TF-IDF techniques convert the requirement sentences to random vectors, where in the case of the Word2Vec and BERT, the sentences are converted into vectors with adopting the semantic meaning. These vectors pass through three layers in the CNN as shown in figure 10.The following section describes how these layers are performed.

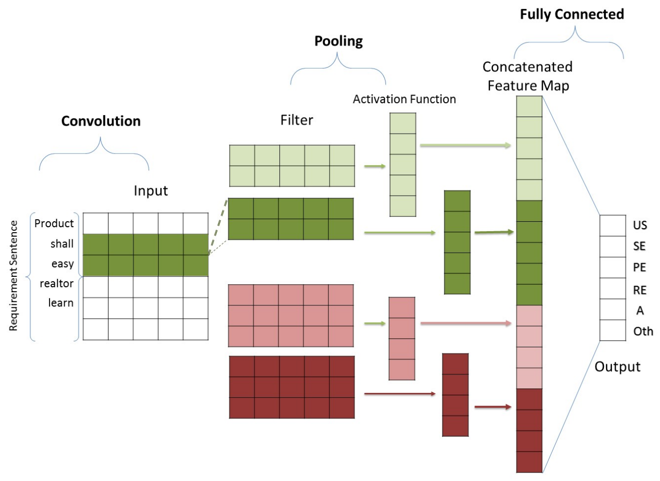


FIGURE 10. CNN architecture for sentence classification

Convolution layer: The input layer of the Convolution Layer is 2 dimensions other than what is common in case of image recognition. The x dimension represents the vector of each word. Where, the Y dimension represents the words in each sentence. Figure 3 shows the representation vectors for the requirement sentence: " The product shall be easy for a relater to learn." in Word2Vec model.

	Vector				
website	1.266 e-03	-1.718 e-01	...	2.812 e-01	6.494 e-02
shall	5.273 e-02	-2.246 e-02	...	3.437 e-01	2.832 e-01
achieve	-2.275 e-01	9.4726 e-02	...	-2.812 e-01	6.738 e-02
time	-4.736 e-02	-4.687 e-02	...	8.251 e-02	1.245 e-01

TABLE 3. Sentence representation in Word2Vec model

The size of the x dimension for CNN input is equal to x-dimension of the vector that represented by NLP method. The size of y-dimension for CNN input is equal to the total number of words for longest requirement sentences in the dataset that we used. The x-dimension of convolution layer in case of W2V and BERT model equal to 300, which is equal to the vectors that are represented by these models for each word.

Feature map (filter): In our model, we perform three sizes of feature map; two,three and four y-dimension with full x-dimension depend on the input layer. This method, somewhat, like bi-grams, tri-grams and 4-grams that used in text mining and NLP tasks.The purpose of this layer is to select new features from the words combination. The y-dimension for both filters are the same in the convolutional Layer, which is equal to the total number of words in the dictionary. While, the size of x-dimension using word2vec and BERT methods are 300 as mentioned earlier.

Stride: Stride: is the number of shift pixel (step) that the filter move over the input matrix. In our model, the filter is moving vertically (y-axis) only. Because the width of the filter represents a single word and dividing this vector is useless. The stride number for vertical move is one. So, the feature map moves in Y-demotion one step at a time until reaches the last word.

Fully connected layer: The function of a fully connected layer is the last layer used to classify values from features extracted to final classes (our five NFR categories and the others). Fully connected layer takes the output of convolution and pooling layers and transform them to classify the input requirement sentence into its types (Usability, Security, Availability,..etc). After pooling layer, we convert our matrix into one vector and feed it into a fully connected layer like normal neural network. The resulting vector is then multiplied by weights and pass through an activation function. Soft-max activation function, which is common function used in deep learning, is used in our case. After that we forward the vector to the output layer, Where, each neuron represents a label belongs to one of the target NFR categories and the others.

E. FUSION MODELS

In the previous sections, we introduced different feature extraction techniques. Each technique has its own properties and advantages. In this paper we propose another approach to achieve better results based on assumption that each of features extraction technique has its own advantages and can provide an addition value if we combine different features together. The idea of this approach is to combine the four NLP techniques in on fused model. The objective of this model is to exploit all of the good features from all NLP methods in one combined module. In this model each NLP technique has distinctive features. For example, TF method characterized by syntactical analysis of requirements sentence. W2V models characterized by the criteria of semantic meaning of the requirement sentence based on pre-trained big data model. TF-IDF method characterized by giving weight to keywords in requirements sentences based on all documents. Finally, BERT model is bidirectional encoder representations.To achieve our goal, we combine all subsystems that use different NLP methods into one overall system to get the best classification result.

Since the CNN is the most recent and successful approach, We combine four CNN classifiers trained on the four different features into one model. CNN classifiers was chosen due to its results superiority over the rest of the other classifiers. The output scores for all CNN models was combined into one vector for each sentence that are used training a front-end classifier. logistic regression uses the natural logarithm to give wight for each feature and transform its output using the logistic sigmoid function to return a probability value which can then be mapped to two or more discrete classes. Other classification algorithms were used for this task like NB and SVM, but logistic regression achieved best results for this mission. Figure 11 shows the architecture of proposed combined model. In front-end, each classifier produces one vector contains score probability for each NFR category. The four produced scores vectors are combined into one vector and used to train and evaluate the back-end classifier.

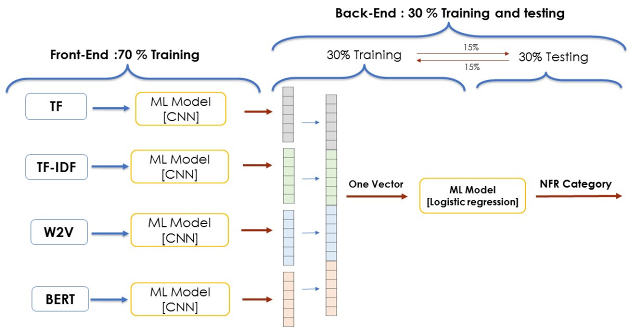


FIGURE 11. Fusion model architecture

F. EVALUATION :

In our methodology, the system evaluation is achieved by randomly splitting dataset into two subsets; train and test. The training set is used for training ML classifiers, while test set is used only for testing the performance of the classifiers. It is worth saying the test dataset has never been used in training (holdout dataset).

A general thumb rule that we followed is to use 70:30 train/test split. Which is the common rule that deal with small size dataset. The total number of requirement instances are 1247. The splitting process split 872 instances for training, and 375 instances for testing. In fusion model, the dataset was split as in all experiments 70:30. Training dataset was used to train front-end classifiers. The test set was used for both training and testing the back-end classifier. The test set was divided into two equal non-overlapped subsets. One subset was used to train the backend classifier and the second was used to evaluate it. Then, the two subsets were exchanged and the same experiment is repeated (2-fold cross-validation). The overall system performance is reported based on the results of the two experiments.

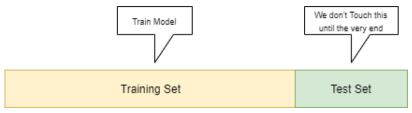


FIGURE 12. Train/Test Method

The classification performance metrics were used in all presented experiments are accuracy, precision, recall and F1-score [24]. Precision measures the percentage of the number of correctly classified requirements to the total number of true positive and false positive prediction (positive prediction value). In other words, the percentage of retrieved NFR that are relevant, where high precision relates to the low false positive rate. This measure called type 1 error. Equation 1 defined precision measurement. TP "true positive" denoted the number of correct classified requirements. FP "false positive" denoted the number of incorrect classified.this is called type 1 error.

$$precision = \frac{TP}{TP + FP} \tag{1}$$

Recall denotes the percentage of relevant NFR that related successfully. This is also called type 2 error. The importance of both measurements (precision and recall) depend on the objective behind the measurement.In this research our object is to identity most NFR without losing a number NFR that may be necessary in early steps of SDLC. Equation 2 shows the formulation of recall

$$recall = \frac{TP}{TP + FN} \tag{2}$$

F-measure is the weighted harmonic average of precision and recall. Therefore, F1-score takes both false positives and false negatives in calculation.F1-score can be formulated as:

$$F - measure = \frac{2.precising \times recall}{precising + recall} \tag{3}$$

Statistical test was used in all experiments to evaluate statistically the results of repeated experiments. We used popular non-parametric test named Wilcoxon statistical test. Wilcoxon compares two paired groups, and most of previous studies used it in case the experiment be run many times [30] [31]. The goal of the test is to determine if two or more sets of pairs are different from one another in a statistically significant manner. In this thesis we compared between four classification algorithms using different feature extraction techniques and evaluated through Wilcoxon statistical test establish if they are statistically significantly different from one another and the results are real and not caused by luck or chance.

IV. EXPERIMENTS AND RESULTS

Recall that this study aims to investigate the effectiveness of NLP techniques and ML approaches for NFR classification from unconstrained requirements documents. In order to achieve this goal, we performed various experiments using

PURE dataset which were described in the previous section. In this section, we present our experiments and their results.

In order to investigate the effectiveness of the most commonly used ML classifiers, we conducted a set of experiments using the following four ML techniques:

1) Naiev Base (NB).
2) Support Vector Machine (SVM).
3) Logistic Regression (LR).
4) Convectional Neural Network (CNN).

Also, to study the effectiveness of the most common feature extraction techniques, the following four common feature extraction techniques were used in our experiments with each ML techniques.

1) Term Frequency (TF).
2) Term frequency inverse document frequency(TF-IDF).
3) Word2Vec model (W2V).
4) Bidirectional Encoder Representations from Transformers (BERT).

The main strategy in our study is to use the four NLP techniques for mapping requirement sentences into numeric vectors then train and evaluate the four ML classifiers, each on the four vectors types. By this, we conducted 4 experiments.

V. ENVIRONMENT SETUP:

To perform all of our experiments, we used Google Colab[6] cloud service which supports GPU processor. It is Jupyter notebook environment that runs entirely in the cloud. Table 4 shows detailed specifications of the processing capability of Colab cloud service.

TABLE 4. Environment setup

	Type	Specification
1	CPU model	2 * Intel(R) Xeon(R) CPU @ 2.00GHz
2	CPU MHz	2000.168
3	cache size	39424 KB
4	Ram	32 GB
5	SSD	69 GB
6	GPU	Tesla K80
7	OS	Ubuntu 18.04.3
8	Environment	Cloud service : Google Colab
9	Pro-Language	Python 3.7

Python programming language was used for developing the systems and conducting all the experiments. It contains massive number of frameworks and libraries for NLP techniques and data pre-processing modules [32]. Python, with its rich technology stack, has an extensive set of libraries for artificial intelligence and machine learning.

VI. PRE-PROCESSING :

First of all, we applied number of pre-processing steps in order to clean and prepare text data into a form that is predictable and analyzable for our experiments. In all of our experiments, we performed the pre-processing task through three steps: tokenization, data cleaning and normalization.

A. TOKENIZATION :

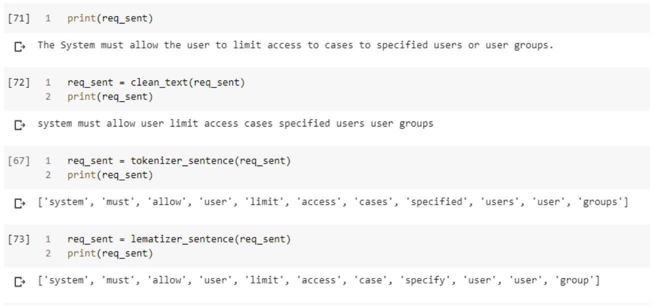
Requirement sentences were extracted from requirement document using python Texttract package[7]. This package is used to extract content from any type of file without any irrelevant markup. Then all sentences were broken up into small chunk using python platform used to work with human language data called natural language tool kit (Python NLTK). This module includes tokenizer package to divide strings into lists of sub-strings (tokens) based on white space and punctuation.

B. DATA CLEANING :

In this step all irrelevant data are removed including punctuation, stop-words and non alphabetic tokens. We also used python NLTK package to remove English stop-word and stripped library to remove punctuation and non alphabetic tokens.

C. NORMALIZATION :

In normalization pre-processing step, a number of related tasks was done meant to put all the words into a more uniform sequence. This process consists of three steps : case folding, part of speech tagging and lemmatization. In this mission we used python WordNet, which is a large lexical database contain sets of cognitive synonyms in English language. The example below describes pre-processing task including normalization step.



```
[71]  print(req_sent)

The System must allow the user to limit access to cases to specified users or user groups.

[72]  req_sent = clean_text(req_sent)
      print(req_sent)

system must allow user limit access cases specified users user groups

[67]  req_sent = tokenizer_sentence(req_sent)
      print(req_sent)

['system', 'must', 'allow', 'user', 'limit', 'access', 'cases', 'specified', 'users', 'user', 'groups']

[73]  req_sent = lematizer_sentence(req_sent)
      print(req_sent)

['system', 'must', 'allow', 'user', 'limit', 'access', 'case', 'specify', 'user', 'user', 'group']
```

FIGURE 13. Pre-Processing Tasks in python

VII. FEATURES EXTRACTION :

Multiple NLP techniques were used to extract features from the requirements sentences. Furthermore, NumPy Python package were used to represent the dataset in ndarray data structure. NumPy is the fundamental python package needed for scientific computation, which supports large, multi-dimensional arrays and matrices.

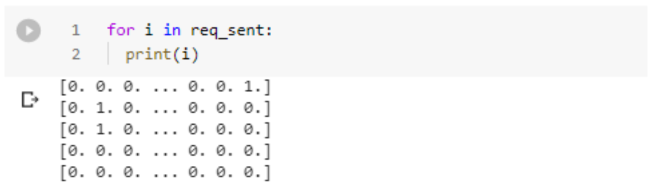
1) TF vectorization method :

To represent requirement sentence in TF method, dictionary of unique words in the data set (BOW) were generated using bow_generate function. The size of this dictionary was 1247 that equal the total number of unique words in all requirement

[6]https://colab.research.google.com/

[7]https://textract.readthedocs.io/en/stable/

sentences. This number also expresses the dimension size of each requirement sentence represented by TF in the experiments. For each word exist in the requirement sentence, one is added to its corresponding on it's vector.



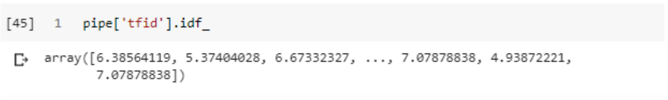
```
for i in req_sent:
  print(i)
```

```
[0. 0. 0. ... 0. 0. 1.]
[0. 1. 0. ... 0. 0. 0.]
[0. 1. 0. ... 0. 0. 0.]
[0. 0. 0. ... 0. 0. 0.]
[0. 0. 0. ... 0. 0. 0.]
```

FIGURE 14. Requirement sentence representation in TF

2) TF-IDF vectorization method :

TF-IDF is distinguished from TF method in words weighting, where the weight of each word is calculated the signifies of its importance in all requirement documents that we adopted. To do that, Tfidf transformer class were used from sklearn[8] feature extraction library to transform requirement sentences to TF-IDF vectors. Each requirement sentence represented by vector in 1247 features, which represent the total number of unique words in all requirement document. Figure 15, shows how TF-IDF method calculate the weight of each word in requirement sentence. The words that are frequently repeated will have little weights such as the word "system" as its shown in yellow color.



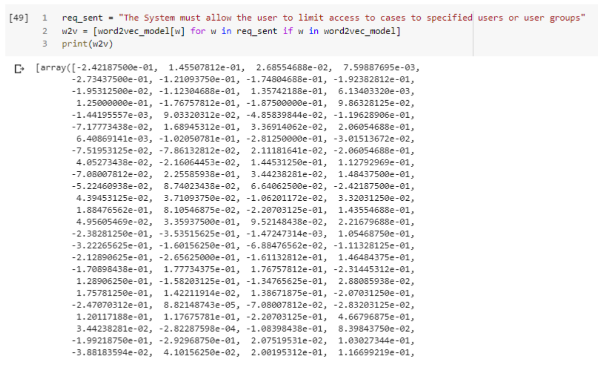
```
[45]  pipe['tfid'].idf_
```

```
array([6.38564119, 5.37404028, 6.67332327, ..., 7.07878838, 4.93872221,
       7.07878838])
```

FIGURE 15. Requirement sentence representation in TF-IDF

3) W2V vectorization method :

Word2Vec is one of the most popular google model to produce word embedding. In this model, each word represented by vector with 300 dimension. In this experiment the words in requirement sentences convert to 300 dimensional vector. Thus each requirement sentences will be represented in 2 dimensional vectors. In traditional ML such as NB, LR and SVM that deal with one dimensional feature vector, we calculate the mean value for overall words vectors in each requirement sentence. So, one vector contain 300 features will represent each requirement sentence. While in CNN classifier that deal with 2 dimensional vectors requirement sentences represented by 2 dimension numpy array. Where x-dimension represent each word vector. And y-dimension represent the words that make up requirement sentences. Padding techniques are required to unifies the dimension of numpy 2d array. The y-dimension for each requirement sentences set to 50, which is equal to the longest requirement sentence in dataset after pre-processing task. Pad_sequences package where used from tensorflow keras library[9] to handle the various lengths of requirement sentences. Gensim libraries[10] also used to load W2V model which is an open-source library implemented in Python for topic modeling and NLP tasks. Figure 16 shows how requirement sentences represented in this model.



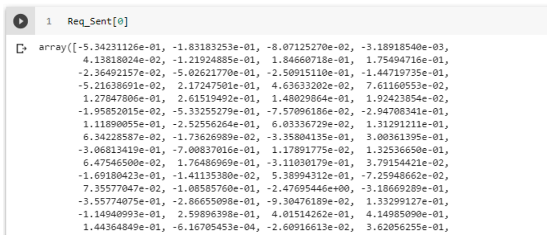
```
[49]  req_sent = "The System must allow the user to limit access to cases to specified users or user groups"
      w2v = [word2vec_model[w] for w in req_sent if w in word2vec_model]
      print(w2v)
```

```
[array([-2.42187500e-01,  1.45507812e-01,  2.68554688e-02,  7.59887695e-03,
       -2.73437500e-01, -1.21093750e-01, -1.74804688e-01, -1.92382812e-01,
       -1.95312500e-02, -1.12304688e-01,  1.35742188e-01,  6.13403320e-03,
        1.25000000e-01, -1.76757812e-01, -1.87500000e-01,  9.86328125e-02,
       -1.44195557e-03,  9.03320312e-02, -4.85839844e-02, -1.19628906e-01,
       -7.17773438e-02,  1.68945312e-01,  3.36914062e-02,  2.06054688e-01,
        6.40869141e-03, -1.02050781e-01, -2.81250000e-01, -3.01513672e-02,
       -7.51953125e-02, -7.86132812e-02,  2.11181641e-02, -2.06054688e-01,
        4.05273438e-02, -2.16064453e-02,  1.44531250e-01,  1.12792969e-01,
       -7.08007812e-02,  2.25585938e-01,  3.44238281e-02,  1.48437500e-01,
       -5.22460938e-02,  8.74023438e-02,  6.64062500e-02, -2.42187500e-01,
        4.39453125e-02,  3.71093750e-02, -1.06201172e-02,  3.32031250e-02,
        1.88476562e-01,  8.10546875e-02, -2.20703125e-01,  1.43554688e-01,
        4.95605469e-02,  3.35937500e-01,  9.52148438e-02,  2.21679688e-01,
       -2.38281250e-01, -3.53515625e-01, -1.47247314e-03,  1.05468750e-01,
       -3.22265625e-01, -1.60156250e-01, -6.88476562e-02, -1.11328125e-01,
       -2.12890625e-01, -2.65625000e-01, -1.61132812e-01,  1.46484375e-01,
       -1.70898438e-01,  1.77734375e-01,  1.76757812e-01, -2.31445312e-01,
        1.28906250e-01, -1.58203125e-01, -1.34765625e-01,  2.88085938e-02,
        1.75781250e-01,  1.42211914e-02,  1.38671875e-01, -2.07031250e-01,
       -2.47070312e-01,  8.82148743e-05, -7.08007812e-02, -2.83203125e-02,
        1.20117188e-01,  1.17675781e-01, -2.20703125e-01,  4.66796875e-01,
        3.44238281e-02, -2.82287598e-04, -1.08398438e-01,  8.39843750e-02,
       -1.99218750e-01, -2.92968750e-01,  2.07519531e-02,  1.03027344e-01,
       -3.88183594e-02,  4.10156250e-02,  2.00195312e-01,  1.16699219e-01,
```

FIGURE 16. Requirement sentence representation in W2V model

4) BERT Model :

In this experiment we used ktrain python library that contain BERT-Base pre-trained models on tensorflow [33] using 12-layer, 768-hidden, 12-heads and 110M parameters. As W2V model, BERT transform the words into 300 embedding vectors. And each requirement sentence was represented by 2d numpy array. In traditional ML the 2d numpy array reshaped into 1d features vector using mean value for all words in requirement sentence. In CNN model requirenebt sentences represented by 2d numpy array which is compatible with convolution layer. Padding technique is also required to unifies the dimension of numpy 2d array. The size of padding in our experiment was 50 dimension, which equal to the maximum number of words in requirements sentences in our dataset. Figure 17 shows how requirement sentences represented in this model.



```
Req_Sent[0]
```

```
array([-5.34231126e-01, -1.83183253e-01, -8.07125270e-02, -3.18918540e-03,
        4.13818024e-02, -1.21924885e-01,  1.84660718e-01,  1.75494716e-01,
       -2.36492157e-02, -5.02621770e-01, -2.50915110e-01, -1.44719735e-01,
       -5.21638691e-02,  2.17247501e-01,  4.63633202e-02,  7.61160553e-02,
        1.27847806e-01,  2.61519492e-01,  1.48029864e-01,  1.92423854e-02,
       -1.95852015e-02, -5.33255279e-01, -7.57096186e-02, -2.94708341e-01,
        1.11890055e-01, -2.52556264e-01,  6.03336729e-02,  1.31291211e-01,
        6.34228587e-02, -1.73626989e-02, -3.35804135e-01,  3.00361395e-01,
       -3.06813419e-01, -7.00837016e-01,  1.17891775e-02,  1.32536650e-01,
        6.47546500e-02,  1.76486969e-01, -3.11030179e-01,  3.79154421e-02,
       -1.69180423e-01, -1.41135380e-02,  5.38994312e-01, -7.25948662e-02,
        7.35577047e-02, -1.08585760e-01, -2.47695446e+00, -3.18669289e-01,
       -3.55774075e-01, -2.86655098e-01, -9.30476189e-02,  1.33299127e-01,
       -1.14940993e-01,  2.59896398e-01,  4.01514262e-01,  4.14985090e-01,
        1.44364849e-01, -6.16705453e-04, -2.60916613e-02,  3.62056255e-01,
```

FIGURE 17. Requirement sentence representation in BERT

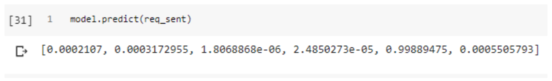
Finally, the classifier predict based on its experience to which category the requirement sentence belongs using softmax activation function. Figure 18 shows an array represents

[8]https://scikit-learn.org

[9]https://www.tensorflow.org

[10]https://pypi.org/project/gensim/

how much percentage the requirement sentence belong to each NFR category: US, RE, A, SE and Other.



```
[31] 1   model.predict(req_sent)

[0.0002107, 0.0003172955, 1.8068868e-06, 2.4850273e-05, 0.99889475, 0.0005505793]
```

FIGURE 18. Output layer for CNN model

A. PARAMETERS SETTINGS FOR ML CLASSIFIERS :

With different NLP techniques we fixed the classifiers parameters to evaluate the effect of NLP techniques on classification results. We adopted the default parameters sitting as it is in Scikit-learn library. Which is the most common free machine learning library for Python and matlab language. In this subsection we list all parameters for each classifiers that were used in our experiments.

1) Naive bayes :

Scikit-learn with Gaussian NB package was used to implement NB classifiers. The setup parameters for this classifiers was : [variance smoothing = 1e-9, Number of class = 6, Max-iter = 100]

2) Support vector machines :

For SVM classifiers, we used scikit-learn with SVM package. The followings parameters was used in our experiments for this classifier: [Regularization =True, kernel='rbf', coef0=0.0]

3) Logistic regression :

In LR classifiers, we used linear model library with Logistic-Regression package. The setup parameters for LR experiments : [Tolerance = 0.001,fit intercept=True, intercept-scaling=1, class-weight=None, random-state=None, Max-iter = 100, Regularization =True]

4) Convolution neural network :

For all CNN based experiments, we used keras tensorflow library which is open-source neural-network library written in Python. Embedding dimension parameter was fixed to 300, that equal to word vector dimension generated from embeding model. And number of output layer were fixed to the number requirement categories target classes, which equal to 6. While we tuned the number of filters until the best result was converged. And filter size was fixed to bigrams,trigrams and fourgram. The reset of parameters were fixed as its defaults: [Conv layer = 3, pooling-Layer = 3, Embedding-DIM=300, NB-FILTERS = 200, FFN-UNITS = 512, NB-classes = 6,Stride = 1, Filter-sizes = [2,3,4], DROPOUT_RATE =0.05, BATCH-SIZE = 32, NB-EPOCHS = 20, regularizers = l2(0.01), activation-output= sof-Max,training-Options = 'adam']

VIII. EXPERIMENT 1: OPTIMAL ML CLASSIFIER USING TF METHOD

The Objective of this experiment is to identify the optimal ML classifier using TF method. The four described ML approaches were adopted including the three traditional approaches NB, SVM, logistic regression and deep learning approach CNN. Each experiment was repeated 10 separate times to verify and avoid the randomness of results. Statistical test was also performed to assess the validation of results.

For the traditional ML approaches, the requirements sentences are fed to the classifier using one vector that represents the count of all words in requirement sentence. Each item in the list represents one feature, and the type of requirement sentence represents the label. In CNN, the output vector of TF method is fed to the classifier vertically as an image. Figure 19 shows the accuracy results for the 4 classifiers in 10 runs. The median value in box plot figure represent in orange line inside each box, while the green triangle represent the mean of accuracy results for the 10 runs.

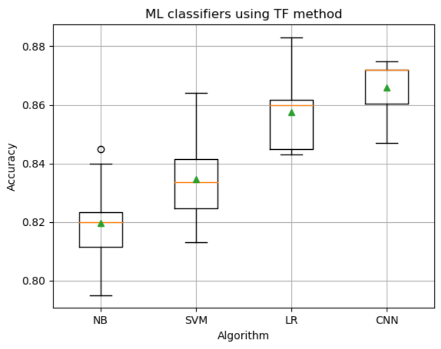


FIGURE 19. ML classifiers accuracy using TF method

We also report three performance metrics, precision, recall and F1-score metrics for each NFR type using Sklearn libraries. The results for each NFR types are shown in table 5.

A. EXPERIMENT 2: OPTIMAL ML CLASSIFIER USING TF-IDF METHOD

In this experiment, we aim to determine the optimal ML classifier using TF-IDF method. The TF-IDF feature vectors representing all sentences extracted from the training dataset are used to train the four ML classifiers. The TF-IDF features extracted from the testing dataset are used to evaluate each system. Each experiment was repeated 10 times. The accuracy results for the four classifiers are shown in figure 20.

Table 6 also shows three performance metrics that used in classifiers for each NFR types that we adopted in this paper.

B. EXPERIMENT 3: OPTIMAL ML CLASSIFIER USING W2V MODEL

In this experiment, the Google W2C model was used to transform the words in requirement sentences into numerical

TABLE 5. ML performance metrics Using TF method

	Naive Bayes			SVM			Logistic Regression			CNN		
	Presesion	Recall	F1-score	Presesion	Recall	F1-score	Presesion	Recall	F1-score	Presesion	Recall	F1-score
US	0.789	0.843	0.843	0.793	0.852	0.819	0.838	0.866	0.850	0.820	0.876	0.846
RE	0.934	0.831	0.831	0.898	0.874	0.884	0.917	0.860	0.886	0.942	0.881	0.910
PE	0.847	0.739	0.739	0.810	0.883	0.844	0.806	0.880	0.840	0.847	0.870	0.858
A	0.951	0.917	0.917	0.896	0.960	0.926	0.949	0.865	0.904	0.939	0.929	0.933
SE	0.703	0.744	0.744	0.728	0.836	0.777	0.762	0.817	0.787	0.826	0.829	0.825
Oth	0.736	0.848	0.848	0.893	0.709	0.788	0.866	0.859	0.861	0.838	0.825	0.829
Avg	**0.827**	**0.820**	**0.820**	**0.836**	**0.852**	**0.840**	**0.856**	**0.858**	**0.855**	**0.869**	**0.868**	**0.867**

TABLE 6. ML performance metrics using TF-IDF method

	Naive Bayes			SVM			Logistic Regression			CNN		
	Presesion	Recall	F1-score	Presesion	Recall	F1-score	Presesion	Recall	F1-score	Presesion	Recall	F1-score
US	0.823	0.867	0.842	0.830	0.907	0.863	0.814	0.818	0.814	0.842	0.867	0.853
RE	0.848	0.878	0.862	0.905	0.822	0.861	0.916	0.836	0.874	0.926	0.849	0.885
PE	0.819	0.845	0.831	0.847	0.867	0.862	0.822	0.891	0.852	0.967	0.921	0.943
A	0.900	0.823	0.858	0.901	0.901	0.921	0.879	0.938	0.907	0.828	0.897	0.860
SE	0.761	0.782	0.770	0.806	0.870	0.836	0.773	0.901	0.831	0.806	0.832	0.818
Oth	0.877	0.834	0.854	0.906	0.851	0.865	0.917	0.786	0.845	0.859	0.866	0.860
Avg	**0.838**	**0.838**	**0.836**	**0.866**	**0.870**	**0.868**	**0.854**	**0.862**	**0.854**	**0.871**	**0.872**	**0.870**

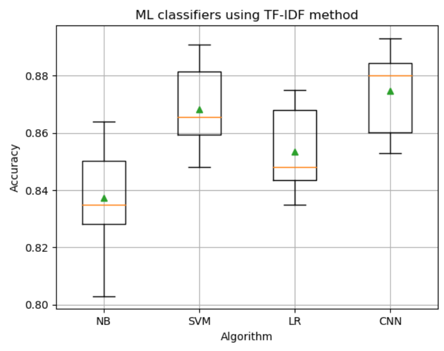


FIGURE 20. ML classifiers accuracy using TF-IDF method

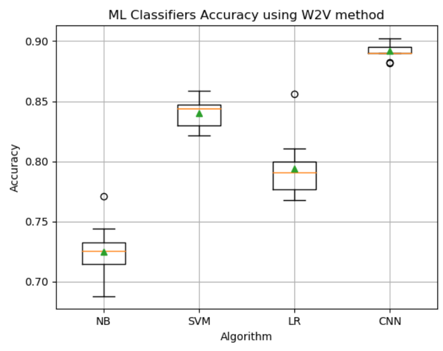


FIGURE 21. ML classifiers accuracy using W2V model

victors semantically. For the three traditional classifiers that we used, a one-dimensional vector for each sentence is computed as the arithmetic mean of all words vectors that consists the sentence. The resulted sentence-level vectors extracted from the training datatset are used to train the classifiers, and the vectors extracted from the testing dataset to test the classifiers.

For the CNN classifier, the word-level vectors of each sentence are stacked into two-dimensional array, similar to the image, and used in training and evaluation. Similar to the previous experiments, each experiment is repeated 10 times. Figure 21 shows the classification mean and median accuracy in 10 runs for each ML classifiers.

C. EXPERIMENT 4: OPTIMAL ML CLASSIFIER USING BERT MODEL

In this experiment we used Google Bert model, this model is considered as the state of the art language model for text representation. Similar to W2V, BERT model represents each word in 300-dimensional vector. For the three traditional classifiers,as W2V one sentence-level vector is computed by taking the mean of all word vectors. The resulting vectors of the training dataset are used to train the NB, SVM, and RF classifiers. The sentence-level vectors extracted from the testing dataset are used to evaluate each of the three classifiers. For the CNN classifier, the word-level BERT vectors are stacked into two-dimensional array and used to train and evaluate the CNN-based classifier. The accuracy of ML classification for the 10 runs was reported through box plot, as its shown in figure 22.

Furthermore, we present three performance metrics for each NFR types using the BERT representation model. Table 8 shows the precision, recall and F1 score for each NFR category.

TABLE 7. ML performance metrics using W2V model

	Naive Bayes			SVM			Logistic Regression			CNN		
	Presesion	Recall	F1-score	Presesion	Recall	F1-score	Presesion	Recall	F1-score	Presesion	Recall	F1-score
US	0.644	0.731	0.682	0.792	0.792	0.822	0.756	0.825	0.788	0.853	0.901	0.877
RE	0.767	0.676	0.718	0.904	0.904	0.852	0.812	0.787	0.797	0.879	0.810	0.843
PE	0.752	0.667	0.704	0.843	0.843	0.863	0.819	0.801	0.808	0.950	0.891	0.919
A	0.726	0.930	0.814	0.882	0.882	0.894	0.813	0.861	0.834	0.845	0.938	0.889
SE	0.719	0.662	0.687	0.758	0.758	0.789	0.748	0.767	0.754	0.891	0.854	0.872
Oth	0.751	0.742	0.745	0.862	0.862	0.821	0.829	0.749	0.787	0.969	0.969	0.969
Avg	**0.727**	**0.735**	**0.725**	**0.840**	**0.845**	**0.840**	**0.796**	**0.798**	**0.794**	**0.898**	**0.894**	**0.895**

TABLE 8. ML performance metrics using BERT model

	Naive Bayes			SVM			Logistic Regression			CNN		
	Presesion	Recall	F1-score	Presesion	Recall	F1-score	Presesion	Recall	F1-score	Presesion	Recall	F1-score
US	0.621	0.501	0.553	0.647	0.731	0.682	0.778	0.790	0.782	0.910	0.964	0.933
RE	0.609	0.559	0.581	0.678	0.652	0.660	0.884	0.769	0.820	0.962	0.941	0.950
PE	0.575	0.577	0.572	0.577	0.722	0.638	0.797	0.833	0.814	0.853	0.913	0.880
A	0.689	0.651	0.668	0.814	0.799	0.805	0.900	0.881	0.890	0.959	0.891	0.923
SE	0.593	0.706	0.641	0.674	0.706	0.687	0.738	0.801	0.766	0.986	0.836	0.903
Oth	0.536	0.652	0.587	0.765	0.617	0.681	0.779	0.792	0.785	0.863	0.914	0.915
Avg	**0.604**	**0.607**	**0.600**	**0.693**	**0.705**	**0.692**	**0.813**	**0.811**	**0.810**	**0.922**	**0.914**	**0.915**

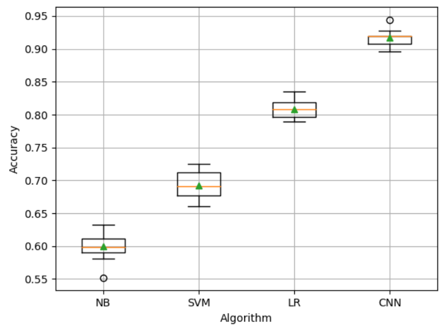


FIGURE 22. ML classifiers accuracy using BERT model

D. NFR CLASSIFICATION ACCURACY USING DIFFERENT NLP TECHNIQUES :

After we performed the four experiments we summarize the mean of the accuracy results for all classifiers using the used NLP feature extraction methods, as shown in figure 23.

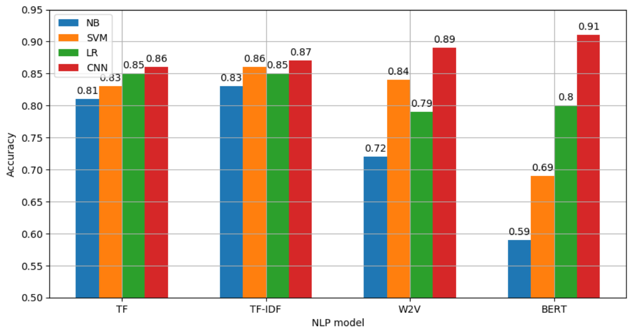


FIGURE 23. ML classifiers with all NLP methods

E. OPTIMAL NLP TECHNIQUES TO TRANSFORM NFR USING CNN:

As noted from the results of the previous experiments, CNN approach achieves the best accuracy in NFR classification with all NLP feature extraction methods. In figure 24, we compare the accuracy of CNN classifiers using the four NLP feature vectors.

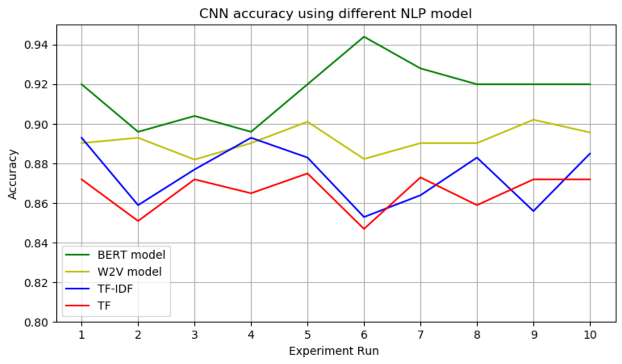


FIGURE 24. Optimal NLP techniques using CNN classifiers

F. EXPERIMENT 5 : FUSION MODEL

The objective of this experiment is to exploit the best of the four features extraction techniques together. In previous experiments CNN classifiers outperforms the other classifiers with all extraction techniques. In this experiment, we present a system consisting of two classification stages: front-end and back-end. In front-end part four CNN classifiers are trained using the four different features extraction techniques. Each classifier is configured to produce a probability score for each NLP category (the category with the highest score is the recognized one), i.e. each classifier produces a 5-dimensional scores vectors for each testing sentence. The score vectors of the four classifiers are concatenated together into one 20-dimensional vector. The produced concatenated vectors of the all testing sentences are divided into equal subsets. At back-end, a logistic regression classifier is trained on one subset and evaluated on the second one. The two subsets are exchanged and the same experiment is repeated. The evaluation result of the back-end classifier using the two testing subsets are used to find the overall fused system performance.

It may be recalled that, the computation time in this experiment takes four times the time of the previous experiment, where we ran the model with four different extraction techniques, in addition to back-end model that takes less than 3 second. But mainly, the training process is a one-time process, then the model is ready to use.

TABLE 9. Performance metrics report for fusion model

	Fusion Model		
	Presesion	Recall	F1-score
US	0.95	0.98	0.96
RE	0.97	0.91	0.94
PE	1.00	0.92	0.96
A	1.00	0.98	0.99
SE	0.85	0.96	0.91
Oth	0.94	0.91	0.92
Avg	**0.95**	**0.95**	**0.94**

In this experiment we achieved accuracy 94.6% using the first subset of testing data. And 94.1% accuracy for the second subset. In average, we achieved accuracy of 94.3%. Table 5.6 shows the classification performance matrices for fusion system.

By this results, our fusion system outperforms all of the previous systems, with an accuracy improvement by 2.4% compared to the best results achieved by the BERT model. We also performed a comparison between the CNN classifiers using the four NLP methods that we adopted, in addition to our fusion model. Figure 25 shows how the accuracy for the BERT model outperforms the statistical-based feature extraction techniques with the CNN classifier, and how the fusion improves the BERT-based system.

To analyze the results obtained from fusion model we formulated a confusion matrix, shown in figure 26 which describes the true positive,true negative, false positive and false negative for the fusion system results, since it gives the best result.

G. STATISTICAL TEST :

In this study, each classification experiment is executed 10 times to estimate the variability of the results and to evaluation how close to each other. Furthermore, to increase the accuracy of the estimate assuming that no bias or systematic error is present. We compared the obtained results using Wilcoxon statistical test to establish if they are statistically significantly different from one another and the results are

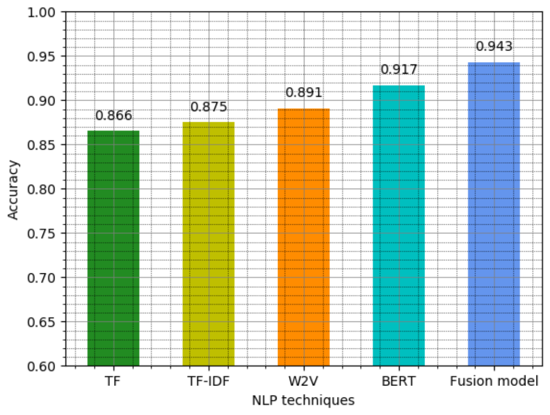


FIGURE 25. CNN classification accuracy using NLP techniques and fusion model

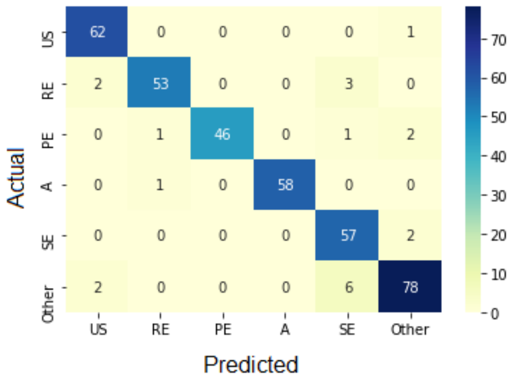


FIGURE 26. Confusion matrix for fusion model results

real and not caused by luck or chance. Wilcoxon signed-rank test is non-parametric statistical hypothesis test used to compare the mean and median of the values. To do that, first we found the mean with standard deviation for the 10 runs, see table 10. Then, we found the median and Interquartile range for each classifiers in the 10 runs, see table 11. Then we calculated a wilcoxon statistic test between each classifiers.

Figure 12 represents the wilcoxon results compared two paired groups of classifiers results. Triangle symbols (▽ , ▲) indicate that results are statistically significance in which p-value < 5%. The inverted white triangle ▽ represents the superiority of the algorithm at the top of the table statistically over the algorithm from the side of the table, while the black triangle ▲ represents the opposite relation. The dash line (-) indicate that there are no statistically significance between the two classifiers, p-value > 5%.

IX. DISCUSSION

In this section we discuss the experiments results presented in the previous section.

The objective of the first four experiments is to investigate the efficiency of machine learning techniques to classify

TABLE 10. Mean and Standard Deviation of Accuracy indicator.

	NB	SVM	LR	CNN
BERT	$5.99e-01_{2.31e-02}$	$6.93e-01_{2.15e-02}$	$8.08e-01_{1.51e-02}$	$9.17e-01_{1.47e-02}$
TF	$8.20e-01_{1.47e-02}$	$8.35e-01_{1.50e-02}$	$8.57e-01_{1.36e-02}$	$8.66e-01_{1.01e-02}$
TF-IDF	$8.37e-01_{1.71e-02}$	$8.68e-01_{1.45e-02}$	$8.53e-01_{1.47e-02}$	$8.75e-01_{1.53e-02}$
W2V	$7.25e-01_{2.26e-02}$	$8.40e-01_{1.24e-02}$	$7.94e-01_{2.54e-02}$	$8.92e-01_{6.70e-03}$

TABLE 11. Median and Interquartile Range of the Accuracy indicator.

	NB	SVM	LR	CNN
BERT	$5.99e-01_{2.12e-02}$	$6.91e-01_{3.53e-02}$	$8.08e-01_{2.27e-02}$	$9.20e-01_{1.20e-02}$
TF	$8.20e-01_{1.18e-02}$	$8.34e-01_{1.67e-02}$	$8.60e-01_{1.68e-02}$	$8.72e-01_{1.15e-02}$
TF-IDF	$8.35e-01_{2.20e-02}$	$8.66e-01_{2.20e-02}$	$8.48e-01_{2.45e-02}$	$8.80e-01_{2.42e-02}$
W2V	$7.25e-01_{1.80e-02}$	$8.44e-01_{1.73e-02}$	$7.91e-01_{2.34e-02}$	$8.90e-01_{4.73e-03}$

TABLE 12. Wilcoxon values of the accuracy indicator (TF, TF-IDF, W2V, BERT).

	SVM	LR	CNN
NB	▽ ▽ ▽ ▽	▽ ▽ ▽ ▽	▽ ▽ ▽ ▽
SVM		▽ ▲ ▲ ▽	▽ – ▽ ▽
LR			– ▽ ▽ ▽

NFR using three common ML classifiers and CNN. The traditional ML approaches: SVM, NB, and LR achieved relatively good results with the NLP techniques that adopted syntactic analysis such as TF and TF-IDF compared with the semantic-based embedding models such as W2V and BERT. NB classifier achieved accuracy range from 60.3 % to 83.7%. SVM classifier achieved accuracy in range between 69% to 86.8%. While CNN outperformed all the traditional approaches by achieving results range from 85% to 92%. By analyzing the results of all experiments, obviously traditional ML approaches outperform word embedding methods in syntactic analysis methods. The CNN approach is more successful with the word embedding models, due to the ability of this model to handle multi dimensions inputs vectors in the convolution layer such as word2vec and BERT model. We can also conclude that NFR classification accuracy mainly depends on two main factors: first: the type of classifier, Second: the NLP extraction techniques that is used.

The results of the measurement metrics: precision, recall and F1-score are close to the accuracy results in the most cases, except for the security requirement class, as shown in tables 22 and 21. Its noted that the systems precision for detecting the security requirement class was low in the most traditional ML classifiers in range between 59.3% to 80.2% . While, CNN classifier was able to improve detecting security class with a precision range between 82% to 98% as shown in table 22. In the average precision (positive rate) for all classes in NB, SVM and LR, the classifiers achieved results between 60% to 80%. While, CNN classifier was able to achieve precision between 80% to 90%.

The sensitivity of ML models was measured using recall metric. Recall measures the fraction of the total amount of relevant instances that actually retrieved. In other word, recall measures the ratio between the correctly identified as positive NFR to the actual positive NFR. This metric also expresses the ability to find all relevant requirement categories. CNN classifier achieved best recall results in range between 86% to 91.4%. This means out of every 100 NFR, the system succeeded in extracting 91 NFR. In experiment 5, we used fusion model to employ all NLP techniques to extract features from requirement sentence. We achieved weighted recall equal 94%. This results can minimize the number of unrevealed NFR from requirement document. By these results, the number of extracted NFR increased from the presented requirement document. And this will help the software engineer to define most NFR that the customer has expressed through the requirement document.

Also, we investigated the effectiveness of the NLP techniques in representing the NFR categories in the unconstrained requirement documents. Two experiments were performed for this investigation. First, we used 4 different NLP techniques with each ML classifier, including statistical vectorization methods [TF and TF-IDF], and word embedding models [W2V and BERT]. The results in figure 23 shows how natural language techniques affect the classification results. And while some NLP techniques do well with some types of ML classifiers, the others achieved lower accuracy with other types of classifiers. For example, BERT model achieved the highest accuracy with CNN classifier equal to 91.4 %, while it achieved low results with traditional ML classifiers such as NB and SVM. In contrast, the traditional ML classifiers such as NB, SVM and LR achieved better results using random vectorization methods than word embedding methods. Another result can be extracted from figure 5.9, TF method achieved the lowest results in each classifier. Anf the TF-IDF representation method achieved good results ranging between 83 % and 87%. All of this leads us to one conclusion: random vectorization methods achieve better results with traditional ML (SVM,NB and LR) that have lower number of feature dimensions. Whereas, word embedding methods that have high dimensional representation achieved better results with deep learning approach (CNN) which have rich information retrieval for requirement sentences, such as word2vec and BERT model.

|Fusion model combines the CNN-based systems with the four feature extraction techniques. Two of these methods represent the syntactic features from requirement sentences

such as TF and TF-IDF. While the rest such as W2V and BERT, represent semantic features. The results show that fusion system outperforms all the individual proposed systems, with an accuracy of 94.3% (i.e. 2.4% improvement). It achieves precision in range between 85% to 100%, and recall in range between 91% to 98%. Table 5.6 presents the classification results of the fusion model. And figure 5.13 shows a comparison between the CNN classifiers using the four feature types, in addition to the fusion model. The results show how the accuracy of the semantic-based BERT features outperform the statistical-based features with the CNN classifiers, and how the fused system improves the accuracy of CNN-BERT system by 2.4%.

Figure 26 shows the confusion matrix of the classification results for the fusion model, colored as a heat map. The correct classifications (true positive) are depicted on the diagonal, and have been highlighted through dark blue color. For example the matrix shows that from 63 usability requirements sentences the fusion model categorized correctly 62 requirement sentences as usability requirement sentences, and categorized one as other incorrectly (false negative). By looking at the first column (usability predicted), we see that 2 reliability sentences and other requirement sentences incorrectly classified as usability (false positive). It can also be noted that there are 10 NFR sentences classified as security, where they are actually related to the other categories, such as reliability, performance and others. This means that there is a confusion between the security and the other categories. We also noticed that fusion model can classify correctly 58 availability NFR categories from 59 actual sentences. While achieved 53 reliability from 58 actual sentences. We also find that the usability and the availability achieved the best sensitivity value. From 63 usability requirements sentences, the fusion model was able to categorize correctly 62 requirement sentences as usability from 63 usability requirement sentences. And it was able to categorize 58 requirement sentences correctly of the 59 sentences that are actually represent the availability requirement sentences.

Finally, to obtained results are statistically significantly and different from one another we used Wilcoxon statistical test. Table 10 represents the mean and standard deviation for the accuracy in the 10 runs of each experiment. Table 11 also represents the median and interquartile range of the accuracy. The darker shaded level in the two tables represents the better result. This means that the CNN classifier exceeds all the other ML classifiers with all features types. Table 12 represents the results of Wilcoxon statistical test. This table shows that the CNN model outperforms the other classifiers in different NLP techniques in statistically significant manner. The inverted white triangle (▽) represents the superiority of the algorithm on the top of the table statistically over the algorithm from the side of the table, while the black triangle (▲) represents the opposite relation. And the dash line represents that there is no statistical superiority between the two parties. That means the relation between the two classifiers is not statistically significant where p-value higher than 0.05 (> 0.05).

X. CONCLUSIONS AND FUTURE WORK:

Software requirements are mainly captured and documented in human natural language. It is documented in a form called requirements document. In general, requirements document contains both functional and non-functional requirements. Analyzing NFR is tedious and time consuming in SDLC. In this study, we proposed an automated systems to identify and classify five NFR from the unconstrained requirement documents using ML algorithms and NLP techniques. Different NLP techniques were presented in this study, include random and word embedding vectorization methods to feed in different ML classifiers includes traditional approaches and deep learning approaches.

All of the reported experiments were performed using (PURE) public dataset, which consists of 79 unconstrained requirements documents. The requirements sentences in the documents were manually labelled by a group of software experts volunteers. Serious criteria was applied to accept the label of each requirement sentence to verify its authenticity.

A set of experiments are conducted for investigating the effectiveness of the NLP techniques and the effectiveness of the ML techniques for classifying the requirement sentences extracted from unconstrained requirement documents. The traditional well-known NLP techniques such as TF and TF-IDF were used for representing requirement sentences into a numerical feature vectors. Furthermore, we used state-of-the-art word embedding techniques such as Word2vec and BERT. TF and TF-IDF techniques exploit the statistical information of the requirement sentence. Whereas, the embedding methods exploit the semantic information of the sentences.

The traditional ML models based on the statistical NLP techniques achieved precision 87% and recall 86%. Where, the CNN model achieved relatively higher precision of 92% using the state of the art language modeling method BERT. Furthermore, fusing four CNN systems with the four NLP vectorization methods improved the classification accuracy by 2.4%.

From the presented results in this paper, we can draw three primary conclusions. First, CNN approach identifies and classifies NFR efficiently and outperforms other traditional ML approaches such as SVM, NB and logistic regression. Second, word embedding models are more effective than other traditional NLP methods in representing software requirement sentences for NFR classification. Third, NLP techniques have a valuable impact on the NFR classification results, and fusion multiple NLP techniques, significantly improves the classification accuracy.

A. FUTURE WORK

The research of adopted NLP techniques and ML algorithm into requirement classification is still continuing. Two primary direction can be investigated to extend our work. First direction, expanding the number of adopted NFR categories, such as (mutability, solubility, etc..). Second direction, continually investigating the effectiveness of the other ML approaches in classification NFR such as recurrent neural network (RNN) and fusion more models, in addition to using more and different data-sets.

.

REFERENCES

[1] A. Van Lamsweerde, Requirements engineering: From system goals to UML models to software, vol. 10. Chichester, UK: John Wiley , Sons, 2009.

[2] I. Hussain, L. Kosseim, and O. Ormandjieva, “Using Linguistic Knowledge to Classify Non-functional Requirements in SRS documents,” in Natural Language and Information Systems, Berlin, Heidelberg, 2008, pp. 287–298.

[3] A. Tripathy, A. Agrawal, and S. Rath, Requirement Analysis using Natural Language Processing. 2014.

[4] V. S. Sharma, R. R. Ramnani, and S. Sengupta, “A framework for identifying and analyzing non-functional requirements from text,” in Proceedings of the 4th International Workshop on Twin Peaks of Requirements and Architecture, New York, NY, USA, Jun. 2014, pp. 1–8, doi: 10.1145/2593861.2593862.

[5] X. Xiao, A. Paradkar, S. Thummalapenta, and T. Xie, “Automated extraction of security policies from natural-language software documents,” in Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering, 2012, pp. 1–11.

[6] J. Cleland-Huang, R. Settimi, X. Zou and P. Solc, "The Detection and Classification of Non-Functional Requirements with Application to Early Aspects," 14th IEEE International Requirements Engineering Conference (RE’06), Minneapolis/St. Paul, MN, 2006, pp. 39-48, doi: 10.1109/RE.2006.65.

[7] E. Knauss, S. Houmb, K. Schneider, S. Islam, and J. Jürjens, “Supporting Requirements Engineers in Recognising Security Issues,” in Requirements Engineering: Foundation for Software Quality, Berlin, Heidelberg, 2011, pp. 4–18

[8] Z. Kurtanović and W. Maalej, “Automatically Classifying Functional and Non-functional Requirements Using Supervised Machine Learning,” in 2017 IEEE 25th International Requirements Engineering Conference (RE), Sep. 2017, pp. 490–495.

[9] J. Slankas and L. Williams, “Automated extraction of non-functional requirements in available documentation,” in 2013 1st International Workshop on Natural Language Analysis in Software Engineering (NaturaLiSE), May 2013, pp. 9–16.

[10] W. Zhang, Y. Yang, Q. Wang, and F. Shu, “An empirical study on classification of non-functional requirements,” in The twenty-third international conference on software engineering and knowledge engineering (SEKE 2011), 2011, pp. 190–195.

[11] S. Amasaki and P. Leelaprute, "The Effects of Vectorization Methods on Non-Functional Requirements Classification," 2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), Prague, 2018, pp. 175-182, doi: 10.1109/SEAA.2018.00036.

[12] L. Tóth and L. Vidács, “Study of Various Classifiers for Identification and Classification of Non-functional Requirements,” in Computational Science and Its Applications – ICCSA 2018, Cham, 2018, pp. 492–503

[13] J.-Y. Jiang, S.-C. Tsai, and S.-J. Lee, “FSKNN: Multi-label text categorization based on fuzzy similarity and k nearest neighbors,” Expert Systems with Applications, vol. 39, no. 3, pp. 2813–2821, Feb. 2012

[14] D. Ramadhani, S. Rochimah, and U. Yuhana, “Classification of non-functional requirements using Semantic-FSKNN based ISO/IEC 9126,” TELKOMNIKA (Telecommunication Computing Electronics and Control), vol. 13, p. 1456, Dec. 2015

[15] J. Winkler and A. Vogelsang, “Automatic classification of requirements based on convolutional neural networks,” in 2016 IEEE 24th International Requirements Engineering Conference Workshops (REW), 2016, pp. 39–45.

[16] C. Baker, L. Deng, S. Chakraborty and J. Dehlinger, "Automatic Multi-class Non-Functional Software Requirements Classification Using Neural Networks," 2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC), Milwaukee, WI, USA, 2019, pp. 610-615, doi: 10.1109/COMPSAC.2019.10275.

[17] A. Dekhtyar and V. Fong, "RE Data Challenge: Requirements Identification with Word2Vec and TensorFlow," 2017 IEEE 25th International Requirements Engineering Conference (RE), Lisbon, 2017, pp. 484-489, doi: 10.1109/RE.2017.26.

[18] Md. A. Rahman, Md. A. Haque, Md. N. A. Tawhid, and Md. S. Siddik, “Classifying non-functional requirements using RNN variants for quality software development,” in Proceedings of the 3rd ACM SIGSOFT International Workshop on Machine Learning Techniques for Software Quality Evaluation, New York, NY, USA, Aug. 2019, pp. 25–30.

[19] A. Casamayor, D. Godoy, and M. Campo, “Identification of non-functional requirements in textual specifications: A semi-supervised learning approach,” Information and Software Technology, vol. 52, no. 4, pp. 436–445, 2010.

[20] U. Shah, S. Patel, and D. Jinwala, “Specification of Non-Functional Requirements: A Hybrid Approach,” Mar. 2016.

[21] R. Vlas and W. N. Robinson, “A rule-based natural language technique for requirements discovery and classification in open-source software development projects,” in 2011 44th Hawaii International Conference on System Sciences, 2011, pp. 1–10.

[22] A. Rashwan, O. Ormandjieva, and R. Witte, “Ontology-Based Classification of Non-functional Requirements in Software Specifications: A New Corpus and SVM-Based Classifier,” in 2013 IEEE 37th Annual Computer Software and Applications Conference, Jul. 2013, pp. 381–386, doi: 10.1109/COMPSAC.2013.64.

[23] A. Mahmoud and G. Williams, “Detecting, classifying, and tracing non-functional software requirements,” Requirements Eng, vol. 21, no. 3, pp. 357–381, Sep. 2016.

[24] M. Binkhonain and L. Zhao, “A review of machine learning algorithms for identification and classification of non-functional requirements,” Expert Systems with Applications: X, vol. 1, p. 100001, 2019.

[25] A. Ferrari, G. O. Spagnolo, and S. Gnesi, “PURE: A Dataset of Public Requirements Documents,” in 2017 IEEE 25th International Requirements Engineering Conference (RE), Sep. 2017, pp. 502–505.

[26] "IEEE Recommended Practice for Software Requirements Specifications,” IEEE Std 830-1998, pp. 1–40, Oct. 1998

[27] L. Ma and Y. Zhang, “Using Word2Vec to process big text data,” in 2015 IEEE International Conference on Big Data (Big Data), Oct. 2015, pp. 2895–2897, doi: 10.1109/BigData.2015.7364114.

[28] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” arXiv preprint arXiv:1810.04805, 2018.

[29] M. Toussaint, “Introduction to machine learning,” Online Course Notes. July, 2016.

[30] M. Haigh, “Software quality, non-functional software requirements and IT-business alignment,” Software Quality Journal, vol. 18, no. 3, pp. 361–385, 2010.

[31] X. Lian and L. Zhang, “Optimized feature selection towards functional and non-functional requirements in software product lines,” in 2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER), 2015, pp. 191–200.

[32] “NumPy.” https://numpy.org/ (accessed Oct. 25, 2020).

[33] A. S. Maiya, “ktrain: A Low-Code Library for Augmented Machine Learning,” arXiv:2004.10703 [cs], Jul. 2020, Accessed: Oct. 25, 2020. [Online]. Available: http://arxiv.org/abs/2004.10703.

[34] S. Tiwari, S. S. Rathore, and A. Gupta, “Selecting requirement elicitation techniques for software projects,” in 2012 CSI Sixth International Conference on Software Engineering (CONSEG), Sep. 2012, pp. 1–10, doi: 10.1109/CONSEG.2012.6349486.

[35] M. Bilal, A. Gani, M. Haseeb, N. Bashir, and N. Malik, “Risk Assessment Across Life Cycle Phases for Small and Medium Software Projects,” Journal of Engineering Science and Technology, vol. 15, pp. 572–588, Feb. 2020.

QAIS A. SHREDA was born in Nablus city, Palestine, in 1982. He received the B.S. in computer engineering from Al-Quds university. And master degree in software engineering from Birzeit university in 2020. From 2007 to 2010, he was a computer engineer in industrial school. Since 2010, he has been system and network administrator in ministry of education. He worked as a coordinator for a set of projects in artificial intelligence and machine learning Fields. His research interest includes Genetic Algorithms and Engineering Optimization.

ABUALSOUD A. HANANI was born in Beit-Furik, Palestine in 1977. Dr. Abualsoud Hanani received BSc degree in electrical engineering from an-Najah national university, Palestine in 2000, and MSc and PhD degrees in computer engineering from the University of Birmingham, UK in 2008 and 2012 respectively. Dr. Hanani worked more than six years in the information and communication industry in different areas and positions. Dr. Hanani worked for Siemens Information and Communication Technologies for more than four years in the research and development until June 2004. Then he worked for Palestine Telecommunication Company (Paltel) for about two years in the ADSL project. In conjunction with his PhD, he worked as a Research Assistant at the University of Birmingham from January 2009 to December 2011. His PhD thesis was in the field of spoken language and accent recognition. Particularly, recognizing regional British accents and ethnic groups from speech and use this for adapting English speech recognizer. He has more than 24 publications in this field and related fields and his current research interest is on Arabic dialects and Arabic speech recognition. Dr. Hanani has received Google faculty research award in 2015 to study the impact of Arabic dialects on the Arabic speech recognition. This project was one of 122 projects selected from 808 proposals worldwide submitted to Google in 2015. Since January 2012, Dr. Hanani has joined Birzeit University as an assistant professor in the Electrical and Computer Engineering department.