BIRZEIT UNIVERSITY

Faculty of Engineering and Technology
Master of Software Engineering

THESIS

Identifying Non-Functional Requirements From Unconstrained Documents Using Natural Language Processing and Machine Learning Approach

---

Author: Qais Gafer Sharida

Supervisor: Dr. Abualsoud Hanani

September 14, 2020

BIRZEIT UNIVERSITY

Faculty of Engineering and Technology
Master of Software Engineering

Identifying Non-Functional Requirements From Unconstrained Documents
Using Natural Language Processing and Machine Learning Approaches

تحديد المتطلبات غير الوظيفية في الوثائق غير المقيدة باستخدام نهج معالجة اللغة الطبيعية والتعلم الآلي

Committee:

Dr. Abualsoud Hanani

Dr. Yousef Hassouneh

Dr. Ahmed Abusnaina

A thesis submitted in fulfilment of the requirements
for the degree of Maters in Software Engineering

September 14, 2020

Identifying Non-Functional Requirements From Unconstrained Documents Using Natural Language Processing and Machine Learning Approaches

---

Thesis

Author : Qais Sharida

Approved by the thesis committee:

Dr. Abualsoud Hanani : (Chairman of the Committee)

Dr. Yousef Hassouneh : (Member)

Dr. Ahmed Abusnaina : (Member)

Date of Defense:

August 25, 2020

Declaration of Authorship

I, Qais Sharida , declare that this thesis titled, “Identifying Non-Functional Requirements From Unconstrained Documents Using Natural Language Processing and Machine Learning Approaches ” and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a master degree at Birzeit University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Abstract

Requirements Engineering (RE) is the first phase in software development life cycle (SDLC). It plays one of the most important and critical roles in software projects, where all subsequent stages depend on it. Software requirement is usually documented in a form called requirements document. It mainly contains both functional requirements (FR) and Non-functional requirements (NFR). NFR are very significant to describe the properties and constraints of the system. Early identification of NFR is important and has direct impact on the system architecture and initial design decision in the first stages. Practically, NFR requirements are extracted manually from the requirement document. This makes it tedious, time consuming task and prone to various errors. In the literature, there are numerous studies that adopted traditional ML approach to classify NFR. However, majority of the approaches don't investigate features extraction techniques for requirements sentences in unconstrained requirements documents. In this thesis we proposed an automatic approach to identify and classify NFR using semantic and syntactic analysis with Machin learning algorithms from unconstrained documents. We used PURE dataset that consists of 79 unconstrained requirements documents in different forms. The features were extracted from requirement sentences using 4 different NLP methods include statistical and state-of-the-art semantic analysis presented by google through BERT model. And we adopted ML algorithms include: Naïve Base, support vector machine, logistic regression, and deep learning CNN approach to classify NFR to its categories . Our results indicate that CNN model can efficiently

classify NFR by achieving accuracy between 84% and 87% using statistical vectorization method and achieved accuracy between 88% to 92% using word embedding semantic methods. Furthermore, we proposed fusion model for combining all NLP methods into one model with the CNN classifier. The fusion model achieved a classification accuracy of 94%, with 2.4% improvement over the best individual classifier.

المستخلص

هندسة المتطلبات ،هي المرحلة الأولى في دورة حياة تطوير البرمجيات. وتلعب واحد من أهم الأدوار وأكثرها حساسية في مشاريع البرمجيات ، حيث تعتمد عليها جميع المراحل اللاحقة. عادة ما يتم توثيق متطلبات البرنامج في نموذج يسمى وثيقة المتطلبات. تحتوي بشكل أساسي على كل من المتطلبات الوظيفية والمتطلبات غير الوظيفية. تعد المتطلبات الغير وظيفية مهمة للغاية لوصف خصائص وقيود النظام. التعرف المبكرعليها مهم وله تأثير مباشر على بنية النظام وقرار التصميم الأولي في مراحله الأولى. عمليا ، يتم استخراج المتطلبات الغير وظيفية يدويا من وثيقة المتطلبات. هذا يجعلها مهمة مضنية ومضيعة للوقت، وعرضة لكثير من الأخطاء المختلفة.

في الأدبيات السابقة، هناك العديد من الدراسات التي اعتمدت نهج التعلم الالي التقليدي لتصنيف المتطلبات الغير وظيفية. ومع ذلك ، فإن معظم الأساليب لا تحقق في تقنيات استخراج الميزات لعبارات المتطلبات في مستندات المتطلبات الغير منظمة.

في هذه الأطروحة اقترحنا نهجًا آليًا لتحديد وتصنيف المتطلبات الغير وظيفية من المستندات غير المنظمة باستخدام التحليل الدلالي والنحوي مع مناهج التعلم الالي. استخدمنا مجموعة وثائق المتطلبات العامة (PURE) التي تتكون من 79 وثيقة متطلبات غير منظمة بأشكال مختلفة. في نهجنا ، نستخلص الميزات من جمل المتطلبات باستخدام أربع طرق مختلفة في البرمجة اللغوية العصبية تشمل والتحليل الدلالي الأحدث الذي تقدمه جوجل من خلال نموذج (BERT) . واعتمدنا خوارزميات تعلم الالة لتصنيف المتطلبات الغير وظيفية من خلال: المصنف البايزي الساذج (NB)، آلة المتجهات الداعمة (SVM) ، والانحدار اللوجستي (LR)، بالاضافة الى نهج التعلم العميق في الشبكات العصبونية الالتفافية (CNN).

تشير نتائجنا إلى أن نموذج الشبكات العصبونية الالتفافية يمكن أن يصنف الاحتياجات الغير وظيفية بكفاءة من خلال تحقيق دقة ما بين 84 ٪ و 87 ٪ باستخدام طريقة المتجه الإحصائي، وتحقيق الدقة بين 88 ٪ إلى 92 ٪ باستخدام طرق الدمج الدلالي. علاوة على ذلك ، اقترحنا نموذج الانصهار لدمج جميع طرق البرمجة اللغوية العصبية في نموذج واحد مع مصنف الشبكات العصبونية الالتفافية . حقق نموذج الاندماج دقة تصنيف بنسبة 94٪ ، مع تحسن بنسبة 2.4٪ عن أفضل مصنف فردي.

Acknowledgements

Firstly, I would like to express my sincere gratitude to my supervisor Dr. Abualsoud Hanani for the continuous support of my master thesis, for his patience, motivation, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. In addition to my supervisor, I would like to thank the other faculty at my great university, Birzeit University.

My sincere thanks also goes to my colleagues who accompanied me in this educational path. Thanks also for volunteers who helped me with the manual classification process in this thesis.

Finally, I must express my very profound gratitude to my family for providing me with unfailing support and continuous encouragement throughout my years of study. This accomplishment would not have been possible without them. Thank you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Requirement Engineering 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"[49, p 35]. It is documented in a form called requirements document (RD), that is suitable for communication, analysis and subsequent implementation. RE is a crucial phase at the beginning of SDLC, and one of the most important and critical role in systems and software projects. Where 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. [49].

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 be 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 requirements document.

There are a wide range of techniques that are usually used for requirements specification and documentation. Unconstrained document is one of these techniques. Its prose in natural language with no specific rules. This technique has several advantages: there are no limitations in expressiveness on what we can specify 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 [20].

Typically,software requirements distinguish into two types of requirements: Functional Requirements (FR) and Non-Functional Requirements (NFR). This mechanism will help 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 NFL have existed, its categories number estimated to be more than 150 categories. IEEE-Std 830-1993 identify

only 13 NFR, D.Mairiza [9]. identified from 252 NFR types the most commonly considered NFR in most domains and they are (Reliability, Performance, Security, Maintainability, and Usability. In our research we will focus on those NFR in addition to availability that most of projects types probably need.

1.1 Motivation

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 design in the early stages. Undetected ignored requirements until a later stage in software development life cycle would be very costly and greatly affect on customer satisfaction .The arising errors 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 USA (2012).[8].

Early identification of NFR is very important in the evaluation of alternatives architectural and design decisions. System architects need NFR to determine constraints as: scalability, security, reliability, performance, 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. [21].

Requirements document are written in natural language and can be processed as any 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 state-of-the-art NLP techniques

such as bidirectional encoder representations from transformers and W2V 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.

1.2 Problem statement

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, analyse, and classify the requirement sentences manually. Furthermore, the major problems of identification the requirements are concentrated in NFR, where identification of FR are relatively easier than NFR. That 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 thesis, we propose an automated approach to identify and classify NFR. We use syntactic and semantic analysis to extract features from requirement sentences. And we adopt ML approaches to classify NFR sentences to its categories.

1.3 Research objectives

In this study, we aim to achieve four objectives.

1. Preparing a dataset that contain a sufficient number of requirements sentences extracted from unconstrained requirement documents including NFR in various categories, and manually label the requirement sentences to its related categories using a group of software engineers experts.

2. Extract meaning-full features from requirement sentences using statistical analysis and state-of-the-art word embedding models.

3. Classify requirement sentences using ML approaches to its NFR relative categories.

4. Investigate the effect of fuse multiple features that are extracted from different NLP techniques on the performance of NFR classification.

1.4 Overview of research approach

In this thesis we propose automated approach using NLP techniques and machine learning algorithms to identify and classify NFR from unconstrained requirement documents.

Online website were developed in order to manual label requirement sentences that had been extracted from requirement documents depending on a group of volunteer experts in software engineering. Furthermore. In this approach NLP techniques were used to represent the requirement sentences features syntactically and semantically in numeric forms, this task is a prerequisite process for machine learning classifiers. Two main NLP techniques were adopted to extract features from requirement sentences. The first one are random vectorization methods such as TF and TF-IDF. The second is word embedding methods include Word2Vec and BERT, which are two 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 in order to identify and classify NFR we adopted traditional ML approaches includes naive base(NB), support vector machines(SVM) and logistic regression (LR) in addition to deep learning algorithm through convolutional

neural network (CNN). Furthermore, we performed an approach to combined multi NLP techniques in one fusion model, to enhance features extraction from requirement sentence.

1.5 Research Questions

By this research, we aim to present answers to the following research questions:

RQ 1: How well natural language processing techniques can identify effectively NFR from unconstrained requirements documents?

RQ 2: How well machine learning algorithms can automatically classify the five NFR categories efficiently, based on IEEE-Std 830-1993 standard?

RQ 3: How well the proposed system performance can improve by fusing different NLP features together into one system?

1.6 Structure of thesis

The rest of this thesis is structured as follows. Chapter 2 introduces background in software requirements engineering, NLP and machine learning. Chapter 3 discusses related work in the field of requirements classification. It mainly discusses the studies objectives, extraction techniques, and algorithms employed to classify requirements. Chapter 4 provides complete details about the research methodology. Chapter 5, presents the experiment and results. Chapter6, discuss the experiment results. Finally, chapter 7 provides conclusion, future works and threats to validity.

Chapter 2

Background

2.1 Software requirements :

Software requirements are description of services that the software system must provide and the constraints under which it must operate. IEEE standard glossary defines a term requirement in software engineering as "A condition or capability needed by a user to solve a problem or achieve an objective" [12, p 62]. The software requirements are mainly divided into two types, FR and NFR.This mechanism will help to understanding the common characteristics of different types of needs.

2.1.1 Functional requirement:

FR is a description of the service that the software must perform. It defines the feature of the system or its component. In other word, FR are features that allow the system to function as it was intended. If the functional requirements are not met, customer satisfaction will not be achieved and the system will fail.[6]

2.1.2 Non-Functional requirements:

NFR is a requirements that define the quality attribute of a software system and describe how the system should work . NFR are often called "quality attributes. Since the early days of software engineering NFL have existed, its categories number estimated to be more than 252 categories. From those requirements IEEE-Std 830-1993 identify only 13 main NFRs to be included in a software requirements document. NFR can be divided into two main categories: 1. Execution qualities, such as safety and usability, which are realizable at run time. 2.Evolution qualities, such as scalability and maintainability which are associated with in the structure of the system. [10]

2.2 Natural language processing

Natural language processing (NLP) is a branch of Artificial Intelligence (AI) lying between linguistics and computer science, it helps machines to handle natural human language. Unlike human beings, computers can only understand numbers. And this make NLP is a difficult issue to solve. The main objective of NLP is representing words in a numeric format that is understandable by the computers. Often, NLP techniques rely on machine learning to derive meaning from human languages. NLP isn't a new science. Recently, the interest in this field has increased significantly with the increase in interest in human contact with computers. This coincided with a revolution of big data and improved algorithms. NLP extract natural language rules using algorithms into a form that the computer can understand. In this section we will discuss the main common techniques that used in NLP tasks. [48]

2.2.1 Syntactic analysis

Syntax is the grammatical structure of the text. Whereas syntactic analysis analyzing natural language with the rules of a formal grammar with assigning a semantic structure to text. It involves determining the subject and predicate and the place of nouns, verbs, pronouns, etc. With reference to language dictionaries the computer can recognize the part of speech for the words and would be able to produce a structural description for the sentence through reading word by word. Basically, syntactic analysis involved many tasks such as tokonization, parts of speech tagging (POS) and lemmatization [35].

2.2.2 Semantic analysis

Semantic analysis is the process of relating syntactic structure from the levels of phrases, clauses, sentences and paragraphs to the level of the writing . The purpose of semantic analysis is to draw the exact meaning (dictionary meaning) from the text and begin with the relationship between individual words [35]

2.2.3 Text Pre-processing

Text pre-processing is practice for cleaning and preparing text data into a form that is predictable and analyzable for different tasks. Text pre-processing include fundamental techniques to make the text data more usable with computer. The most common techniques involved data preparation, non-alphanumeric data, tokenization , stemming , lemmatization, and normalization. [35]

2.2.3.1 Tokenization

Tokenization is one of the most common tasks when it comes to working with text data. In this process, the text is broken up into smaller chunks or segments,it also called segmentation. In this process the document is broken into paragraphs, then the paragraphs divided into sentences. In English the sentence is generally defined as "a word or a group of words that expresses thorough idea by giving a statement order", or asking a question, or exclaiming [40]. There are many criteria to extract sentences in English language . For example the sentence can be identified the boundary with a capital letter and ends with stop marks such as full stop, question mark or an exclamation mark.[35]

2.2.3.2 Punctuation removal

On the sentence level, punctuation such as stops, question marks, commas, colons, etc, does not help in the syntactic field [24]. In this process all punctuation is removed. Where the semantic meaning of sentences is based on the basic words in the sentences.

2.2.3.3 Stop-word removal:

In any language there are many words frequently repeated without adding any essential meaning. In English language stop words do not contribute to the context or content of textual documents such as :"they, you, have,should etc" [4, p 2].

2.2.3.4 Non-alphabetic tokens:

Natural language text may contain non-alphabetic tokens such as date, numbers, symbols, etc. In text pre-processing non-alphabetic tokens is deleted.[35]

2.2.3.5 Normalization:

Normalization is a process of converting all the words to a more uniform sequence by transforming it to a common base form. The normalization process improves the text modelling and the text matching. This task work on the words level. [8]

2.2.3.6 Case folding:

In this step, all letters will be changed to lowercase . Case folding is one of the most common tasks in NLP for the purposes of similarity check.This task allows words such as ‘Machine’ in the beginning of a sentence to match the word ‘machine’ in other objects.[8]

2.2.3.7 Parts of speech tagging (POS):

It is a process of converting a segmented sentence to a list of tuples (word, tag). The tag indicates whether the word is "noun", "verb","adjective", "adverb".. etc. For example, if the word is a noun the tuple is (word , n). This process is required to lemmatization step in text pre-processing.[8]

2.2.3.8 Lemmatization:

Lemmatization usually takes into consideration with morphological analysis of words . The objective of this process is to reduce inflectional forms and sometimes derivationally related forms of a word to a common base form which is known as lemma, in which different inflected forms of a word are grouping together so they represent as a single item[4].

2.2.4 Features extraction (vectorization) :

Features extraction in NLP is a technique for extract the words as features from text. Feature extraction include minimize the number of resources required to describe the data. It is a vital requirement step before any ML classification process. This process is also known as vectorization. In this process sentences properties are extracted in a format supported by machine learning algorithms. Various methods are used to extract features from text, include random methods and word embedding methods. All this methods aimed to convert text features into numerical features.

2.2.4.1 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 document or sentences. This method count how many times each word in the sentence appears in all documents and represent it as vector form. To perform TF, words dictionary should be created that contain all normalized words in the document. This process called bag of words (BOW). In this method for each sentence vector will be generated in the document in dimension equal the total number of normalized words which is equal to BOW size. The rows corresponds to a sentences and each column represents a unique word. The occurrence number in case the word is exist in the sentence equal one. And the if the word not found is zero. [1]

Term ordering doesn't considered in TF method , and the relationship among the words are ignored. This is an obstacle in NLP. N-gram is a technique to improve TF to adopt local ordering when we generate vectors to avoid the disadvantages of un-ordered words. In N-gram we look at token pairs, triplets, or

different combinations . One gram stands for traditional TF where we take one word in each dimension. Bigram stands for a token pair and the words in trigram. In our model we will use both bigram and trigram to handle the ordering issues in TF method.

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

TF-IDF is shorthand for two parts, "term frequency" and "document frequency". The term TF counts how many times each word in the sentence appears in the documents as we describe. Where the weight of the words don't be considered. TF only count the number of times that words appear in a given document. [1]

IDF part found to solve this issue by systematically weight the words. The weight of frequent words are calculated across all documents. The weight of the words that occur rarely in the corpus should be scaled up. While, high frequent terms such as 'I', ,'and' ,'He' 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 . Equation 2.1 , shows how the weight of each word will be computed. tf : term frequency , df: document frequency.

$$W_{i,j} = tf_{i,j} \times log(\frac{N}{df_{i,j}}) \tag{2.1}$$

2.2.4.3 Word embedding:

Word Embedding is a form of word representation. It allows words with similar meaning to be understood by machine learning algorithms semantically. Technically, word embedding is a mapping of words into vectors of real numbers to

bridges the human understanding of language to that of a machine. In word embedding billions of words distributed semantically in vector space model. For example, the vector of "vegetables" will be placed far away from vector representation of "books".While vector of word "King" will be close to the vector of word "Queen". Several method currently exist in word embedding, the most well known methods for producing word embedding models are Word2Vec , Global Vectors and BERT model . And all of them have their pros and cons. [45]

2.2.4.4 Word2Vec:

Word2vec is a common word embedding model provided by Google for improve words representation. This model was trained on nearly 100 billion of words from Google news dataset [31]. 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 will be represented in a vector of 300 dimensions. And each dimension represents one feature encoded from millions of words. The value of each feature ranging from zero to one. Figure 4.9 shows how the word “authorized” for example is represented in vector using Word2Vec model.

```
[8]  1  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 2.1: Word representation in Word2Vec model

If two words share the same meaning,In word2vec model they are represented by similar vectors. Figure 2.2 shows the most similar words to the word : "Quickly" using Cosine similarity, the resulting words are not just synonyms,it may be antonyms,hypernyms, etc.

```
In [4]: model.most_similar('Quickly')

Out[4]: [('quickly', 0.6232685446739197),
         ('Swiftly', 0.6026138663291931),
         ('Rapidly', 0.5781094431877136),
         ('Slowly', 0.5476222038269043),
         ('Instantly', 0.5315083265304565),
         ('Steadily', 0.523484468460083),
         ('Soon', 0.5119870901107788),
         ('swiftly', 0.5068031549453735),
         ('Quietly', 0.4989999532699585),
         ('Calmly', 0.49786293506622314)]
```

FIGURE 2.2: Similarity of words in Word2Vec model

Word2Vec contain two different models the first model called continuous bag of words (CBOW).This model context of word is represented by the words that occur around it. In the CBOW model, the distributed representations of context (or surrounding words) are combined to predict the word in the middle. While in the Skip-gram model, the distributed representation of the input word is used to predict the context. While Skip-gram use a word to predict a target context. In this study we adopted Skip-gram which is more deal with semantic meaning of words [1].

2.2.4.5 BERT :

BERT is a text representation technique stands for Bidirectional Encoder Representations from Transformers[2]. BERT confirmed to be state-of-the-art for a wide

[1]https://code.google.com/archive/p/word2vec/
[2]https://ai.googleblog.com/2018/11/open-sourcing-bert-state-of-art-pre.html

range of NLP tasks such semantic analysis and text classification. This breakthrough was the result of Google research in 2018 [14]. 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. Figure 2.3 describe the architecture of Masked-LM strategy. Where the "w" represent the words in the input sentence. And litter "o" represent the encoder output vector for each words using transformers. Finally, the model calculate the probability of each word in the vocabulary with softmax and predict the masked values.

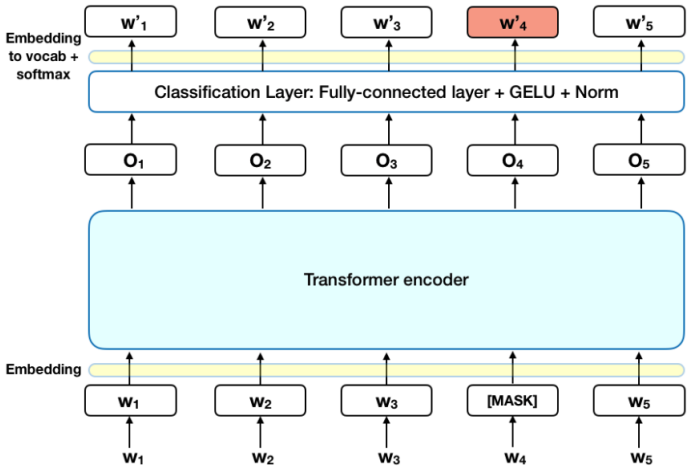


FIGURE 2.3: Masked-LM strategy. [19]

2.3 Machine Learning (ML)

Machine learning is a sub-field within artificial intelligence that enables a system to learn from train data. ML also defined as "the study of computer algorithms that allow computer programs to automatically improve through experience", Tom M. Mitchell [34]. Machine learning uses two main types of techniques: supervised and unsupervised learning. In supervised learning algorithms try to build a model relationship between the input features and target prediction, and the we can be able to predict the output values for new dataset. While in unsupervised learning, the algorithms try to find hidden patterns or relationship in input data.

2.3.0.1 Support vectors machine (SVM):

SVM is a popular supervised machine learning model. SVM uses classification algorithms for binary classification problems. The objective of the SVM algorithm is to find a hyperplane line that segregating the instances into two classes based on its features. SVM also used to solve multi classification problem using one-against-one and one-against-all strategies. This classifier used to solve linear classification problems and can also used to solve non-linear classification problems using a "kernel trick" function.[44]

For maximize the margin between hyperplane, gradient descent algorithm is used with cost function algorithm to tune the weights for each feature. See equation 2.2

$$W_{t+1} \leftarrow W_t - \eta t \bigtriangledown_w C(w_t) \tag{2.2}$$

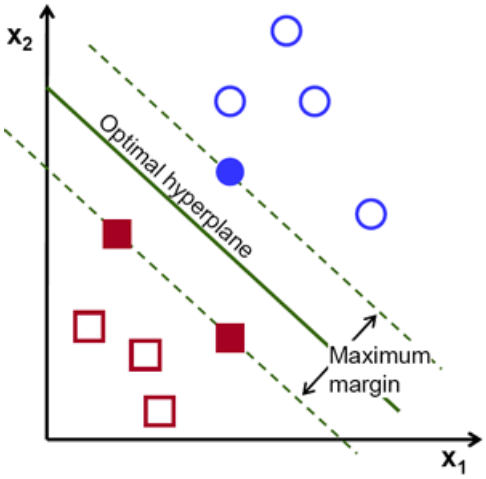


FIGURE 2.4: Support vector machine for linear classification

Equation 2.3 used to minimize w and maximize the margin:

$$min_w C(w) = \frac{\lambda}{2} \|w\|^2 + \frac{1}{N} \sum_{i}^{N} max(0, 1 - y_i f(x_i)) \tag{2.3}$$

2.3.0.2 Naive bayes classifier (NB):

NB is a supervised classification algorithms and probabilistic model based on Bayes theorem [25]. This algorithm is called "Naive" because it makes a naive assumption that each feature is independent on other features which is incorrect in real life.And the part "Bayes" helps us to find the probability of a hypothesis given our prior knowledge. It is one of the most common used supervised ML classifiers. NB have been demonstrated to be accurate and reliable in natural language classification issues. See equation 2.4 for Bayes Theorem:

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)} \tag{2.4}$$

2.3.0.3 Logistic Regression (LR):

LR is classification algorithm and also called logit model. logistic regression is the baseline supervised machine learning algorithm for classification and has a very close relationship with neural networks. Unlike linear regression which only dealing with continuous variables. LR dealing with discrete classes using the natural logarithm that transforms its output using the logistic sigmoid function to return a probability value which can then be mapped to two or more discrete classes figure 2.5. There are three Types of Logistic Regression. First, binary LR the categorical response has only two possible outcomes. Second, multinomial LR the categorical response has three or more categories without ordering. Third, ordinal LR three or more categories with ordering. [29]

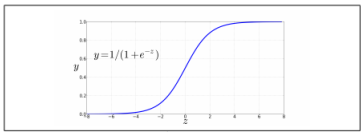


FIGURE 2.5: Sigmoid Function used in LR

2.3.0.4 Convolution neural network (CNN):

CNN is a type of neural network that commonly originally designed for deep learning computer vision tasks primarily used in image recognition and classification. Now a day CNN is a state-of-the-art technique in text classification. It 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). CNN architecture mainly comprised of three layers, convolutional layer, pooling layer and Fully connected input layer [30].

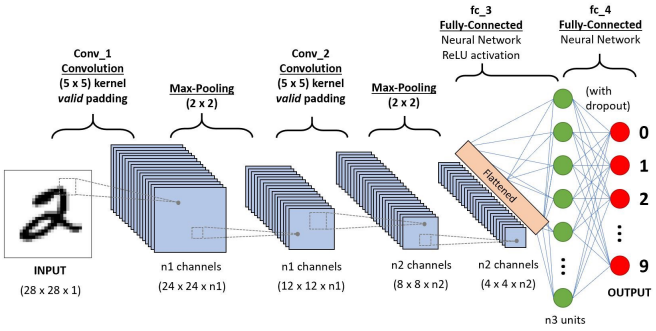


FIGURE 2.6: CNN architecture

1. **Convolution layer :**

   Convolution is the first layer in CNN architecture. The objective of this layer is to extract features through passing a filter over the input data (image) respect to its dimensions. Convolution preserves the relationship between pixels by learning image features using small squares of input data.

2. **Pooling layer :** Pooling layer typically applied after a convolution layer.The objective of this layer is to the number of features. Which reduces the dimensional of convolutional layer with retrieve the most important information. Pooling layer mainly use max or average pooling to reduce the dimensions of the data.

3. **Fully connected layer :** Fully connected layer are usually found towards the end of CNN architectures that takes the output of convolution and pooling layer to predicts the best label to find the class scores.

4. **Stride :** Stride: is the number of shift pixel that the filter move over the input matrix. In our model the filter will only move vertically (y-axis).

Chapter 3

Literature review

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

This chapter elucidates various recent studies directions observed in requirements classification. Rule-based approaches became an interesting topic to classify requirements in the past decade as it performed well for specific dataset. When the number of documents increased and the 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 chapter is organized in three subsections as follows : rule-based approaches review, machine learning approaches review and discussion.

3.1 Rule-based approaches :

Rule-based approaches classify text into organized groups by using linguistic analysis. These rules used to construct a model using syntactic elements of a text to identify pertinent categories based on its content. Each rule consists of pattern and a predicted category. This approach has been demonstrated by many researchers. Sharma et al, proposed a pattern based rule approach in order to parsing the requirements based on NLP [42]. They Suppose presence of a certain combination of words and its relationship are uniquely 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 [52]. 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 [11], suggested information retrieval approach to identify and classify NFR. They proposed a classification approach depend on training dataset for specify a 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 adopted linguistic knowledge to classify NFR in software requirement document [20] . The researches identified 9 groups of keywords: adjective, adverb, model keywords, etc. Where the frequency of each keyword was incorporated as feature in main feature list and was ranked using smoothed and non-smoothed probability measure. They set a threshold for each keyword to attribute it for 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[26]. 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 its used with other data-sets.

3.2 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 [27]. They used a data set from Amazon software reviews in order to train the model. The research used two classifiers: support Vector Machines and naive bayes 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. In this research they 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 in more effectively to extracting 14 categories of NFR from unconstrained requirements documents [43]. They collected 11 requirements documents from "iTrust and PROMISE" data-set. 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 through SVM performed twice effectively compared to naive Bayes . Furthermore,they found that k Nearest Neighbor (KNN) classifier with distance matrix had F1 measure score (precision and recall) of 54 %,while Naive Bayes 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 [53]. 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 [3]. 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 NFR is more difficult to identify than others.

While most studies mainly focus classification performance measure using precision and recall. Laszlos et al, considered time factor as part of measured performance [46].They selected 12 classifiers such as SVM, NB, linear kernel, KNN, Extra Trees and Linear logistic regression. The research used TERA-PROMIS data-set. They used data-set that consists of 625 requirements sentences. They found that Naive Bayes 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 [23]. In another research, Ramadhani et al [38], They proposed an automation system of identification of non-functional requirements from the requirement sentence-based classification algorithms. The researchers suggested additional semantic factors to use with 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 [51]. 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 [5].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 [13]. 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 multilayer 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 [37]. 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 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 [9]. 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 3.1 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.

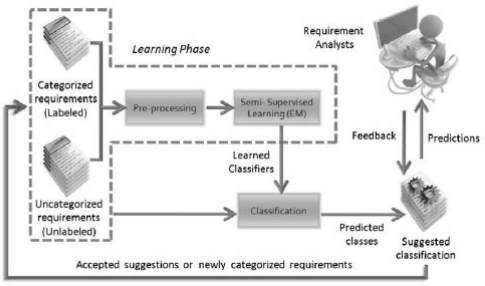


FIGURE 3.1: Semi-supervised approach for requirement classification

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 [41]. Their approach architecture starts from preprocessing, 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 [50] 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 [39], 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. Figure 3.2 shows layers in system design, which contains classifier layer and application layer that contain ontology. The research found that the ontology foundation of this work allow to automated convert requirements documents into a form of semantic representation.

While most research proposed supervised ML approach for requirements classification. Few research proposed unsupervised ML approach to handle the

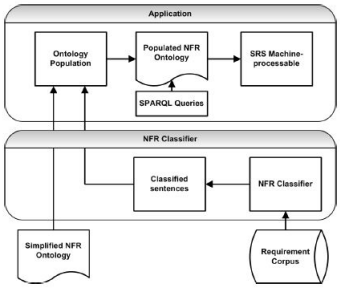


FIGURE 3.2: Hyprid Approach based on Ontology [39]

same issue. One of them done by Mahmoud and Williams [32]. 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 [7]. 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.

3.3 Literature review summery table:

In table 3.1 all reviewed studies were summarized. Research objectives and proposed approach were focused on for each study. Further, extraction features that were used in each study.

Study	Author	Study goal	Proposed approach	Extraction technique
S1	Sharma et al	Framework to automatically classify NFR from natural language requirements.	Proposed a pattern based rule approach in order to parsing the requirements based on natural language processing	Pattern based rules
S2	Xiao et al	Automatically extract access control policies from natural language software documents and resource access information from scenario-based functional requirements.	They proposed a linguistic analysis model to parse requirement documents with semantic meaning using semantic pattern matching	Syntactic-Pattern Matching.
S3	Huang et al	Identify NFR in both structured and unconstrained documents, including requirements specifications that contain scattered and non-categorized NFR	They proposed approach based on training dataset for specify a set of keywords "Indicator term" for NFR categories.	Syntactic analysis
S4	Hussain et al	Automate the process of detecting NFR sentences by using a text classifier equipped with a part-of-speech (POS) tagger.	Decision tree learning algorithm to classify NFR	Syntactic features extraction
S5	Knauss et al	Identify and classify security requirements	Proposed a statistical approach using Bayesian statistics	Syntactic analysis
S6	Kurtanovic and Maalej	Identify both FR and NFR from Amazon software reviews	Supervised machine learning approach using SVM and NB algorithm.	Syntactic and lexical features
S7	Slankas and Williams	Identify NFR from unconstrained requirements documents	Machine learning classifier using 5 supervised learning (KNN ,NB,)	Statistical analysis
S8	Zhang et al	Using text mining techniques to classify NFRs automatically	Machine learning (SVM with linear kernel)	Linguistical analysis (N-grams, individual words, and multi word expressions)
S9	Amasaki and Leelaprute	Evaluated the effect of vectorization methods on NFR classification	ML approach (SVM,RF,LR,NB)	TF, TF-IDF,W2V,D2V
S10	Laszlos et al	Comparison of performances of NFR classification processes	ML approach(NB , SVM, LLR, Label, Propagation, DT, Extra Tree, KNN.	Statistical analysis (TF,TF,IDF)
S11	Jiang et al	Multi-label text classification	Fuzzy similarity approach with SVM, MLKNN, FSKNN	Statistical analysis
S12	Ramadhani et al	Automation system of identification of non-functional requirements from the requirement sentence.	Classification algorithms of FSKNN with the addition of semantic factors-based ISO / IEC 9126.	Semantic factors (Hypernym, synonym based on WordNet)
S13	Winkler and Vogelsang	Classifying requirement specification as requirement and information using 89 requirements specification documents to train the model.	Deep learning approach	Random vectorization methods
S14	Baker et al	Classify NFR from PROMIS dataset include five requirement categories.	Deep learning approach (CNN)	Random vectorization Methods (TF, TF-IDF)
S15	Dekhtyar and Fong	Classify the requirements into FR and NFR categories using SecRec dataset that include security and non-security requirements	ML approach (NB , CNN)	TF-IDF and Word Count techniques
S16	Abdur-Rahman et al	Classifying non-functional requirements using RNN variants for quality software development	ML approach (RNN, GRU and LSTM)	Word2Vec
S17	Casamayor et al	Reduced the number of labelled requirements using knowledge provided by un-categorized requirements	Semi-supervised learning approach, Bayesian classifiers	Statistical analysis
S18	Shah et al	Specify NFR from informal requirements NFR-Specifier)	Hybrid approach based on ontology	Ontology formulation
S19	Vlas and Robinson	Discover and classify software requirements from both unconstrained and structured documents.	Semi-automated method	Multi-level ontology
S20	Rashwan et al	Classify requirements sentences into different ontology classes include (FR , several types of NFR, constraints and others.)	ML (SVM) based on ontology	Ontology with OWL individuals
S21	Mahmoud and Williams	Clustering NFR to its various categories, such as security ,usability , reliability, and performance	Unsupervised ML approach (Clustering techniques using K-mean)	Systematic analysis

TABLE 3.1: Summary of the reviewed studies

3.4 Summary:

In this chapter, the literature relevant to software requirements classification has been presented and discussed. In this section we will summarize our conclusions from these literature in 5 key points :

1. Rule based approach for classify software requirements was an interesting topic in the past decade. While the number of requirements documents increased and the fields have expanded, this approach faced significant challenges. One of them is the generalization.Where rule based approach performed well only for specific data-sets within number of confined domains.But in case the dataset have changed ,this approach definitely fail and the precision goes down as we indicated in the results of the previous literature. Recently, with the emergence of ML researchers shifted towards statistical methods using models generated by ML algorithms that most able to deal with various dataset. Recently, few studies have adopted hybrid approach, mixing machine learning techniques with rule based approach and getting good results.

2. Few number of pre-classified software requirements dataset were found in the previous literature. This problem is common within researchers in SE field. It is noted that most of the research used the same dataset. Furthermore, the used dataset contain few number of instances. Which is just a bit of what is commonly used for training models in ML. Another challenge in the usually used dataset are unbalanced requirements classes. And due to small size of samples, under sampling techniques will exacerbate the

problem of the small number of samples .While the over sampling technique in text mining don't usually support well the performance of the model.

3. Most of the literature didn't focus on the feature extraction techniques.And they didn't extract meaningful features in features extraction phase. Where the extracted features was not distinctive or shared among multiple classes. This problem got due to extraction techniques that they used where the most studies focused on the used ML approaches while the extraction techniques was ignored. Furthermore, the small number dataset prevent extract evident features from requirements sentences.

4. Requirements classification face challenge of diversity of dialects.The same requirement sentence can be written in several ways using a different wording or different forms. Semantic analyses can mitigate this challenge. We hardly found little number of literature have adopted semantic analysis to classify software requirements. Where, most of studies used syntactic analysis through random vectorization methods to transform the requirements sentences to numeric form. As a result, the model failed when its evaluated from different dataset as we mentioned in a number of literature.

5. Although Deep learning methods are proved to be very good for text classification, and achieved state-of-the-art results on a suite of standard academic benchmark problems. However, few number of studies adopted deep learning approaches in requirements classification. This encourages us to study how successful adopt deep learning approach to solving such issues.

Chapter 4

Research Methodology

The literature review summary in the previous chapter highlighted the need for semantic analysis and feature extraction techniques to identify and classify NFR. Furthermore, we highlighted on the failures of the rule-based approach. This chapter introduce our methodology that adopted NLP techniques and ML algorithms to identify and classify NFR from unconstrained documents. Figure 4.1 shows a block diagram represent 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.

4.1 Data description :

In this thesis 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" [16]. In this dataset requirements documents had written in natural

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

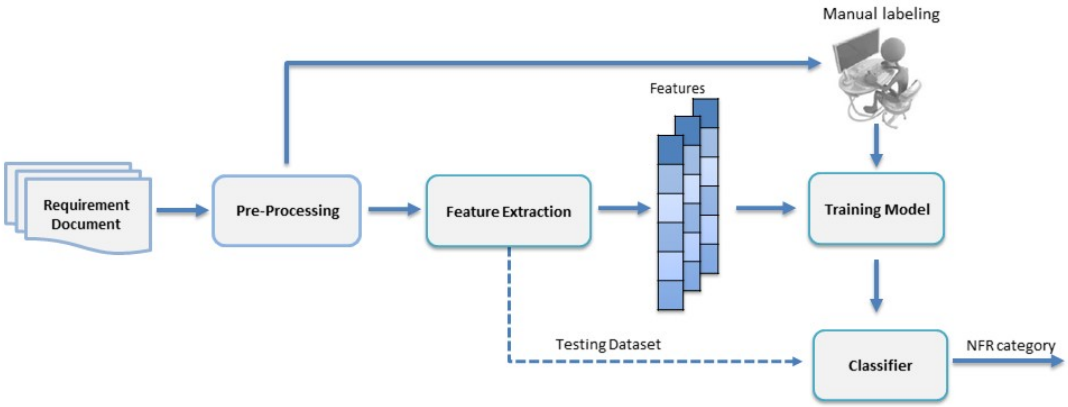


FIGURE 4.1: Overview of research approach

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), unstructured (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 4.2 shows the distribution of documents in the PURE dataset.

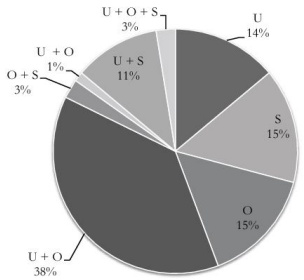


FIGURE 4.2: PURE dataset distribution [17]

4.1.1 PURE dataset annotation:

Supervised learning in any ML approach needs a pre-labeled dataset in order to train the models. As mentioned earlier, the PURE dataset include unconstrained requirements documents and unlabeled. In order to conduct supervised learning experiments on this dataset, a sentence-level manual annotation is required. This methodology in adopting manual labeling were used in a number of the studies we discussed earlier. For example, Casamayor et al [9] used manual labeling to identify requirement sentences categories in order to identify the type of further requirement sentences in iterative process. 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. In most cases, in the requirement document, each sentence is talking about one software requirement.Therefore, we decide to do the annotation (labelling) process at the 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 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, as shown in figure 4.3.

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

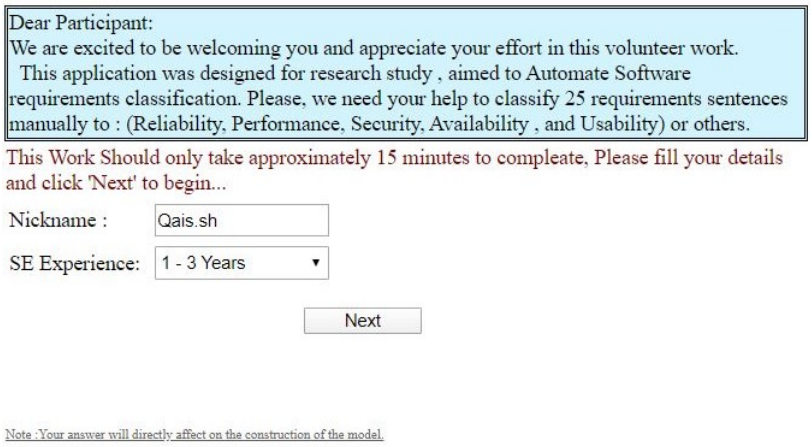


FIGURE 4.3: Requirements labeling site : Registration Page

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. To ensure the volunteers' accurate knowledge about the different types of NFR. The definition of included NFR listed below:

1. Reliability : specify the factors required to establish the required reliability of the software system at time of delivery. [22, p 6]

2. Availability: specify the factors required to guarantee a defined availability level for the entire system such as checkpoint, recovery, and restart. [22, p 6]

3. Security: specify the factors that protect the software from accidental or malicious access, use, modification, destruction, or disclosure.[22, p 6]

4. Usability :specify the ease with which the user is able to learn, operate, prepare inputs, and interpret outputs through interaction with a system.[22, p 7]

5. Performance :specify the factors the handles software system such as capacity, throughput, and response time. [22, p 7]

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 shown in figure 4.4. 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 by at least 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.

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 4.5 show the distribution of requirements for 5 types of

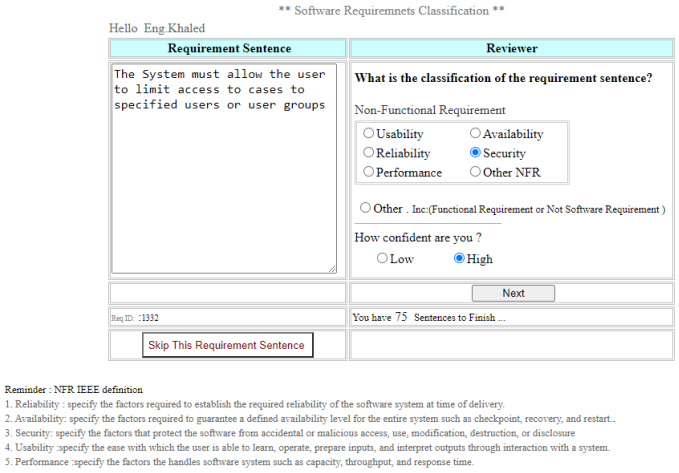


FIGURE 4.4: Requirements manual classification page

NFR.

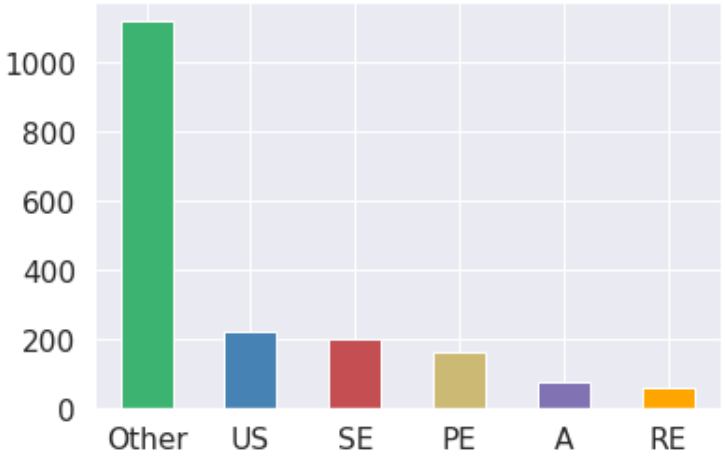


FIGURE 4.5: PURE dataset

4.1.2 Dataset balancing :

Usually, in any real data-sets, there is always some degree of imbalance between classes. And 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 4.5, there is high degree of imbalance between requirements categories (classes).

This issue is common in requirement engineering domain. 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 4.6.

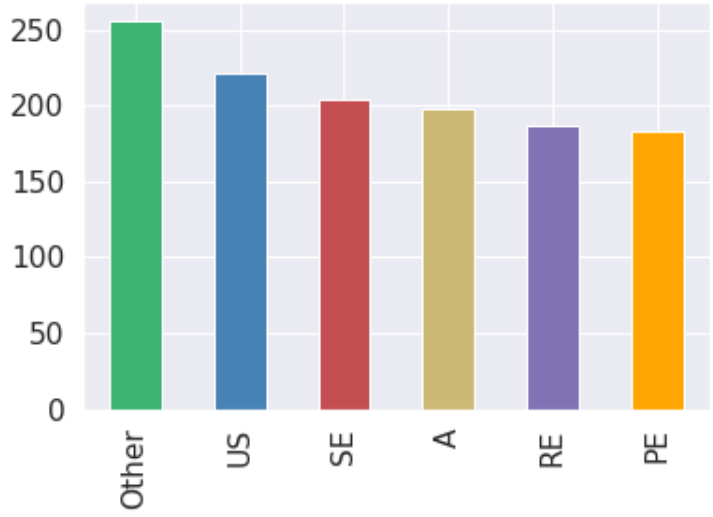


FIGURE 4.6: PURE dataset after balancing

4.2 System design:

The system design of our study consists of three main parts, namely:dataset pre-processing, features extraction and ML classification as shown in figure 4.7 . The following sub-sections describe each of these components:

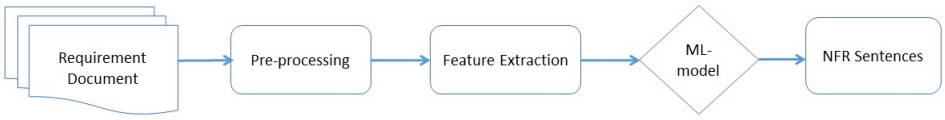


FIGURE 4.7: System Design:

4.2.1 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. As described in the following subsequent subsections.

4.2.1.1 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 were 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

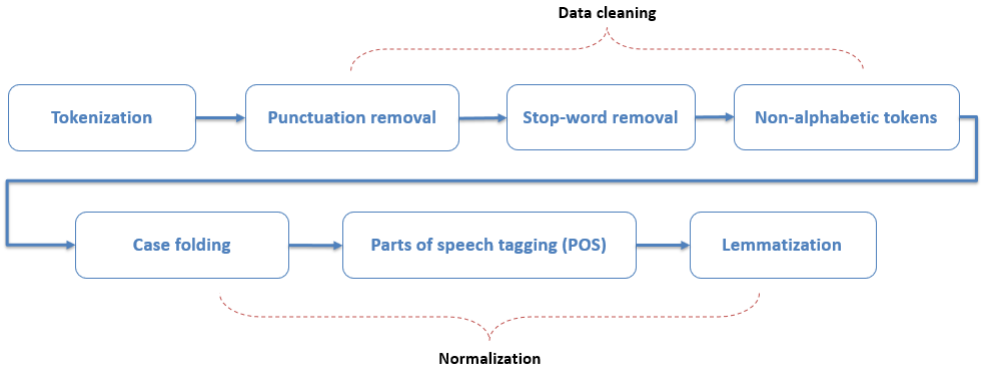


FIGURE 4.8: Data Pre-processing Tasks

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 which is a python library [3] used to extract English sentences from a document. 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', '.']

4.2.1.2 Data cleaning:

Data cleaning is one of the first steps in text pre-processing. It is an important step before the data is ready for analysis. In its nature, requirement sentence, as most of 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.

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

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 (he ,you ,that,should..etc), upper-case and lower-case words, symbols. 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 such as stops, question marks, commas, colons, etc were removed from the requirement sentences. That because they do not help in the semantic meaning of the sentience. In our model, the semantic meaning based on the basic words in that sentence. The second step is stop-word removal. In this step, all high frequency words, such as they, you, have,should etc, don't add any essential information to the requirement sentence were removed. In our model, we used python library called natural language tool kit (NLTK) [4]. This library contains most of the stop words in English language. The last step in this 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', '.']

Cleaned sentence :

['number', 'mistakes', 'noted', 'students', 'shall', 'decreased', 'first', 'year', '.']

4.2.1.3 Normalization:

In normalization process, we aimed to convert all the words to a more uniform sequence by transforming it to a common base form. In this task, we improve

[4]https://www.nltk.org/

the text modelling and the text matching. This task is applied on the words level by three steps:case folding, Parts of Speech (POS) tagging and Lemmatization.

Cleaned requirement sentence :

['Number', 'mistakes', 'noted', 'students', 'shall', 'decreased', 'first', 'year', '.']

Normalization :

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

4.2.2 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 state-of-the-art vectorization techniques in NLP . Two of them are syntactical based methods: TF and TF-IDF. As mentioned in literature review chapter in this thesis. These vectorization methods are the most common used in text representation. The other two vectorizatin methods are semantical based methods: Word2Vec[5] and BERT[6]. These methods are the state of the art language representations based on unlabelled 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 section explains in more details how these NLP methods are used to transform the requirements sentences from text to numerical vectors:

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

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

4.2.2.1 Random vectorization methods :

- **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 we generated vector in dimension equal to the total number of normalized words which are equal to BOW size. 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 4.1: 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. N-gram

is a technique to improve TF to adopt local ordering when we generate vectors to avoid the disadvantages of un-ordered words.

- **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 such as 'I', ,'and' ,'He' 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 features that can be extracted from the requirement sentences so that can differentiate between NFR categories. Table 4.2 shows how the requirement sentence will be represented in TF-IDF:

TABLE 4.2: Requirement sentence representation in TF-IDF

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

4.2.2.2 Word embedding method :

- **Word2Vec :**

  In our research, Word2Vec [7] 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

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

of words. The value of each feature in the word representation ranging from zero to one. Figure 4.9 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.

```
[8]  1  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 4.9: Word2Vec vector representation

- **BERT model:** BERT [8]is an inflection point in the application of machine learning for NLP.In this thesis we used BERT model to represent requirement sentences in semantic numerical vectors. Then, we trained the classifiers on top of the transformer output of the BERT model.

4.2.3 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

[8]https://github.com/google-research/bert

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 sections we propose how we performed the four ML models that we used in our system:

4.2.3.1 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 [7]. 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.

4.2.3.2 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 [7]. It is widely used to solve NLP classification problem as mentioned in Literature review. Naive Bayes 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 discussed earlier in this chapter (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. In our case, if we want to calculate the probability of requirement sentence to be, for example, "Security" NFR:
Yi : represent security label.
(x1...xn) : requirement sentence feature vector.
P(yi | x1, x2, . . . , xn) : probability of yi being (security requirement) given (x1.. xn) features exist .
P(yi) : the prior probability of security requirement in the dataset.
P(x1, x2, . . . , xn | yi) : probability of requirement sentence features given its security requirement.

4.2.3.3 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 [47]. 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.

4.2.3.4 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 4.10.The following section describes how these layers are performed.

1. **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 4.3 shows the representation vectors for the requirement sentence: " The product shall be easy for a relater to learn." in Word2Vec model.

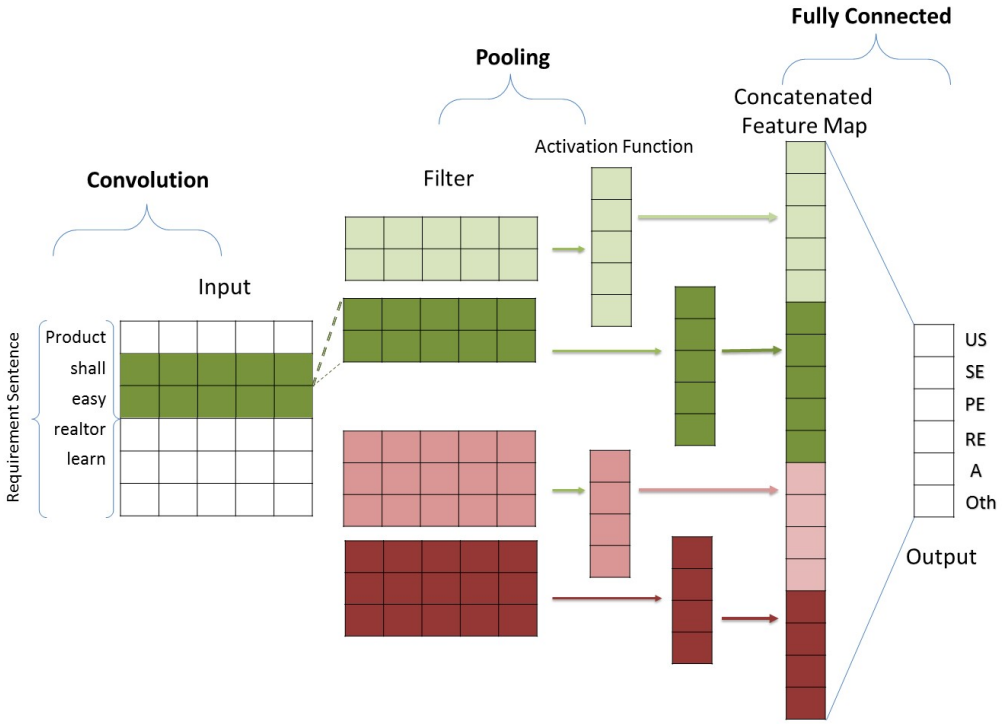


FIGURE 4.10: CNN architecture for sentence classification

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.

2. **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

TABLE 4.3: Sentence representation 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

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.

3. **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. The number of steps is calculated by equation 4.1:

$$Numberof Steps = (CL_{height} - Filter_{height}) + 1 \tag{4.1}$$

CL : Convolutional layer

4. **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.

4.2.4 Fusion models

In previous models we introduce different feature extraction techniques. Each techniques has its own properties and advantages. In this thesis 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 them together. The idea of this approach based on combined the four NLP techniques in on fusion 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 sub-systems that use different NLP methods into one overall system to get the best classification result. We combine four CNN classifiers trained on the four different features as front-end 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 for training a front-end classifier. We used logistic regression classifiers in the back-end model because we are deal with only 25 features continues value. logistic regression use 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 4.11 shows the architecture of proposed combined model. In front-end model each classifier produce one vector include probability for each NFR category. In back-end model we fuse (combined) the four produced vector in one vector and we feed it in back-end model.

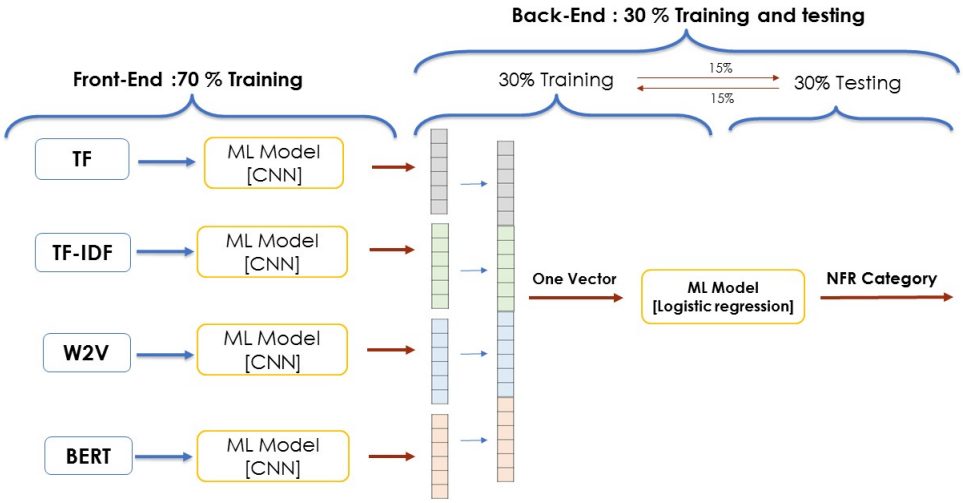


FIGURE 4.11: Fusion model architecture

4.3 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 will 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 model. Test set in this model was used for both training back-end model and testing overall fusion model. Where the test set was divided into two equal parts. And we swapped between them by performing the experiment twice.

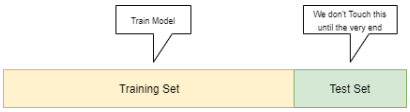


FIGURE 4.12: Train/Test Method

Most common classification performance metrics was used in all experiments; accuracy, precision, recall and F1-score metrics [7]. Precision metric, measure 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 4.2 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.

$$precision = \frac{TP}{TP + FP} \tag{4.2}$$

While 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 4.3 shows the formulation of recall

$$recall = \frac{TP}{TP + FN} \tag{4.3}$$

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:

$$F - measure = \frac{2.precising \times recall}{precising + recall} \tag{4.4}$$

Statistical test was used in this experiment 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 they run the experiment many times [18] [28]. 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.

Chapter 5

Experiments and results

Referring back to the research questions mentioned in the introduction, this thesis aims to investigate the effectiveness of NLP techniques and ML approaches for NFR classification from unconstrained requirements documents. In order to present answers to those questions, we performed various experiments using PURE dataset were described in the previous chapter. In this chapter we present our experiments in view of this thesis research questions.

Recalling to the first research question [RQ1]: How well can natural language processing techniques can identify effectively NFR from unconstrained requirements documents. We used four vectorization techniques to represent requirement sentences within each classifier :

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

And recalling to the second research question [RQ2] : How well machine learning algorithms can automatically classify NFR categories efficiently". We used 4 machine learning classifiers :

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

The main strategy is to use the four NLP techniques for mapping requirement sentences into numeric vectors and then train and evaluate the four ML classifiers, each on the four vectors types. By this, we conduct 16 experiments.

5.1 Environment Setup:

To perform our experiments, we used Google Colab [1] cloud service which supports GPU processor. It is Jupyter notebook environment that runs entirely in the cloud. Table 5.1 shows detailed specifications of the processing capability of Colab cloud service.

Python programming language was used for developing the systems and conducting all the experiments. Python is used because of its preference compared to other languages in ML and NLP techniques [2]. It contains massive number of frameworks and libraries for NLP techniques and data pre-processing[36].

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

TABLE 5.1: 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

5.2 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.

5.2.1 Tokenization :

Requirement sentences were extracted from requirement document using python Texttract package [2]. 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) [3]. This module includes tokenizer package to divide strings into lists of sub-strings based on white space and punctuation.

[2]https://textract.readthedocs.io/en/stable/
[3]https://www.nltk.org/

5.2.2 Data cleaning :

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

5.2.3 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 toolkit [4], which is a large lexical database contain sets of cognitive synonyms in English language [15]. The example below describes how a cleaned sentence is normalized.

FIGURE 5.1: Pre-Processing Tasks in python

[4] https://wordnet.princeton.edu/

5.3 Features extraction :

Multiple NLP techniques were used to extract features from the requirements sentences. First of all, NumPy Python package [5] 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.

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

FIGURE 5.2: Requirement sentence representation in TF

5.3.0.2 TF-IDF vectorization method :

TF-IDF is distinguished from TF method in words weighting. In this method, in addition to count each word. The weight of each word is calculated, which signifies of its importance in all training requirement documents. To do that,

[5] https://numpy.org/

TfidfTransformer class were used from sklearn [6] feature extraction library to transform requirement sentences to TF-IDF vectors. Each requirement sentence represented by vector in 1247 features, which represent the number of unique words in all requirement document. Figure 5.3, shows how TF-IDF method calculate the weight for each word in requirement sentence, in which the words that are frequently repeated will have little weights such as the word "system" as its shown in yellow color.

```
1  print(req_sent)
```

```
system must allow user limit access case specify user user group
```

```
1  X = tfidf.fit_transform(req_sent).astype('float32')
```

```
  (0, 5)        0.25628293
  (0, 63)       0.28183177
  (0, 183)      0.3249719
  (0, 533)      0.4549304
  (0, 689)      0.5171206
  (0, 788)      0.23715179
  (0, 1179)     0.40177298
  (0, 1236)     0.15141967
  (0, 1339)     0.18524955
```

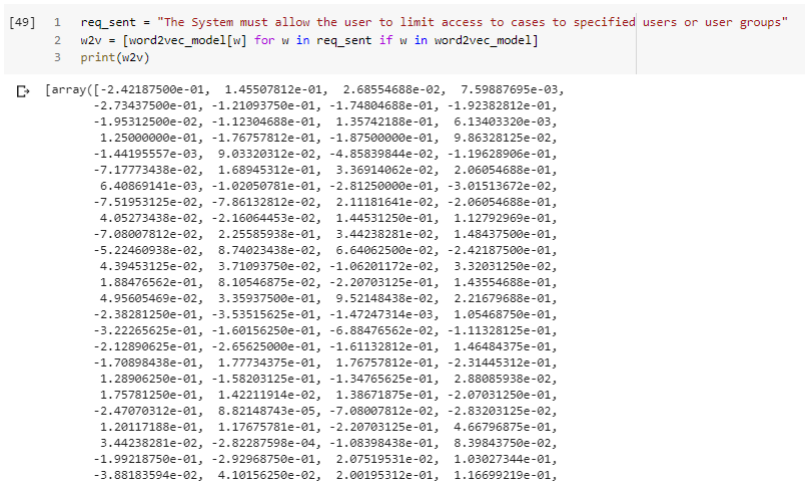
FIGURE 5.3: Requirement sentence representation in TF-IDF

5.3.0.3 W2V vectorization method :

Word2Vec is one of the most popular google model to produce word embeddings [7]. 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 vectors in each

[6] https://scikit-learn.org
[7] https://code.google.com/archive/p/word2vec/

requirement sentence. So, one 300 dimension vector will represent each requirement sentence.In CNN classifier that deal with 2 dimensional vectors requirement sentences represent by 2 dimension numpy array.x-dimension represent each word vector. While y-dimension represent the words that make up requirement sentences. Padding 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 PURE dataset after pre-processing. We used pad_sequences package from tensorflow keras library [8]. Gensim libraries[9] also used to load W2V model, which is an open-source library implemented in Python for topic modeling and NLP.Figure 5.4 shows how requirement sentences represented in this model.



```
[49]  1  req_sent = "The System must allow the user to limit access to cases to specified users or user groups"
      2  w2v = [word2vec_model[w] for w in req_sent if w in word2vec_model]
      3  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 5.4: Requirement sentence representation in W2V model

5.3.0.4 BERT Model :

In this experiment we used ktrain Python library that contain BERT-Base pretrained models on tensorflow [33]. using 12-layer, 768-hidden, 12-heads, 110M

[8]https://www.tensorflow.org
[9]https://pypi.org/project/gensim/

parameters. We used TPU processor on Google colab platform with 32G of RAM to handle this large model. As W2V model, BERT transform the words into 300 embedding vector.And each requirement sentences were represented by 2d numpy array.In traditional ML the 2d numpy array reshaped into 1d features vector using mean statistics. In CNN model 2d numpy are is compatible with convolution layer. Padding technique is also required to unifies the dimension of numpy 2d array. For PURE data set the maximum number of words in requirements sentences was 50. Thus, the size of padding technique was 50. Figure 5.5 shows how requirement sentences represented in this model.

```
[49]  1  print(req_sent)

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

[50]  1  BERT_feature(req_sent)

array([[4.83246142e-04, 1.09464148e-04, 9.99349415e-01, 1.50639635e-05,
        3.25840592e-05, 1.02761023e-05],
       [9.99956608e-01, 5.35872644e-08, 1.14620798e-05, 1.30818760e-08,
        3.12663797e-05, 7.21135052e-07],
       [9.99872088e-01, 9.01327212e-07, 9.10192739e-06, 4.83709606e-08,
        4.21396408e-06, 1.13546419e-04],
       ...,
```

FIGURE 5.5: Requirement sentence representation in BERT

Finally, the classifier model predict based on its experience, to which category the requirement sentence belongs to using softmax activation function. Figure 5.6 shows an array represent 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 5.6: Output layer for CNN model

5.3.1 Parameters sitting 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 its in Scikit-learn. Which is the most common free machine learning library for Python and matlab [10]. In this subsection we list all parameters for each classifiers that were used in our experiments.

5.3.1.1 Naive bayes :

Scikit-learn [11] 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]

5.3.1.2 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]

5.3.1.3 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]

5.3.1.4 Convolution neural network :

For all CNN based experiments,we used keras tensorflow library which is open-source neural-network library written in Python [12]. Embedding dimension parameter was fixed to 300, that equal to word vector dimension that generated from embeding model. And NB-class was fixed to the number requirement

[10]https://pypi.org/project/scikit-learn/
[11]https://scikit-learn.org/
[12]https://www.tensorflow.org/

categories target classes, which equal to 6. While we were tuning the number of filters until the best result was converged. And filter size was fixed to bi-grams,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']

5.4 Experiment 1: Optimal ML classifier using TF method

The Objective of this Experiment is to identify the optimal ML classifiers using TF method. The four described ML approaches were adopted include three traditional approaches NB, SVM, logistic regression and deep learning approach CCN. Each experiment is repeated 10 separate times to verify and to avoid the randomness of results. Then statistical tests were performed to assess the validation of results.

For the traditional ML approaches the requirements sentences feeds to the classifier using 1 hot vector in binary list form, that represent the count of all words in requirement sentence. Each item in the list represents one feature, and the type of requirement sentence represent the label. In CNN, the output vector of TF method fed to the classifier vertically as an image. Figure 5.7 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.

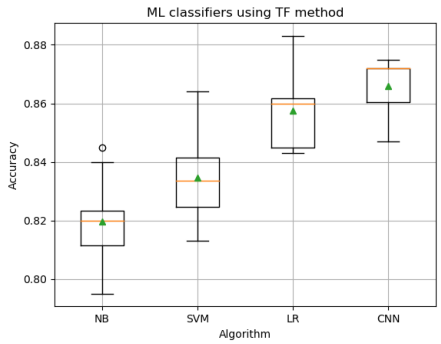


FIGURE 5.7: ML classifiers accuracy using TF method

We also report three performance metrics, precision, recall and F1-score metrics for each NFR typee using Sklearn libraries [13]. The results for each NFR types are shown in table 5.2.

TABLE 5.2: 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**

5.5 Experiment 2: Optimal ML classifier using TF-IDF method

In this experiment,we aimed to determine the optimal ML classifier using TF-IDF method. For the traditional ML approaches the requirement sentence feeds to the classifier using 1 vector in float list as we did in TF method. Where the weight of each word in requirement sentence were evaluated. As previous experiment this experiment ran 10 times. The accuracy results for 4 classifiers is shown in figure 5.8.

[13]https://github.com/scikit-learn/scikit-learn

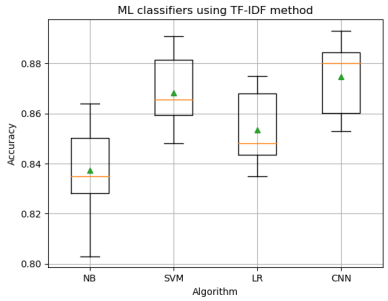


FIGURE 5.8: ML classifiers accuracy using TF-IDF method

Table 5.3 also shows the three performance metrics that used in classifiers for each NFR types that we adopted in this thesis.

TABLE 5.3: 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**

5.6 Experiment 3: Optimal ML classifier using W2V model

In this experiment, the requirement sentence was feeded in first three classifiers NB,SVM and LR in 1 dimension vector. Statistics mean, were used to perform this task. While, CNN classifiers has an architecture to feed in with 2 dimensions vectors as its performed with images. Figure 5.9 shows the classification mean and median accuracy in 10 runs for each ML classifiers.

5.7 Experiment 4: Optimal ML classifier using BERT model

In this experiment we used the state of the art language model for requirement sentence representation. BERT model words as w2v in 300 victor size. Similar to

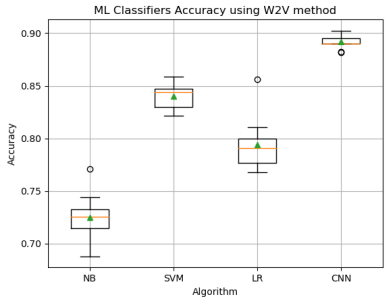


FIGURE 5.9: ML classifiers accuracy using W2V model

TABLE 5.4: 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**

the previous experiments, the mean for all vectors was calculated in first three ML classifiers.In CNN classifiers the 2 dimension of the requirement sentence feed in convolution layer in 2 dimension. The accuracy of ML classification for the 10 runs was reported through box plot, as its shown in figure 5.10.

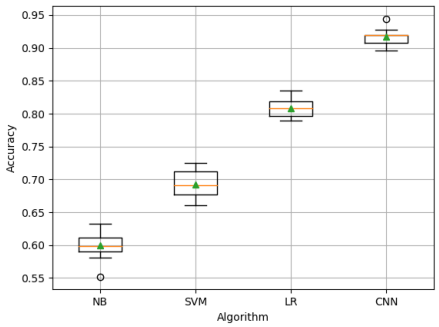


FIGURE 5.10: ML classifiers accuracy using BERT model

Furthermore, we found the three performance metrics for each NFR types using BERT representation model. Table 5.5 presents precision, recall and F1

score for each NFR category.

TABLE 5.5: 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**

5.8 NFR classification accuracy in different NLP techniques:

After we performed the four experiments we summarize the mean of the accuracy results for all classifiers using all used NLP transform methods. See figure 5.11.

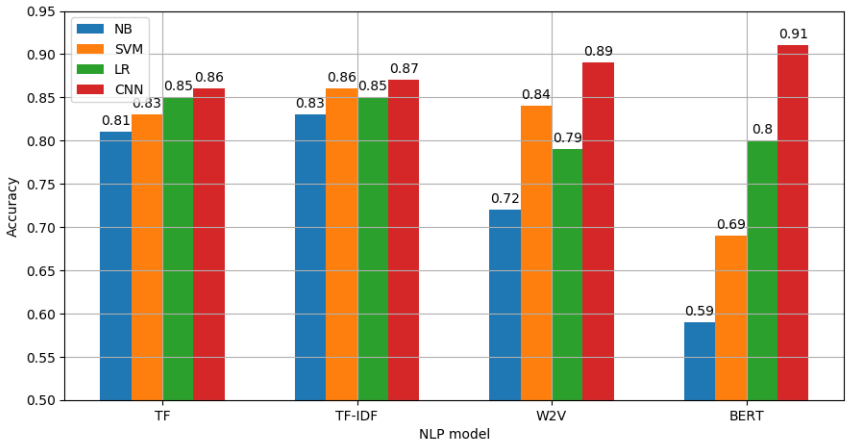


FIGURE 5.11: ML classifiers with all NLP methods

5.8.1 Optimal NLP techniques to transform NFR using CNN:

From the results of the previous presented experiments, CNN approach achieves the best accuracy in NFR classification in all NLP transform methods. In figure

5.12, we compare the accuracy of CNN classifier using four NLP feature extraction methods.

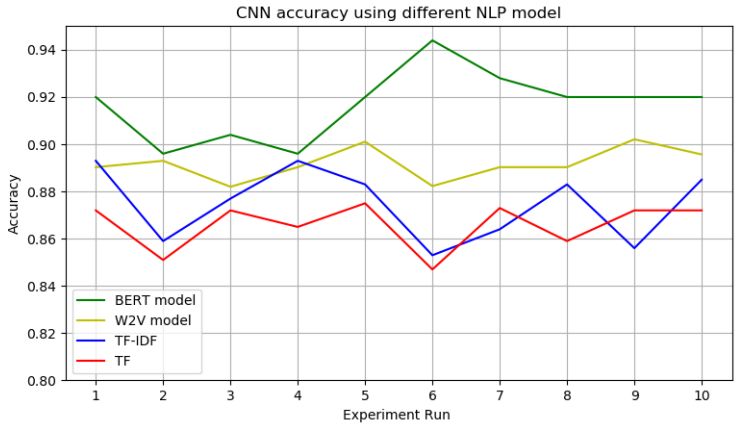


FIGURE 5.12: Optimal NLP techniques using CNN classifiers

5.9 Experiment 5 : Fusion model

The objective of this experiment is to exploit the best of the four extraction features types together. CNN classifiers in previous experiments outperforms the other classifiers four all features extraction types. In this experiment we built four CNN classifiers feeded by the four different features extraction methods in parallel. The four CNN classifiers were combined together into one fusion system. The fusion is done at the model level by concatenating the output vectors of each CNN classifier (each trained on one feature extraction type) for the six classes into one feature vector, each feature represent weight to predict the correct NFR category. The resulting new feature vectors are then used to train a logistic regression classifier at the back-end model. Figure 7.1 shows sample results for fusion front-end model. To train the back-end logistic regression, we need a new training data different than the 70% training subset. Therefore, we split the 30% testing dataset into two equal subsets (i.e. 15% of the dataset).

The scores of the first subset are used to train the logistic regression back-end, whereas, the second subset is used to test the overall fused system. Then, we repeat the same experiment, but this time we exchange the two subsets. I.e. the second subset is used to train the back-end classifier, whereas, the first subset is used to test the overall fused system. The overall fused system performance is the results obtained from the two subsets. 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 take less than 3 second. But mainly, the model building process is a one-time process, then the model will be ready to use.

TABLE 5.6: 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 report for fusion multi models.

By this results fusion model outperform the accuracy of the all previous by improving the classification accuracy by 2.4% Compared to the best results we achieved in BERT model. We also performed a comparison between the CNN classifiers using 4 NLP methods that we adopted, in addition to our fusion

model. Figure 5.13 shows how the accuracy for BERT model outperform statistical feature extraction techniques. using CNN classifiers, and how fusion model improve BERT model by 2.4%.

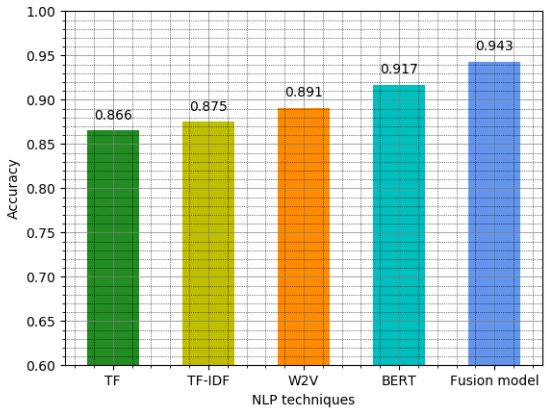


FIGURE 5.13: CNN classification accuracy using NLP techniques and fusion model

To analyze the results obtained from fusion model we formulated confusion matrix in figure 5.14 which describe true positive,true negative, false positive and false negative for fusion model results.

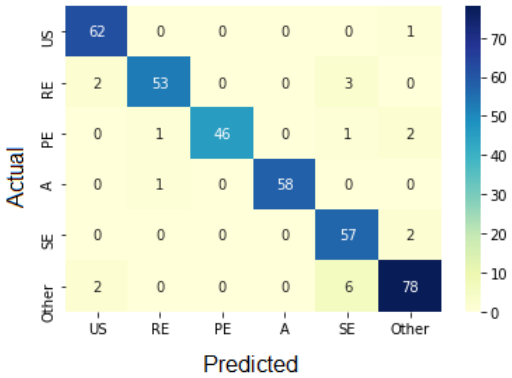


FIGURE 5.14: Confusion matrix for fusion model results

5.10 Statistical Test :

In this thesis, each classification experiment is executed 10 times to estimate the variability of the results and to evaluation how close to each other. Also, 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 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 5.7.Furthermore we found the median and Interquartile range for each classifiers in the 10 runs, see table 5.8. Then we calculated a wilcoxon statistic test between each classifiers.

TABLE 5.7: 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 5.8: 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	$7.25e-01_{1.80e-02}$	$8.44e-01_{1.73e-02}$	$7.91e-01_{2.34e-02}$	$8.90e-01_{4.73e-03}$

Figure 5.9 represent wilcoxon results compared two paired groups of classifiers results. Triangle symbols ($\triangledown$, ▲) indicate that results are statistically significance in which p-value < 5%. The inverted white triangle $\triangledown$ represents the superiority of the algorithm 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 (-) were used to indicate that the caparison between the two classifiers are not statistically significance, p-value > 5%.

TABLE 5.9: Wilcoxon values of the accuracy indicator (TF, TF-IDF, W2V, BERT).

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

Chapter 6

Discussion:

In this chapter we discuss the experiments results presented in the previous chapter. We also reflect those results on the research questions listed in the introduction of this thesis and how the obtained results present an answers to the questions.

Let's start with the second research question [RQ2]: **How well machine learning algorithms can automatically classify NFR categories efficiently, based on IEEE-Std 830-1993 standard?**. For this question we need to investigate the effectiveness of the traditional and the state-of-the-art ML techniques in classifying NFR from unconstrained software requirement documents.

The experiments from 1 to 4, three common ML classifiers were used in addition to the CNN. The objective of these experiments is to investigate the efficiency of machine learning techniques to classify NFR. The classifiers achieved varying results. Where the traditional ML approach (SVM.NB and LR) achieved relatively good results with syntactic analysis. While its results fell with embedding models. NB classifier achieved accuracy range from 60.3 % to 83.7%. SVM classifier achieved accuracy in range between 69% to 86.8%. While Deep learning approach (CNN) outperform all traditional approaches by achieving results

range from 85% to 92%. By analyzing the results of all experiments, obviously traditional ML approaches achieve better results in random vectorization methods than word embedding methods. This conclusion can be explained due to failure traditional ML approaches to deal with high dimensions vectors in word embedding models. While CNN approach was more successfully with word embedding model due to ability of this model to handle 3D inputs in the convolution layer such as word2vec and BERT model. We also can conclude that the NFR classification accuracy mainly depends on two factors; the type of classifier and the vectorization method that was used.

The results of the measurement metrics; precision, recall and F1-score metrics were closed to the accuracy results in the most cases except for the security requirement class, as shown in tables 5.10 and 5.9. Precision metrics measure positive prediction value. 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 solve this problem through improve detecting security class with a precision range between 82% to 98% as shown in table 5.10. In the average precision (positive rate) for all classes in NB, SVM and LR the classifiers achieved results between 60% to 80%. Whereas, the CNN classifier was able to achieve precision between 80% to 90%.

Sensitivity of ML model were measure in this study using recall metric. Which measure the fraction of the total amount of relevant instances that were actually retrieved. In other word measure the ratio between how much were correctly identified as positive NFR to how much were actually positive NFR.This metric also expresses the ability to find all relevant requirement category. CNN classifier achieved best recall results in range between 86% to 91.4%. This means out of every 100 NFR we 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 94% weighted recall. 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 thus the software engineer can define most NFR that the customer has expressed through the requirement document.

Recalling to first research question [RQ1]: **How well natural language processing techniques can identify effectively NFR from unconstrained requirements documents?**. We investigated the effectiveness of the NLP techniques in representing the NFR categories in the unconstrained requirement documents. Two experiments were performed to present an answer to RQ1. First, we used 4 different NLP techniques with each ML classifier, including random vectorization methods [TF and TF-IDF], and word embedding models [W2V and BERT]. The results in figure 5.11 shows how natural language techniques effect on classification results. And while some NLP techniques do well with some types of ML classifiers, the others achieved low 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. 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.

This finding explained by the nature of each model and the architecture on which it is built. Traditional NLP techniques transform each requirement sentence into one vector, whereas, word embedding model represent each word in the requirement sentence into one high-dimensional vector (300). In this case, more amount of information and semantic meaning of the requirement sentences are preserved.

Another comparison we made presented in figure 5.12 to investigate the optimal NLP method to represent NFR with CNN approach. The results show that BERT model achieved the best accuracy result in all experiments runs, Where, TF achieved the worst. This leads us to conclude that NLP techniques that represent the requirements sentences in high dimensional multiple feature vectors such as word embedding achieve a higher accuracy than NLP methods that represent the requirement sentences in one feature vector with low mount of information. Finally, we present all previous results in par plot figure 5.11. It is clear the superiority of CNN algorithm over all traditional approaches using semantic analysis (BERT, W2V). While CNN doing worse with syntactic analysis (TF, TF-IDF).

[RQ3]. **How well the proposed system performance can improve by fusing different NLP features together into one system?**. This question were handled in experiment 5. We preformed fusion model to combined the four NLP techniques to feed it in 4 parallel CNN models. The objective of this experiment is to exploit all the features from all NLP techniques. Some NLP techniques have the ability to extract syntactic features from requirement sentences like TF

and TF-IDF, Other NLP techniques like W2V and BERT have the ability to extract semantic features. The results show that fusion system outperforms all the individual proposed systems in previous experiment, with an accuracy of 94.3%. It achieved 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. Comparing to the previous experiments fusion model outperform the accuracy of all previous experiment by improving the classification accuracy by 2.4%. Figure 5.13 also shows a comparison between the CNN classifiers using 4 NLP techniques that we adopted in addition to our fusion model. The figure shows how the accuracy for BERT model outperform statistical feature extraction techniques using CNN classifiers, and how fusion model improve the accuracy of BERT model by 2.4%.

Confusion matrix also have been used to analyzing classification results to depict true and false positives as well as true and false negatives. figure 5.14 illustrates the confusion matrix for the classification results for fusion model and include heat map to represent the number by darker color.

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 fusion model categorize correctly 62 requirement sentences as usability requirement sentences. and categorize one Incorrectly as other (False negative). By looking in 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, while it actually related to other categories three reliability, one performance and 6 others. That mean number of

extracted features from security sentences are shredded among multiple categories. It also can conclude that fusion model can classify correctly 58 availability NFR categorise from 59 actual sentences. While achieved 53 reliability from 58 actual sentences. Comparison between all categories of NFR categories, we find that performance NFR category was able to achieve the best positive prediction value, where all the extracted positive performance sentences were actually performance requirements sentences. And we also find that usability and availability achieved the best sensitivity value. Where from 63 usability requirements sentences fusion model was able to categorize correctly 62 requirement sentences as usability from 63 usability requirement sentences. And it was able to categorize correctly 58 requirement sentences Of the 59 sentences that are actually availability requirement sentences.

Wilcoxon statistical test were used to establish if they are statistically significantly different from one another and the results are real and not caused by luck or chance. Table 5.7 represents the mean and standard deviation for the classification accuracy in the 10 runs of the experiments. Table 5.8 also represent the median and interquartile range accuracy. The darker shaded level in two tables represents the better result. That mean CNN classifier exceeded all other ML classifiers in all NLP methods. The results of Wilcoxon statistical test represented in table 5.9. This table shows that CNN outperformed other classifiers in different NLP techniques in statistically significant manner. Where the inverted white triangle ($\triangledown$) 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 ($\blacktriangle$) represents the opposite relation. And the dash line represent that there is no statistical superiority between the two parties. That mean the relation between the two classifiers is not statistically significant where p-value higher than 0.05 (> 0.05).

Chapter 7

Conclusion and future work:

Software requirements are captured and documented in human natural language. It documented in a form called (requirements document). In general, requirement document contains both functional and non-functional requirements. Manually analyzing NFR is tedious and time consuming in SDLC. In this thesis, we presented four ML approaches to identify and classify NFR from unconstrained requirement documents. Furthermore, we investigated the effectiveness of NLP feature extraction techniques on NFR classification. Different NLP techniques were presented in this thesis, include random and word embedding vectorization methods to feed in four differnet ML classifiers.

The experiment was performed using (PURE) public dataset, which consists of unconstrained requirements documents. We had to labeled the requirements sentences in the documents manually by a group of software experts volunteers. Serious criteria were 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 TF and TF-IDF NLP techniques, and the state-of-the-art word embeddings techniques Word2vec and BERT were used for representing requirement sentences into a numerical feature vectors.The TF and TF-IDF exploit the statistical information of the requirement sentence, whereas, the embeddings methods exploit the semantic information of the sentences.

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

From the presented results in this theses, we can draw four primary conclusions. First, CNN approach 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 for NFR representation have a valuable impact on the NFR classification results, and fusion multiple NLP techniques, significantly improves the classification accuracy. Fourth, CNN approach and fusion multi NLP techniques can identify unrevealed NFR efficiently.

7.0.1 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, we plan to expand the number of adopted NFR categories, such as (mutability, solubility, etc..). Second direction,we can continually

investigate the effectiveness of other ML approaches in classification NFR such as recurrent neural network (RNN) and fusion more models.

7.0.2 Threats to validity

One of the study objectives is to collect a sufficient number of requirements sentences contain NFR in various domain. This task include manual classification requirement sentences based on its definition in IEEE-Std 830-1993. In this task we adopted on group of volunteers who are experts in software engineering. Mainly, the success of this process depends on the expert knowledge of the definition of each NFR category. The threats to internal validity in this process include human factor to determining correct identification of NFR categories. In our methodology we performed a set of hard acceptance criteria to reduce this threats as much as possible. This criteria include labeling each sentence by two different experts. Furthermore, the two experts should have the same answer for each review. In case the experts have assigned two different answers for one sentence, we choose the review that has higher confidence. For each answer two levels of confidence adopted in labeling task, low level and high level. We avoided more than two levels of confidence to avoid the neutral choice that most volunteers prefer. And if the two confidences are equal, the preference is for the reviewer with higher experience. If experience are equal the sentence is excluded.

Furthermore, in this study the classification results have been obtained only for 5 NFR categories which don't include all NFR categories. The large number of NFR types and available dataset obliged us to adopted only this number of NFR categories.However, we chose the top 5 NFR types that commonly considered in most domains and identified from among the 14 main NFR. In future-work we can expand in more NFR categories to include in our study.

	TF					TF-IDF					W2V					BERT					
	US	RE	PE	A	SE	Oth	US	RE	PE	A	SE	Oth	US	RE	PE	A	SE	Oth	US	RE	PE	A	SE	Oth	Label
2	0.94629	0.01046	0.03149	0.00648	0.0018	0.00348	0.91999	0.00167	0.00157	0.07247	0.00027	0.00403	0.3784	0.48683	0.05442	0.04277	0.01993	0.01764	0.60685	0.04671	0.33271	0.00092	0.00591	0.0069	US
3	0.00466	0.0079	0.00148	0.01831	0.46882	0.49883	0.00158	0.02089	0.00238	0.02761	0.10682	0.84072	0.04722	0.08801	0.03486	0.39359	0.1276	0.30871	0.00071	0.01926	0.00303	0.23082	0.15827	0.58791	Oth
4	9.00E-05	8.99E-05	0.00025	0.99818	0.00116	2.06E-06	3.28E-06	5.11E-07	5.40E-08	0.99995	4.56E-05	0.0039	0.01539	0.00516	0.01557	0.92773	0.03225	0.00022	8.10E-05	1.20E-05	6.26E-05	0.99835	0.00127	SE
5	0.96587	0.00308	0.00059	0.00752	0.00542	0.01751	0.99997	1.99E-06	4.37E-06	8.43E-06	5.98E-06	6.38E-06	0.31819	0.0388	0.06851	0.29132	0.03645	0.24674	0.70455	1.48E-05	0.01825	0.00272	0.00096	0.27351	US
6	9.05E-05	0.00071	0.00123	0.00137	0.00239	0.99421	6.40E-09	3.58E-08	2.41E-08	3.94E-07	6.89E-06	0.99999	0.00248	0.00205	0.00259	0.00154	0.00351	0.98784	9.97E-05	1.79E-06	0.00077	0.00056	0.00026	0.9983	Oth
7	0.889	0.00751	0.09501	0.00156	0.00545	0.00148	0.95858	0.01405	0.01227	0.00074	0.0138	0.00056	0.28347	0.01039	0.74898	0.02474	0.0165	0.01592	0.69719	0.00051	0.28688	0.00017	0.002	0.01325	PE
8	0.00046	0.00027	0.00054	0.00026	0.00087	0.99759	1.52E-05	3.76E-05	3.64E-05	0.00016	0.00217	0.99758	0.01284	0.00612	0.00849	0.0032	0.01305	0.9563	0.00903	3.31E-05	0.00155	0.00022	0.00594	0.98323	Oth
9	0.02117	0.00117	0.00025	0.0001	0.96874	0.00857	7.84E-06	5.89E-06	8.00E-07	7.82E-08	0.99994	4.53E-05	0.32878	0.0516	0.01186	0.00652	0.45881	0.14243	0.55328	8.78E-05	0.00069	4.46E-07	0.41066	0.03529	SE
10	0.00149	0.0003	3.42E-05	0.00035	0.99711	0.00071	1.03E-05	1.32E-05	2.25E-06	2.35E-07	0.99989	8.18E-05	0.00916	0.01805	0.00628	0.02019	0.92941	0.01691	0.00078	0.00012	6.95E-06	1.82E-05	0.99789	0.00119	SE
11	0.99995	3.05E-06	1.64E-05	1.67E-07	7.19E-06	2.57E-05	0.99974	1.04E-05	0.00015	2.09E-06	9.28E-05	6.07E-06	0.99375	0.0007	0.00193	0.00094	0.00087	0.00182	0.99984	3.64E-07	0.00011	3.05E-11	2.65E-06	5.05E-05	US
12	0.00044	0.00055	0.00033	0.99598	0.0017	0.001	0.00027	9.42E-05	0.00017	0.99811	1.71E-05	0.00134	0.03878	0.01335	0.01711	0.891	0.02852	0.01124	3.12E-05	0.00116	0.00378	0.988	0.00208	0.00495	A
13	0.0092	0.0205	0.00098	0.03458	0.93343	0.00131	0.09439	0.00626	0.00329	0.88472	0.00242	0.00891	0.03118	0.35519	0.02405	0.07495	0.48471	0.02993	0.00169	0.23852	0.01107	0.1471	0.59542	0.00618	SE
14	0.99984	1.72E-05	1.44E-06	1.48E-06	8.76E-05	5.64E-05	1	1.45E-08	2.99E-09	2.32E-08	1.03E-07	2.97E-08	0.97448	0.00364	0.00254	0.00405	0.00533	0.00997	0.99939	1.91E-05	0.00019	5.83E-09	0.00011	0.00029	US
15	0.91816	0.00233	0.00212	0.00042	0.02775	0.04921	0.99953	5.43E-06	4.12E-06	6.30E-06	0.00039	6.63E-05	0.80953	0.01835	0.03823	0.02189	0.04149	0.07051	0.98363	2.08E-05	0.00063	3.24E-07	0.00136	0.01435	US
16	3.53E-07	1.85E-06	1.59E-06	0.99994	2.93E-05	2.33E-05	2.64E-08	5.78E-09	1.79E-08	1	8.32E-11	9.27E-07	0.00145	0.00232	0.00256	0.98676	0.00273	0.00418	1.11E-08	7.09E-06	1.37E-05	0.99989	3.93E-06	8.68E-05	A
17	0.00033	0.00104	0.00023	0.00046	0.99739	0.00054	1.65E-05	3.45E-05	1.51E-06	1.14E-07	0.99992	2.70E-05	0.01154	0.16544	0.00872	0.01372	0.78586	0.01471	0.00078	0.00142	1.20E-05	2.62E-05	0.99735	0.00041	SE
18	5.91E-05	0.00044	0.00015	0.99682	0.00093	0.00159	0.0001	2.18E-05	3.61E-05	0.99913	2.35E-06	0.0007	0.01721	0.03918	0.02226	0.83682	0.02598	0.05855	2.77E-05	0.00168	0.00276	0.97908	0.00261	0.01384	A
19	0.03083	0.00612	0.00051	0.00203	0.95803	0.00248	0.09748	0.29202	0.00326	0.00239	0.59117	0.01368	0.19602	0.15734	0.02868	0.08109	0.39303	0.14384	0.03872	0.03264	0.00156	0.00074	0.92228	0.00406	SE
20	0.00181	0.00053	0.00099	0.0004	0.0014	0.99487	0.00024	0.0001	0.00023	0.00076	0.00092	0.99775	0.05379	0.00744	0.01346	0.00626	0.01466	0.90439	0.00512	1.92E-06	0.00217	1.19E-05	0.00084	0.99186	Oth
21	0.00399	0.00231	0.00021	0.00521	0.98697	0.0013	0.00023	2.51E-05	1.15E-05	7.90E-07	0.99967	6.00E-05	0.14252	0.03852	0.02906	0.10128	0.65189	0.03673	0.44412	0.00077	0.00298	3.39E-05	0.54133	0.01077	SE
22	0.00086	0.99647	0.00038	0.00069	0.00145	0.00015	0.00033	0.9979	0.00011	4.32E-05	0.00154	8.41E-05	0.01263	0.96794	0.0092	0.0066	0.00234	0.00129	0.00052	0.99708	0.0011	0.00014	0.00116	2.54E-06	RE
23	0.00426	0.00208	0.00079	0.01656	0.96599	0.01032	0.00032	0.0002	6.99E-05	1.39E-05	0.99814	0.00127	0.04679	0.04825	0.05177	0.07719	0.67673	0.09928	0.00236	0.00147	0.00092	0.01404	0.93785	0.04336	SE
24	0.00046	0.9984	0.00032	5.95E-05	0.00065	0.00012	0.00024	0.99779	0.00026	6.41E-05	0.00158	7.04E-05	0.00191	0.98706	0.00567	0.00196	0.00138	0.00202	0.00012	0.99892	0.00034	3.73E-05	0.00058	3.82E-06	RE
25	0.01327	0.88263	0.06643	0.00726	0.01518	0.01523	0.00996	0.81926	0.0249	0.05936	0.01798	0.06854	0.02914	0.70647	0.04496	0.06478	0.07436	0.08029	0.00375	0.96463	0.01768	0.00378	0.00942	0.00074	RE
26	0.03771	0.62984	0.01201	0.00306	0.05848	0.25891	0.00804	0.45089	0.00225	0.00293	0.42991	0.10597	0.06728	0.5144	0.05112	0.01435	0.0389	0.31394	0.10171	0.60355	0.06825	1.87E-05	0.01675	0.20971	RE
27	0.0268	0.96658	0.00285	0.00117	0.00226	0.00035	0.00279	0.99435	0.00067	0.00021	0.00176	0.00023	0.05796	0.89179	0.01284	0.02206	0.00783	0.00753	0.00172	0.99463	0.00208	0.0003	0.00123	3.24E-05	RE
28	0.0021	0.00546	0.00147	0.0022	0.93382	0.05496	8.72E-05	0.00013	2.72E-05	6.88E-06	0.99865	0.0011	0.04967	0.27471	0.06593	0.03104	0.28062	0.29803	0.03632	0.00623	0.00855	0.0009	0.83467	0.11333	SE
29	4.17E-05	1.69E-05	0.99989	1.17E-05	2.39E-05	1.81E-05	8.54E-05	1.45E-05	0.99985	2.94E-06	2.13E-05	2.19E-05	0.00159	0.00072	0.99409	0.00084	0.00225	0.00051	0.00025	8.99E-07	0.99965	3.08E-06	2.61E-06	9.96E-05	PE
30	0.00652	0.0018	0.99085	0.00018	0.00057	8.20E-05	0.00057	0.00034	0.99872	3.79E-05	0.0002	0.00013	0.00582	0.00849	0.9819	0.00087	0.00169	0.00124	0.00108	0.00018	0.99854	5.98E-05	2.97E-06	0.00013	PE
31	0.99354	0.00087	0.00078	0.00015	0.00374	0.00093	0.94879	0.00047	0.0005	0.00234	0.03372	0.01418	0.95628	0.00375	0.00788	0.01258	0.00679	0.01273	0.96576	0.00088	0.00664	0.0002	0.00986	0.01666	US

FIGURE 7.1: Sample results for fusion front-end model

Bibliography

[1] Ravinder Ahuja et al. "The impact of features extraction on the sentiment analysis". In: *Procedia Computer Science* 152 (2019), pp. 341–348.

[2] *AI Programming: 5 Most Popular AI Programming Languages | Existek Blog*. Feb. 6, 2018. URL: https://existek.com/blog/ai-programming-and-ai-programming-languages/ (visited on 12/31/2019).

[3] Sousuke Amasaki and Pattara Leelaprute. "The Effects of Vectorization Methods on Non-Functional Requirements Classification". In: *2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE. 2018, pp. 175–182.

[4] Muhammad Zubair Asghar et al. "A review of feature extraction in sentiment analysis". In: *Journal of Basic and Applied Scientific Research* 4.3 (2014), pp. 181–186.

[5] Cody Baker et al. "Automatic Multi-class Non-Functional Software Requirements Classification Using Neural Networks". In: *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*. Vol. 2. IEEE. 2019, pp. 610–615.

[6] Vikkiraman Balarajan and Punithan Balarajan. "A Short Review: Software Requirements". In: *Proceedings of the Informatics Conference*. Vol. 2. 2. 2016.

[7] Manal Binkhonain and Liping Zhao. "A review of machine learning algorithms for identification and classification of non-functional requirements". In: *Expert Systems with Applications* (2019).

[8] T Bures et al. "Requirement specifications using natural languages". In: *Technical Report D3S-TR-2012-05* (2012).

[9] Agustin Casamayor, Daniela Godoy, and Marcelo Campo. "Identification of non-functional requirements in textual specifications: A semi-supervised learning approach". In: *Information and Software Technology* 52.4 (2010), pp. 436–445.

[10] Lawrence Chung et al. *Non-functional requirements in software engineering*. Vol. 5. Springer Science & Business Media, 2012.

[11] Jane Cleland-Huang et al. "The detection and classification of non-functional requirements with application to early aspects". In: *14th IEEE International Requirements Engineering Conference (RE'06)*. IEEE. 2006, pp. 39–48.

[12] IEEE Standards Coordinating Committee et al. "IEEE Standard Glossary of Software Engineering Terminology (IEEE Std 610.12-1990). Los Alamitos". In: *CA: IEEE Computer Society* 169 (1990).

[13] Alex Dekhtyar and Vivian Fong. "RE data challenge: Requirements identification with Word2Vec and TensorFlow". In: *2017 IEEE 25th International Requirements Engineering Conference (RE)*. IEEE. 2017, pp. 484–489.

[14] Jacob Devlin et al. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805* (2018).

[15] Ingo Feinerer et al. "Package 'wordnet'". In: *R package Version 0.1-11. Available at https://cran. r-project. org/web/packages/wordnet/wordnet. pdf, accessed* 1 (2017).

[16] Alessio Ferrari, Giorgio Oronzo Spagnolo, and Stefania Gnesi. "PURE: A dataset of public requirements documents". In: *2017 IEEE 25th International Requirements Engineering Conference (RE)*. IEEE. 2017, pp. 502–505.

[17] Alessio Ferrari, Giorgio Oronzo Spagnolo, and Stefania Gnesi. "Towards a Dataset for Natural Language Requirements Processing." In: *REFSQ Workshops*. 2017.

[18] Maria Haigh. "Software quality, non-functional software requirements and IT-business alignment". In: *Software Quality Journal* 18.3 (2010), pp. 361–385.

[19] Rani Horev. *BERT Explained: State of the art language model for NLP*. Medium. Nov. 17, 2018. URL: https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270 (visited on 08/20/2020).

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

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

[22] "IEEE Recommended Practice for Software Requirements Specifications". In: *IEEE Std 830-1993/1998* (1998), pp. 1–40.

[23] Jung-Yi Jiang, Shian-Chi Tsai, and Shie-Jue Lee. "FSKNN: multi-label text categorization based on fuzzy similarity and k nearest neighbors". In: *Expert Systems with Applications* 39.3 (2012), pp. 2813–2821.

[24] Bernard EM Jones. "Exploring the role of punctuation in parsing natural text". In: *Proceedings of the 15th conference on Computational linguistics-Volume 1*. Association for Computational Linguistics. 1994, pp. 421–425.

[25] James Joyce. "Bayes' Theorem". In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta. Spring 2019. Metaphysics Research Lab, Stanford University, 2019. URL: https://plato.stanford.edu/archives/spr2019/entries/bayes-theorem/ (visited on 07/13/2020).

[26] Eric Knauss et al. "Supporting requirements engineers in recognising security issues". In: *International Working Conference on Requirements Engineering: Foundation for Software Quality*. Springer. 2011, pp. 4–18.

[27] Zijad Kurtanović and Walid Maalej. "Automatically classifying functional and non-functional requirements using supervised machine learning". In: *2017 IEEE 25th International Requirements Engineering Conference (RE)*. IEEE. 2017, pp. 490–495.

[28] Xiaoli Lian and Li 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)*. IEEE. 2015, pp. 191–200.

[29] *Logistic Regression — ML Glossary documentation*. URL: https://ml-cheatsheet.readthedocs.io/en/latest/logistic_regression.html (visited on 08/29/2020).

[30] Ashish Kumar Luhach et al. *Advanced Informatics for Computing Research: Third International Conference, ICAICR 2019, Shimla, India, June 15–16, 2019, Revised Selected Papers, Part I*. Google-Books-ID: LAuwDwAAQBAJ. Springer Nature, Sept. 16, 2019. 492 pp. ISBN: 9789811501081.

[31] Long Ma and Yanqing Zhang. “Using Word2Vec to process big text data”. In: *2015 IEEE International Conference on Big Data (Big Data)*. IEEE. 2015, pp. 2895–2897.

[32] Anas Mahmoud and Grant Williams. “Detecting, classifying, and tracing non-functional software requirements”. In: *Requirements Engineering* 21.3 (2016), pp. 357–381.

[33] Arun S Maiya. “ktrain: A Low-Code Library for Augmented Machine Learning”. In: *arXiv preprint arXiv:2004.10703* (2020).

[34] Tom M. Mitchell. *Machine Learning*. Google-Books-ID: EoYBngEACAAJ. McGraw-Hill, 1997. 414 pp. ISBN: 978-0-07-115467-3.

[35] *Natural Language Processing - Syntactic Analysis - Tutorialspoint*. URL: https://www.tutorialspoint.com/natural_language_processing/natural_language_processing_syntactic_analysis.htm (visited on 01/29/2020).

[36] *NumPy — NumPy*. URL: https://numpy.org/ (visited on 12/30/2019).

[37] Md Abdur Rahman et al. “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*. ACM. 2019, pp. 25–30.

[38] Denni Aldi Ramadhani, Siti Rochimah, and Umi Laili Yuhana. “Classification of non-functional requirements using semantic-FSKNN based ISO/IEC 9126”. In: *Telkomnika* 13.4 (2015), p. 1456.

[39] Abderahman Rashwan, Olga Ormandjieva, and René 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*. IEEE. 2013, pp. 381–386.

[40] *Sentence: Definition & Types | Learn English*. URL: https://www.learngrammar.net/english-grammar/sentence-definition-n-types (visited on 01/15/2020).

[41] Unnati S Shah, Sankita Patel, and Devesh Jinwala. "Specification of Non-Functional Requirements: A Hybrid Approach." In: *REFSQ Workshops*. 2016.

[42] Vibhu Saujanya Sharma, Roshni R Ramnani, and Shubhashis 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*. ACM. 2014, pp. 1–8.

[43] John Slankas and Laurie Williams. "Automated extraction of non-functional requirements in available documentation". In: *2013 1st International Workshop on Natural Language Analysis in Software Engineering (NaturaLiSE)*. IEEE. 2013, pp. 9–16.

[44] *SVM | Support Vector Machine Algorithm in Machine Learning*. Analytics Vidhya. Sept. 12, 2017. URL: https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/ (visited on 08/29/2020).

[45] Mohammed Terry-Jack. *NLP: Everything about Embeddings*. Medium. June 3, 2019. URL: https://medium.com/@b.terryjack/nlp-everything-about-word-embeddings-9ea21f51ccfe (visited on 06/04/2020).

[46] László Tóth and László Vidács. "Study of various classifiers for identification and classification of non-functional requirements". In: *International Conference on Computational Science and Its Applications*. Springer. 2018, pp. 492–503.

[47] Marc Toussaint. "Introduction to machine learning". In: *Online Course Notes. July* (2016).

[48] Abinash Tripathy, Ankit Agrawal, and Santanu Rath. "Requirement Analysis using Natural Language Processing". In: Dec. 2014.

[49] Axel Van Lamsweerde. *Requirements engineering: From system goals to UML models to software*. Vol. 10. Chichester, UK: John Wiley & Sons, 2009.

[50] Radu Vlas and William 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*. IEEE. 2011, pp. 1–10.

[51] Jonas Winkler and Andreas Vogelsang. "Automatic classification of requirements based on convolutional neural networks". In: *2016 IEEE 24th International Requirements Engineering Conference Workshops (REW)*. IEEE. 2016, pp. 39–45.

[52] Xusheng Xiao et al. "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*. ACM. 2012, p. 12.

[53] Wen Zhang et al. "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.