Journal of Education and Science (ISSN 1812-125X), Vol: 31, No: 03, 2022 (66-90)

Classification of Software Systems attributes based on quality factors using linguistic knowledge and machine learning: A review

Abdulrhman M. Ali[1*], Nada N. Saleem[2]

[1*2]Software Department, College of Computer Science and Mathematics, University of Al Mosul, Mosul, Iraq

E-mail: [1*] abdurhman55331992@gmail.com , [2] nada_n_s@uomosul.edu.iq

(Received May 23, 2022; Accepted August 01, 2022; Available online September 01, 2022)

DOI: 10.33899/edusj.2022.134024.1245, © 2022, College of Education for Pure Science, University of Mosul.
This is an open access article under the CC BY 4.0 license (http://creativecommons.org/licenses/by/4.0/)

Abstract

Both the functionality and the non-functionality for what the software system does and does not do within software systems requirements are documented in a Software Requirements Specification (SRS). In requirements engineering, system requirements classify into several categories such as functional, quality and constraint classes. Therefore, we evaluate several machine learning approaches as well as methodologies mentioned in previous literature in terms of automatic requirements extraction, then classification is performed based on methodically reviewing many previous works on software requirements classification to assist software engineers in selecting the best requirement classification technique. The study aims to obtain answers for several questions: “What were machine learning algorithms used for the classification process of the requirements?”, “How do these algorithms work and how are they evaluated?”, “What methods were used for extracting features from a text?”, “What evaluation criteria were used in comparing results?”, and “Which machine learning techniques and methods provided the highest accuracy?”.

Keyword: Software requirements, functional requirements, non-functional requirements, classification, and machine learning

تصنيف سمات أنظمة البرمجيات بالاعتماد على عوامل الجودة باستخدام المعرفة اللغوية والتعلم الآلي: مراجعة

عبدالرحمن مصطفى علي[1*], ندى نعمت سليم[2]

[1*2] قسم البرمجيات، كلية علوم الحاسوب والرياضيات، جامعة الموصل، الموصل، العراق

الخلاصة:

يتم توثيق كل من متطلبات انظمة البرمجيات الوظيفة و غير الوظيفية ما يفعله النظام وكذلك ما لا يفعله في مستند مواصفات متطلبات البرمجيات (SRS).

و في هندسة المتطلبات تصنف متطلبات النظام إلى فئات منها الوظيفية و منها المتعلقة بالجودة والقيود. لذلك في هذه الدراسة قمنا بتقييم العديد من مناهج ومنهجيات التعلم الآلي المستخدمة في المؤلفات المنشورة حول استخراج المتطلبات تلقائيا ثم تصنيفها من خلال المراجعة المنهجية للعديد من المقالات المتعلقة بتصنيف متطلبات البرمجيات لمساعدة مهندسي البرمجيات من اختيار اكفأ تقنية لتصنيف المتطلبات، لذلك تهدف الدراسة إلى الحصول على إجابة لعدة أسئلة تتعلق بما يلي: ما هي خوارزميات التعلم الآلي التي تم استخدامها في

عملية تصنيف المتطلبات، وكيف تعمل هذه الخوارزميات وكيف يتم تقييمها, وما هي الأساليب المستخدمة لاستخراج الميزات من النص، وما هي معايير التقييم التي تم استخدامها في مقارنة النتائج، وما هي تقنيات وطرق التعلم الآلي التي اعطت نتائجها اعلى درجات الدقة.

الكلمات المفتاحية : متطلبات البرمجيات ، المتطلبات الوظيفية ، المتطلبات غير الوظيفية ، التصنيف ، التعلم الآلي

1. Introduction

Software engineering is defined as a systematic disciplinary approach for the development and maintenance of the software. It provides an analysis of customer requirements, design, in order to satisfy those requirements. Therefore, Requirements Engineering (RE) is considered a special methodology through which the context of a problem is determined. Moreover, it determine the customer's needs and the specifications that meet the requirements of customers in the context[1]. It should be noted that the objective of requirements specifications is to establish the qualities that the system must fulfill in order to be approved. They are used in various requirements engineering (RE) procedures to record outcomes[2].

The features of Software requirements can be discovered before creating the software products[3], thus it represents customer needs and expectations from software systems. Likewise, the capability of a software system that should be possessed by the components of the software in order to satisfy customer needs that are often specified[4]. The classification of Software Requirements (SRs) is accomplished by software professionals in order to identify the requirements that they require or are directly concerned with[5]. The major categories of SRs include both Functional Requirement (FR) and Non-Functional Requirement (NFR). The IEEE Standard for Software Engineering Terminology defines FRs as a function of a system and its components perform [6], consequently NFR, have a system limitation. The process of requirements engineering can classifying system requirements into FRs and NFRs [7] which now universally considered as normal procedure. Despite of the fact that today's requirements are clearly understood and stated [8], the mechanical categorization of natural language needs into distinct FRs and NFR subclasses remains a difficult task [9]. Due to the primary cause for this is that stakeholders and requirements architects use various vocabulary, phrasing, and sentence structures to convey the same set of requirements [10, 11].

Racially, data analysis and intelligent applications have incorporated artificial intelligence (AI) and machine learning (ML)[12]. Whereas ML used to build intelligence systems with capabilities to learned and enhanced automatically from experience [13, 14] [15]. Generally, there are four types of learning algorithms: supervised, unsupervised, semi-supervised, and reinforcement learning [16] .

Classification, regression analysis, clustering, feature engineering and deep learning are consider (techniques) and (methods) in ML[17]. Thus, ML techniques provide high accuracy of the predicted results in data classification performance [18].

Particularly SE is constructing high quality system software's that use an efficient and systematic methods [19, 20]. Recurrently many researches are solved complex problems used ML in SE. Therefore ML approaches can be used in SE life cycle stages of system software's including: requirements analysis, design, assessment, implementation, testing and reengineering, to improve the performance of defect prediction and cost reduction [20, 21].

Several applications of ML methods in SE are: specification extractions, requirements classifications, design pattern recognition, program code generation, test case generation, defect predicate, effort estimation, etc. [22, 23]

Machine learning techniques are applied to text data just as they are used to other forms of data, such as images. Moreover, machine learning is used in the process of classifying the attributes of software systems on the basis of quality factors using linguistic knowledge and machine learning.[24].

Our paper aims to review a number of previous studies on the classification of software requirements. We try to compare the techniques used in the classification process to determine the most appropriate technique that provides better results (accuracy, precision, recall, and F-measurement) in the process of classifying software system features on the basis of quality factors using machine learning.

This paper is divided into sections as follows: section2 represent the theoretical background by explain the Software Systems Requirements, ML techniques and methods used in the Software Requirements Classification, Text vectorization models and feature extraction with performance measures used to evaluate the classifier. Methodology in Section3 state and explain related work with and view literature review on machine learning used in software requirement classification with various data seats which represent different types of software requirements and Summarization of relevant published papers, Discussions and conclusions in sections 4 and 5 respectively.

2. Research Method

This section describe the problem background theory which provide an overview of the software systems requirements (SSR), ML techniques used for classifier of SSR and text features extraction with performance classifier measurements.

2.1 Software Systems Requirements (SSR)

Within the software engineering industry, requirements engineering (RE) is consider one of the most natural language-intensive fields. As a result of, over the years ago, whereas many of previous works have been produced to automate the analysis of natural language artifacts important to RE, such as requirements documents, application reviews, privacy rules, and social media information relating to software goods. Recently the spread of game-changing natural language processing (NLP) techniques and platforms have piqued RE researchers' interest. However, there is currently no reference framework that provides a comprehensive grasp of the subject of NLP for RE [25].

Requirements operations include capturing both FRs and NFRs, which describe what the system must perform and how it must be accomplished, respectively. Business analysts and domain specialists gather and document FRs. On the other side, technology experts ,make architectural decisions. As a result, the croups of knowledge for FRs and architectural solutions are kept distinct. FRs are considered that are criticized, high-risk, fickle and entails costly reworking or has legal implications [26]. Therefore, when software quality is spoken, we should refer to NFRs term. Indeed NFRs are important limitations on a software system's development and behavior. Security, performance, availability, extensibility, and portability are just a few of the properties they specify. These characteristics are crucial in architectural design [27]. Thus, the existing issues with the concept of NFR can be separated into three categories: definition issues, classification issues, and representation issues [28].

2-2 Techniques and methods used in the Software Requirements Classification

Machine learning is a subfield of AI, it is a data analysis method that automates analytical models. The algorithm can generate an output for an input it has never seen before without the need for human interaction. Moreover, machine learning algorithms which learn from input/output pairs are known as supervised learning algorithms. For each example they learn from, a "teacher" gives supervision to the algorithms in the form of desired outputs. [29]. The input data is only known in an unsupervised algorithm, and the method is given no known output data. Although these techniques have many good

uses, they are difficult to comprehend and evaluate [30]. Various ML algorithms were used to automatically classify software requirements in review papers**:**

Latent Dirichlet Allocation (LDA) algorithms, documents are categories based on the frequency of word co-occurrences. [31].

The Biterm Topic Model (BTM) learns topics by studying patterns of words and models subjects based on word co-occurrence patterns (eg the biterm) [32]. Recent research on the categorization of short text documents confirms that, BTM has a superior ability to represent short and sparse text, such as that found in requirements specifications.

Naive Bayes is a pretty famous supervised learning technique for binary classes [33]. It is based on the Bayes theorem that makes considerable feature independent assumptions. Basically it's straightforward, and effective, and unlike most other classes, it doesn't require a vast training set. It based on Bayes' theorem to forecast data that isn't visible [34].

SVM (Supporting Vector Machines): It is one of the supervised classification and regression algorithms which is characterized by its robustness and flexibility[35]. SVM is a versatile and powerful machine learning algorithm that can perform linear and nonlinear classification, regression, and outlier identification. Classification method creates a hyper-linear plane with a maximum margin between two classes. This margin leaves few opportunities to separate the data from the sample, therefore little opportunities for new cases to be misclassified [36].

MNB (Naive Bayes Multinomial): It's a method which calculates the data set conditional probability. The input features in MNB are assumed to be independent of one another (independence under certain conditions). A specific variant of Naive Bayes Multinomial is used to classify documents and text [37].

k-NN (k-Nearest Neighbor) : is a data categorization system based on the neighbor principle, which states that examples within a data collection should be found near other instances with comparable characteristics. [38]. For regression issues, the algorithm classifies incoming data by computing the distance between it and the instances already in the database, then selecting the k closest cases and finding their mean, or getting the position [39] .

LR (Logistic Regression): Is a regression approach used for the estimation of the likelihood of a specific instance that belonging to a specific class [40]. The independent variable is used in the prediction of dependent variable. Therefore when the dependent variable has only two classes, Binary logistic regression is used. However, when the dependent variable has more than two categories Polynomial logistic regression is used [41].

long short-term memory (LSTM): Recurrent Neural Network (RNN) which employs memory blocks to solve the vanishing scaling problem. The model's first layer, the input layer, receives preprocessed data in time steps. In order to produce feature vectors, each component is first given to the embed layer, thus the LSTM's hidden layer only follows the forward direction. The LSTM has three primary gates for controlling cell state and updating weights: input, forget, and output. [42].

BiLSTM (Bidirectional long short term memory): BiLSTM contains two hidden layers, where are coupled to the input and output. Thus take use of the learning information tokens, BiLSTM contains a front LSTM layer as well as a rear LSTM layer, and better predictions can be achieved. The layers of LSTM are Stacking best way to take advantage of BiLSTM. From t=1 through T, the front layers are iterated. The back layers, on the other hand, are repeated from t=T to 1 [43] .

convolutional neural network (CNN): Local features can be produced by applying the concept of (convolutional neural networks) [44]. Varying vertical localities allow filters of different widths L=3, 4, and 5 through the usage of filters with a width set by the size of the word embed vector. This makes it useful for learning many features[45].

Recurrent Neural Networks (RNN): is one of the modern algorithms for processing sequential data. Since it has internal memory as it is the first algorithm that remembers its input[46], it is therefore suitable for machine learning problems involving sequential data. It is the first algorithm that has made breakthroughs in deep learning over the past few years. It is a type of powerful neural network [47].

Gated Recurrent Unit (GRU) : GRU is a sort of RNN that differs from LSTM in that it transforms data quicker. It also necessitates fewer variables [48]. There are two sorts of gates: update and reset. Specifically, it solely deals with unit information because of there is no memory to store it. It should be noted that the amount of data to be refreshed is determined by the update gateway, and the amount of previous data to be forgotten is determined by the reset gateway. The input data is received and the previously calculated state is deleted when the gate is set to zero. [49] .

Singular Value Decomposition (SVD): This technique was developed to process natural language and is frequently used in the field of information retrieval. Therefore in order to recover the most useful attributes for expression, it divides the rectangular matrix A(m*n) into three smaller matrices. During dealing with vast volumes of data and declining dimensions, SVD offers the mathematical framework for text classification and latent semantic indexing. It removes data clutter and repetition from high-dimensional data, resulting in cleaner data after deleting words that appear in almost every page.[50].

MaxEnt: MaxEnt, also known as maximum entropy or multinomial logistic regression, is a multi-layer classification technique. MaxEnt uses a linear collection of features together with some review criteria to calculate the likelihood of each sort of classification review[51].

Decision tree algorithm: is a classifying technique [52]. It presupposes that all features are boundary discrete and that the class classification is represented by a single objective feature (ie the leaves of the tree)[51] [53].

J48: The J48 algorithm is one of the top machine learning algorithms for categorizing and continually checking data. It is used to classify different applications with accurate results when classifying. It breaks down each aspect of the information into sub-groups to base a particular decision on. It then looks at the standard data gain that actually splits the information by selecting an attribute[54].

2-3 Text vectorization models and feature extraction

Machine learning techniques require numerical inputs to perform classification. Software requirements are documented as text dataset, therefore in order to build a classifier ML model, we need to convert text data into numerical vectors as extracting features using word embedding or vectorization model. Several techniques were used to convert text data into numerical vectors:

1-BOW(Bag of Words): is a straightforward and efficient method for extracting information from text sources. This methodology converts text documents to numeric vectors, yielding a vector for each document that is the iteration of all highlighted words in the document vector space [55]. The vector Xj = (x1,j.. xi,j.. xn,j) expresses for requirement "j" using BoW, where xi,j represents feature's weight I computed by iteration of word I in requirement "j" and "n" represents the number of items in the dictionary. The manually specified criteria are then converted into vectors and used to train classifiers using supervised machine learning algorithms. [56].

2-Term Frequency - Inverse Document Frequency (TF-IDF) : The TF-IDF method combines between two main measures: the initial frequency of a term in a given document and the inverse of the document frequency for each term. These measures are calculated by dividing the total number of documents by the document frequency of each term. Thus, they applying to the result logarithmic scaling [57]. It can be represented mathematically as shown in Equation 1:

$$idf_i = \log \frac{total\_requirments}{total\_requirments\_with\_term\_i} \quad (1)$$

When the two scales are combined, It can be represented mathematically as shown in Equation 2:

$$TF\text{-}IDF(Term_{i,j}) = tf_{i,j} \times idf_i \quad (2)$$

For the "I" term and the "j" document, t f indicates the term frequency and id f represents the inverse of the document frequency.

3-Chi square (CHI^2), A statistical test analyzes the deviation from an expected distribution when the occurrence of a characteristic is assumed to be independent of a category value. whereas the amount of independence between the terms t and a class is measured [58]. Know mathematically through Equation 3:

$$x^2(t,c) = \frac{N \times (AD-CB)^2}{A+C) \times (A+B) \times (C+D)} \quad (3)$$

where N is the total number of documents, A is the number of times t and c occur together, B is the number of times t occurs without c, C is the number of times c occurs without t, and D is the number of times neither c nor t occurs[59].

4- Part-Of-Speech Tagging The technique of encoding a word into a textual in accordance with a part of speech based on both its meaning and context is known as (POS tagging, also known as grammatical tagging) [60] .

5- Word2Vec: is consider a deep learning-based predictive model within the category of unsupervised models that is used to compute and create high-quality dense, distributed. Words are represented as continuous vectors that capture contextual and semantic similarities [61].

6- AUR-BoW: When user comments are broken into sentences, most user comments are too short, therefor when text is categorized, the text of the workbook is too short. In order to bypass this problem, several similar words are added to user reviews (comments). This classification technique is called AUR-BoW [59].

7-Bagging: Bagging, also called bootstrap clustering, is a widely used group learning method for reducing variance within a noisy data set. The working mechanism is as follows: Initially, a data's random sample is selected in the set of training with replacement - that is, it is possible to select individual data points a number of times. These weak models are trained separately after producing a huge number of data samples, and depending on the purpose — regression or classification. For example — the mean or majority of those predictions produces a more accurate estimate[62].

2-4 Performance Metrics

A set of performance measures is primarily used to evaluate a classifier's performance in machine learning tasks. Performance verification is perform through static mathematical algorithms that evaluate the results of the user model's predictions with the real values in the dataset being used [63]. We highlight the outline set of a set of measures that are considered when evaluating machine learning tasks.

Matrix of confusion

It's frequently utilized in binary classification tasks, the matrix of confusion shows how good the items in the set of validation are ranked as well as providing more detail about the performance of the classifier. The following table shows the different nomenclature that can be called when class prediction, by giving the difference between the true and predicted values [64] as shown in Figure 1.

		True value	
		positive	negative
Predicted value	positive	TP	FP
	negative	FN	TN

Figure1. Matrix of confusion legend

True-positives (TP) are samples that have positive indications and are correctly expected to be positive. False-positive (FP) are negative samples that are mistakenly projected as positive. True-negatives (TN) are samples that have a negative rating despite being accurately anticipated to be negative. False-negatives (FN) are samples that were anticipated as negative but ended up being positive. [65].

Accuracy

The percentage of samples correctly identified overall is one measure of accuracy for machine learning activities. [66]. If the validation set's size is N, as in Equation 4:

$$accuracy = \frac{TP + TN}{N} \qquad (4)$$

Or through the following equation 5:

$$accuracy = \frac{TP+TN}{TP+FP+TN+FN} \qquad (5)$$

Due to accuracy does not inform how well a model grades a particular classification, it is considered a primitive measure. If a validation set contains four positive samples and six negative samples, and the classifier predicts that all ten samples will be negative, the classifier achieves an apparent accuracy of 60%. However on closer inspection, the model graded everything negatively and failed to capture the features that distinguish the two groups, giving it a poor score.[67] .

Precision

Precision is a machine learning job performance metric that relates the number of samples correctly classified to the total number of samples[68]. The total number of accurate classifications is divided by the total number of classifications performed [69]. The ratio of true positives (TP) to positives (TP + FP) is another name for this metric: The ratio of true positives (TP) to positives (TP + FP) is another name for this measure [63] As in equation 6:

$$Precision = \frac{TP}{TP + FP} \qquad (6)$$

Recall

Recall is the percentage of samples with positive markers that were successfully predicted [70]. Also referred as true positive sensitivity or rate , recall is a measure of how well a classifier is at correctly predicting actual positive samples [71]. It can be calculated using the following mathematical equation As in equation 7:

$$Recall = \frac{TP}{TP + FN} \qquad (7)$$

F1-score

The F1-score: is test accuracy metric. The F1-score is a calculated weighted average of recall and precision [72]. It is often appropriate to combine the two scales of precision and recall into a single scale known as F1-score (also called F-measure), especially used to compare two classifiers .The harmonic means of accuracy and recall is the F1-score. While the standard average takes all numbers into account equally, the harmonic average gives lower values greater weight. A high F1-score is only obtained by a classifier if both recall and precision are good. [73]. The value of F1-score can be found using the mathematical equation 8 :

$$f1 - score = \frac{2TP}{2TP + FP + FN} \quad (8)$$

Fβ-score represent the harmonic weighted average of recall and precision, measuring the relative importance of the two[52]. As in equation 9:

$$F_{B=(1+B^2)}.\frac{Recall.precision}{B^2.Recal+precision} \quad (9)$$

The scale unbalance preference for recall or precision when = 1, which means that the F1 score is highest when recall = precision = 1 and poorest when recall = precision = 0.

2-5 Methodology

Classification of SSR by using ML techniques has been greatly used by researchers. Applying ML algorithms less time is spend by experts with more accuracy.

Zahra Shakeri Hossein et al [74] looked over 625 requirements from the "Open Science tera-PROMISE" collection. The goal of their research was to figure out how to enhance automated requirement classification in FR and NFR. As well as how well various machine learning algorithms perform in the classification process. Their working methodology was a processing strategy. They discovered that preprocessing improved the effectiveness of the present classification technique by standardizing and normalizing criteria prior to using classification algorithms. They also looked at how curriculum like Latent Dirichlet Allocation, Biterm Topic Modeling, and Nave Bayes for subclassifying NFRs performed. Advantages: Using a preprocessing method in the FR/NFR classification process, as well as subclassifying NFR into subcategories, can result in greater classification accuracy.

Bruno Cordeiro Mendes and Edna Dias Canedo [63] classify software requirements into FR and NFR with subcategories using machine learning techniques. In the requirements classification task, the researchers compare different text feature extraction techniques using machine learning algorithms. Techniques for selecting features BoW, TF-IDF, and CHI2 were used in this study, and the classification algorithms: Logistic Regression (LR), SVM, MNB and KNN were used. The PROMISE_exp data set used to perform the search, and using TF-IDF after that for differentiation needs. Better classification result provided by LR with accuracy up to 0.91. An advantages used for binary classification, non-functional requirements classification, the combination of TF-IDF with LR has the best performance metrics. Disadvantages:When the number of requirements for some labels is less in the group unbalanced data, automatic classification performance suffers.

Nouf Rahimi et al[75] published a study aimed at categorizing software requirements (SRs), binary classification of SRs into FRs or NFRs, and multi-label categorization of both FRs and NFRs into various experimental categories. With a combination of four different deep learning models: The strategy employed three group methods: accuracy as a weight ensemble, mean ensemble, and accuracy per class as a weight ensemble, as well as long short term memory (LSTM), bidirectional long short term memory (BiLSTM), a gated recurrent unit (GRU), and a convolutional neural network (CNN). Models were trained and tested using the PROMISE dataset. The two-phase classification system outperformed the single-phase classification approach. The accuracy of the one-phase system was 92.56 %t, while the binary phase accuracy of the two-phase classification system was 95.75 % meanwhile the multiclass classification phase accuracy was 93.4 %. Therefore Advantages: the creation and distribution of SR rating systems that will assist software engineers, developers, and analysts in creating complete SRs for the development of reliable software systems. While disadvantages: is the suggested model's as well as classification systems' limitations; it can only support one language, which is written in a structured document, and sentences can be recovered from SRs by dealing with the extracted structured sentences .

Muhammad Mahmoud Al-Tarawneh [76] presented the relationship between requirements engineering and NLP, in order to classify binary requirements into FRs and NFRs . This class used natural language processing dataset as well as single value analysis (SVD) TERA-PROMISE was used. The author present five models employed are TF, TF-IDF, TF-IDF-CF, Bigram, and Trigram. Advantages: This cosine distance was calculated using the SVD model. This cosine distance, trigram had the best representation model. Disadvantages: The high frequency words in documents belonging to the same category are dependent on the requirements classification, that means the frequency represents both the document and the category at the same time, and this is the method's weakness..

Ishrar Hussain et al[6] present a work with the goal of discovering NFR phrases utilizing a text classifier with a part of speech (POS) tagger and using natural language processing (NLP) approaches to software requirements engineering. The authors Using 10-fold cross-validation on the identical data used in the literature. The search results were accurate to 98.56 %. Advantages: software analysts can indicate NFRs in SRS text documents to users to avoid additional supervision in the development process, which might result in poor quality of the final product and, eventually, project failure. Disadvantages: A complete prototype is not possible to make.

Kortanovic et al[77] used meta-data, lexical, and syntactical characteristics, as well as the support vector machines (SVM) method, to create and evaluate a supervised machine learning technique . The authors depends on these techniques to categorize software requirements into FR, NFR, and subcategories of NFR. Therefore the authors made use of the PROMISE repository. Advantages: Rather than the data set for this challenge, requirements might be gathered from user comments. User evaluations are typically brief, unstructured, and infrequently follow language and punctuation requirements, resulting in reduced accuracy.

Tamai and Taichi Anzai[78] used Machine learning technology. The QRMiner tool was developed in order to analyze Quality Requirements statements from software requirements specifications (SRS) and categorize them into quality characteristics attributes. Thirteen documents were used in the case studies. SRS that was created for real-world applications in mind. Advantages : the use of the latest machine learning, deep learning and Doc2Vec technologies, which have greatly enhanced the

performance of QRMiner, and the use of open source requirements documents, rather than data from student projects or open source projects . Disadvantages: the use of SRS must be written in English only.

Hui Yang & Peng Liang[79] proposed an approach where the requirements information is automatically identified and categorized into FRs and NFRs from user reviews. Using both TF-IDF and NLP (regular expression) intervention. Human selection of keywords to define and categorize requirements. User evaluations from the popular APP iBooks in the English language app store have verified the recommended technique. Advantages: It is useful and practical for APP developers to elicit requirements from user reviews. Disadvantages: It is not possible to prioritize specific requirements that are further categorized to show their importance when hundreds and thousands of requirements flow to developers.

Walid Maalej et al[80] offered several possible methods for classify application reviews such as : user experiences, text ratings, bug reports, and feature requests. Descriptive data such as time and star ratings, text classification, sentiment analysis techniques and natural language processing were used for the review. These series of studies were carried out to assess the accuracy of the approaches utilized and to compare them to simple series similar to. Totally it was discovered that simply having metadata leads to poor categorization accuracy. When combined with basic text classification and natural language text preprocessing - notably with capital and lowercase letters - the classification precision and recall for all review categories rose to 88–92 percent and 90–99 percent, respectively. Single multiclass classifiers were outperformed by multiple binary classifiers. Advantages: Aids in the filtering of evaluations relevant to certain stakeholders like as developers, analysts, and other users. Disadvantages: Stopword removal and lemmatization should be employed in text pretreatment NLP, since stopword removal might lower classification accuracy.

Jonas Winkler and Andreas Vogelsang [2] proposed a proprietary approach to automatically classifying content elements to the NLR specification as "requirement" or "information". This was done through the use of convolutional neural networks. The dataset used was Doors database related to an industrial partner. Advantages: This method can be used for the purpose of classifying content items in documents that have not been categorized before or for the purpose of analyzing documents that are already categorized as well as identifying the author for possible incorrect classifications of content items for the document. Disadvantages: only providing the user with actual results but without explaining why the content item was incorrectly categorized, and accuracy and recall are not reasonably high .

Abderahman Rashwan et al[81] offered a method for doing automated analysis of SRS documents for different forms of NFR utilizing Support Vector Machine (SVM) technology, as well as the Supporting Vector Machine (SVM) class for automatically categorizing requirement strings into distinct ontology classes. Functional, External and Internal Quality, Constraints, and Other NFR are the process's outcomes. PROMISE Corpus and Concordia RE Corpus were the datasets utilized in the procedure. Advantages include: Researchers interested in evaluating the effort made for the purpose of building requirements in general and improving the quality of programs in particular will be interested in the findings of this study. Disadvantages: The focus was specifically on NFR rather than FR.

Muhammad Younas and Karzan Wakil [82] based in their study the method of applying the Word2Vec model and common keywords to identify subtypes of NFR, Therefore it was considered an automated approach based on semantic similarity that does not require pre-classification of requirements to identify NFRs from requirements documents. The performance of the approach used in terms of precision, recall and F-measure was measured by applying the approach based on the PROMISE-NFR dataset. The findings suggest that a semi-supervised automated approach to NFR detection lowers manual human work. Advantages: Because these methods do not require pre-classified criteria for

training the Word2Vec model, human manual work in the NFR identification process is reduced. Disadvantages: The number of NFR kinds in an off-the-shelf PROMISE dataset is limited by the developer of the dataset that specified it. Furthermore, the data set employed may contain some misunderstandings about the NFR categorization, and the Word2Vec model is linked to Wikipedia's lexicon. The model will not be able to find the similarity value if the word in the requirements is not in Wikipedia.

Mengmeng Lu and Peng Liang[59] Users' reviews were automatically divided into four categories of NFR (usability, dependability, performance, and portability), as well as functional requirements (FRs) and others. This is accomplished by combining four classification technologies (TF-IDF, CHI2, BoW, and AUR-BoW) with three machine learning methods (J48, Naive Bayes, and Bagging). The study's data collection included iBooks and WhatsApp. The results show that combining AUR-BoW with Bagging produces the greatest outcomes (71.4 percent accuracy, 72.3 percent recall, and 71.8 percent F-measure) of all formulas. Advantages: Automatic NFR categorization from user reviews may assist application developers better understand user reviews and address user demands from an NFR standpoint, as well as help developers retain and attract new users. Disadvantages: Two categories of NFRs, compatibility and security, do not exist in the experiment data set, and the number of NFRs for portability and performance is relatively small.

Pir Sami Ullah Shah et al[83]. developed an automated classification of software needs into two broad categories, functional and non-functional, utilizing natural language processing and machine learning. they use NLP, TF-IDF, Support Vector Machine, Naïve Bayesian, Recurrent Neural Network (RNN). The Software Requirements Dataset, which was utilized in the search, achieved the maximum accuracy of 92 percent when utilizing the RNN technique. The data was taken from the Kaggle repository. Advantages : The spotlight focuses on the NFR as much as it highlights the FRs because software developers mostly focus on FRs to compare with NFRs which end in massive software failures Users also face problems while describing NFR and sometimes NFR is hidden in user stories. Disadvantages: NFRs are not more specifically categorized into safety, security, performance, and usability requirements .

With Word2vec and rapid Text model technology, S Tiun et al [5] used the RE'17 dataset challenge as a dataset. To see how word embedding compares to typical characteristics (such a bag of words) in the NFR and FR classification. In addition to understanding that the greatest performance for the classification of NFR and FR requires the employment of a complicated neural classifier. The findings revealed that FastText is a good classification model, as it received the highest F1 score of 92.8 percent. Advantages: FastText is successful in binary classification of text when the documents to be classified are very short and contain few vocabulary. Disadvantages: fastText fails to classify large documents with a large vocabulary in which case TFIDF should be considered with NB Naïve Bayes as a classification model.

Alex Dekhtyar and Vivian Fong [84]applied TensorFlow-guided learning and Word2Vec-based representations of classification problems in requirements documents where three classes of machine learning techniques were compared for the purpose of determining requirements for SecReq and NFR data sets. The first category used Na¨ıve Bayes which is the basic method on word count and TF-IDF for representation of requirements. TensorFlow's convolutional neural networks are trained on random, pre-trained Word2Vec merges of words in the requirements in the remaining two category approaches. The SecReq dataset was utilized to do the search. Advantages: Using Word2Vec to represent individual words in requirements improves classification accuracy by a significant amount.
Disadvantages: The classification process focused on two categories only, which are either security requirements or NFR, regardless of other sub-types of NFR (reliability, usability, etc.) and FR .

Vivian Fong [52] had applied deep learning techniques (Naıve Bayes classifier and CNN classifier) for the purpose of automatic classification of software requirements, the author use word embedding when training a convolutional neural network (CNN) to represent documents. The dataset used in the network training and testing process is Quality Attributes (NFR) dataset (PROMISE corpus) and SecReq dataset. Advantages: A comparison of three word embedding strategies to assist represent requirements documents while training CNNs, and lastly a set of evaluations for the purpose of requirements categorization using two well-studied datasets Advantages: When configuring CNNs, the emphasis is on filter sizes, filter count, and number of training epochs, leaving out a vast array of CNN hyper parameter. additionally, the research did not investigate the fast text category and compare its performance as well as training time metrics with CNN outputs.

2-6 Data Sets

When performing a software requirements classification process using deep learning techniques, a data set must be provided for the purpose of training and testing the model built in the classification process.In the following paragraphs, a number of data sets that were used in research in the classification of software requirements are clarified

PROMISE repository There are 625 identified natural language needs (255 Functional requirements and 370 non-functional requirements). First, the labels group the criteria into FR and NFR. Eleven subcategories are identified for the latter category: Performance (PE),Availability (A),Look & Feel (LF),Maintainability (MN), Operability (O), Usability (US), Fault Tolerance (FT), Scalability (SC), Security (SE), Legal & Licensing (L). and Portability (PO) . Table 1 shows the number of requirements for each category of software requirements in this repository

Table1.Number of requirements in the PROMISE repository

class	numbers	percent%
Functionality (F)	255	40.80
Availability(A)	21	3.36
Fault Tolerance(FT)	10	1.60
Legal(L)	13	2.08
Look and Feel(LF)	38	6.08
Maintainability(MN)	17	2.72
Operability(O)	62	9.92
Performance(PE)	54	8.64
Portability(PO)	1	0.16
Scalability(SC)	21	3.36
Security(SE)	66	10.56
Usability(US)	67	10.72
total	625	100%

PROMISE-Exp The PROMISE Orig (PROMISE) range has been expanded. The dataset generated using known machine learning methods was evaluated after adding new software requirements. Determination a model and extract the specifications of the software used in the previous repository from the manual study. The results of the ML algorithms used to validate this extension were compared with the results of the original rule when they were provided for similar methods. It was discovered that

the new PROMISE exp database could be used in research using ML algorithms that did not support the automated software requirements classification task, and that there was an increase of 55% over the original PROMISE database. The amount of requirements for each type of standard before and after the expansion process is shown in the Table 2.

Table 2. Number of requirements in the PROMISE-Exp

Class	PROMISE orig	PROMISE exp	Total expanded	Taxa Aumento (%)
FR	255	443	188	73.73
A	21	34	13	61.9
L	13	14	1	7.69
LF	38	42	4	10.53
MN	17	29	12	70.59
O	62	79	17	27.42
PE	54	72	18	33.33
SC	21	22	1	4.76
SE	66	128	62	93.94
US	67	81	14	20.90
FT	10	16	6	60.00
PO	1	10	9	90.00
Total	625	969	344	55.20

Corpus Repository It is one of the datasets used by [85]that is available for download via [86]. It contains a total of 765 sentences and 15 SRS problem statements from various disciplines. 270 of them (or 35 percent) have the "FR" annotation, while 495 (or 65 percent) have the "NFR" annotation. referred to as CorpusN and CorpusF, respectively.

SRS (NIRS: National Institute of Radiological Sciences, JUAS: Japan Users Association of Information Systems, IPA: Information Technology Promotion Agency, Requests for Proposal RFP) : The following thirteen online social action models are use by Japanese local governments or other public entities. Therefore, the majority of requests for proposals (RFPs) concern information systems, while there are also exceptions, such RFPs for medical systems. 11,538 required sentences in all, all written in Japanese, were gathered and are displayed in Table 3.

Table3.Number of requirements in the SRS dataset

NO	Issued from	systems	Size(lines)
1	Moriyama city	Sewerage accounting system	330
2	NIRS	Medical information system	7083
3	Okayama city	Attendance management system	168
4	Nara prefecture	House construction registration system	50
5	Hayama Town	Public service company management system	88
6	Kanda Town	Public health management system	744
7	Kudarimatsu City	School meal management system	126
8	Yokohama City	Library information system	1458
9	JUAS	Non-functional requirements indicators	288
10	Kyoto Prefecture	Total information system	377
11	Kyoto Prefecture	Library information system	552
12	IPA	Grade table of non-functional requirements	201
13	Ashikaga City	Sewerage accounting system	73
total			11538

ID	reference number	Data Set	Technique used	class	Accuracy %	Precision %	Recall %	F-Measur %
1	[74]	PROMISE repository	Latent Dirichlet Allocation, Biterm Topic Modeling and Naıve Bayes	NFRs	92.97	0.98	0.93	0.95
				FRs	96.47	0.90	0.97	0.93
				Average	94.72	0.94	0.95	0.94
2	[63]	PROMISE_exp	Using Machine Learning Algorithms(SVM, MNB, kNN, LR)with BoW, TF-IDF and CHI^2	FRs and NFRs	0.91	0.78	0.79	0.78
3	[75]	PROMISE dataset	accuracy as a weight ensemble, mean ensemble, and accuracy per class as a weight ensemble with a four different DL models(LSTM, BiLSTM, GRU and CNN)	FRs and NFRs and the multi-label classification of both FRs and NFRs	95.00	0.94	0.95	0.94
4	[76]	TERA-PROMISE	Natural Language Processing and SVD	FRs and NFRs	0.83333	0.8142	0.8437	0.8222
5	[6]	Corpus Repository	text classifier equipped with a part-of-speech (POS) tagger.	Subcategories of NFRs	0.9856	0.98	1.00	0.99
6	[87]	PROMISE dataset	Super Vector Machine	FRs		0.92	0.93	0.93
				NFRs		0.93	0.92	0.92
				Average		92.5	92.5	92.5
7	[78]	SRS (NIRS, JUAS, IPA,RFP)	QRMiner tool by Artificial neural Network	FRs		0.89	0.94	0.91
				NFRs		0.70	0.54	0.61
				Non-requirements		0.86	0.84	0.85
				Average		0.816	0.773	0.79
8	[79]	App ibooks	NLP Technique & TF-IDF	FRs		0.35	0.76	0.48
				NFRs		0.75	0.92	0.83
				Average		0.55	0.84	0.655
9	[80]	Apple store data and Google store data	Naive Bayes; Decision Tree; and MaxEnt	bug reports,		0.94	0.97	0.90
				feature requests,		0.96	0.98	0.89
				user experiences,		0.92	0.88	0.79
				text ratings		0.91	0.94	0.88
				Average		0.9325	0.9425	0.865
10	[2]	Doors database	Convolutional neural network	Requirements	0.81	0.73	0.89	0.80

				Information		0. 90	0.75	0.82
				Average		0.815	0.82	0.81
11	[81]	PROMISE Corpus	Super Vector Machine	FRs , External and Internal Quality , Constraints and other NFR	SVM + promise Corpus			
						0.77	0.60	0.67
					SVM + Concordia RE corpus.			
						0.84	0.84	0.85
12	[88]	PROMISE Repository	Word2Vec model and popular keywords for identification of NFR.	Subcategories of NFRs		0.6225	0.4449	0.4228
13	[59]	iBooks and WhatsApp	BoW, TF-IDF, CHI2, and AUR-BoW) with three machine learning algorithms Naive Bayes, J48, and Bagging	usability		0.757	0.565	0.647
				reliability		0.595	0.552	0.572
				portability,		0.632	0.327	0.431
				performance		0.596	0.233	0.335
				FRs		0.630	0.587	0.608
				Others		0.770	0.881	0.822
				Average		0.714	0.723	0.718
14	[83]	Kaggle repository (PROMISE Repository)	NLP, TF-IDF, SVM, Naïve Bayesian and RNN	FRs and NFRs	Naïve Bayes			
					0.90	0.90	0.40	0.55
					SVM			
					0.91	0.70	0.50	0.63
					RNN			
					0.92	0.60	0.30	0.40
15	[5]	RE '17 Data Challenge Area(PROMISE Repository)	Word2vec and fast Text model	FRs and NFRs		92.8	92.8	92.8
16	[84]	SecReq dataset	Word2Vec and TensorFlow, Naïve Bayes and TF-IDF	SecReq	0.9105	0.9152	0.9138	0.9134
				NFRs	0.9259	0.9268	0.9187	0.9216
				Average	0.9182	0.921	0.9163	0.9176
17	[52]	Quality Attributes (NFR)dataset(PROMISE corpus) and SecReq dataset	Naıve Bayes classifier and CNN classifier	Naıve Bayes NB				
				SecReq -SE	0.759	0.791	0.888	0.837
				NFRs-NF	0.922	0.915	0.926	0.921
				NFRs-SE	0.893	0.958	0.638	0.766
				Average	0.858	0.888	0.817	0.841
				Convolutional Neural Networks CNN				
				SecReq -SE	0.936	0.919	0.915	0.911
				NFRs-NF	0.932	0.929	0.962	0.945
				NFRs-SE	0.951	0.957	0.774	0.854
				Average	0.94	0.935	0.883	0.903

iBooks app Out of 1000 users, 217 user reviews in the English-language app store for the iBooks app contain FR information, while 622 user reviews contain NFR information (some user reviews may contain both FR and NFR information) (i.e. ground truth).

Apple store data and Google store data It gathered around 1.1 million reviews for 1,100 applications, half of which are paid and the other half free, using the Apple AppStore and Google Play Stores to gather experience data. Only 80 applications, of which half were bought and the other half were free, received 146,057 reviews on the Google Store, which was only allowed to gather reviews. A random sample of a portion of the manual tagging was taken from the obtained data. Pick 1,000 reviews at random from the Google Store data and 1,000 reviews from the Apple Store.

DOORS database The DOORS database is a database of DOORS (Dynamic Object-Oriented Requirements) containing 10,000 items extracted from 89 documents,These items fall into two categories: information and requirements.

PROMISE Corpus The PROMISE Corpus consists of 15 SRS documents, developed as semester projects by Master students at DePaul University. This specification contains a total of 326 NFR and 358 FR . The NFR in this group are divided into 9 categories availability (A), look-and-feel (LF), legal (L), operational (O), performance (P),, maintainability (M), security (SE), usability (US) and scalability (SC) . Table 4 shows the NFR Classes and a number of sentences for each Classes in PROMISE Corpus.

Table (4).PROMISE Corpus: NFR Classes and a number of sentences for each Classes

Class	A	LF	L	M	O	P	SC	SE	US	Total
#sentence	18	35	10	16	61	48	18	58	62	326

SecReq is a data set that is used in research to improve the task of recalling security requirements. The data set consists of requirements categorized into two categories, security-related or non-security-related. The Naıve Bayes class was trained on the data for the purpose of classification.

3. Results And Discussion

Summary of results

Summarization of relevant published papers that are considered is illustrated in Table (5).

Results Discussion

By observing the data sets used in studies related to the requirements classification process, we found that most studies used the PROMISE repository as a data set in the training and testing process. It contains a set of publicly available data sets and tools to serve researchers in building predictive software models (PSM) and the software engineering community in general. Thus we found Through a survey on previous studies that the natural language processing is one of the first and most important stages that take place before building models for classification requirements. Moreover, there are also multiple text conversion models and feature extraction from them, where several different techniques were used to convert text data into digital vectors. And the Term Frequency Inverse Document Frequency (TF-IDF) technology overcame all other techniques in performing the same function. Generally there are also multiple methods and techniques used in classifying software requirements through the use of different machine learning algorithms, as some of them are under supervision and

others are without supervision, and there are also semi-supervised algorithms. It turns out that There are two levels when classifying requirements. Indeed, in the first level, the objective is to classify the binary requirements into FRs and NFRs only, and in second level, the objective of which is a multiple classification of requirements, FRs and NFRs, in addition to the sub-types of NFRs.

By observing the results in the previous table, it was found that the highest value of accuracy is 0.9856, through a research presented by Ishrar Hussain et al [6]through their use of a text classifier equipped with a part-of-speech (POS) tagger in order to obtain On non-functional requirement subtypes that have a Precision score of 0.98 and a Recall value of 1 F-Measur of 0.99 .

As for the lowest value for accuracy, it was found in the study presented by Winkler and Andreas Vogelsang [2], which amounted to 0.81 through the application of convolutional neural network techniques in order to classify requirements documents into two basic categories: requirements and information, where the Precision rate reached 0.815 with a value of Recall equals 0.82, F-Measur equals 0.81.

It was also found that there is a large discrepancy in the accuracy of classification in a study presented by Vivian Fong [52] for the purpose of classifying the requirements documents into a triple classification (are they safety requirements or not, are they non-functional requirements from other categories or not, are they non-functional requirements or represent Security requirements) when they applied deep learning techniques in the classification process, where it was found that in the case of using the Naıve Bayes classifier, the classification accuracy rate was about 0.858, where the Precision rate reached 0.888 with a Recall value equal to 0.817, F-Measur equal to 0 .841.

When applying the techniques of convolutional neural networks for the same classification, it was found that the accuracy rate was higher than the accuracy rate reached in the case of using Naıve Bayes, where the accuracy rate reached 0.94 and the Precision rate reached 0.935 with a Recall value equal to 0.883, F -Measur equals 0.903 **.** This proves that the use of convolutional neural networks provides better results in the classification process compared to the use of the Naıve Bayes classifier.

As explained by forms (2,3,4,5) .

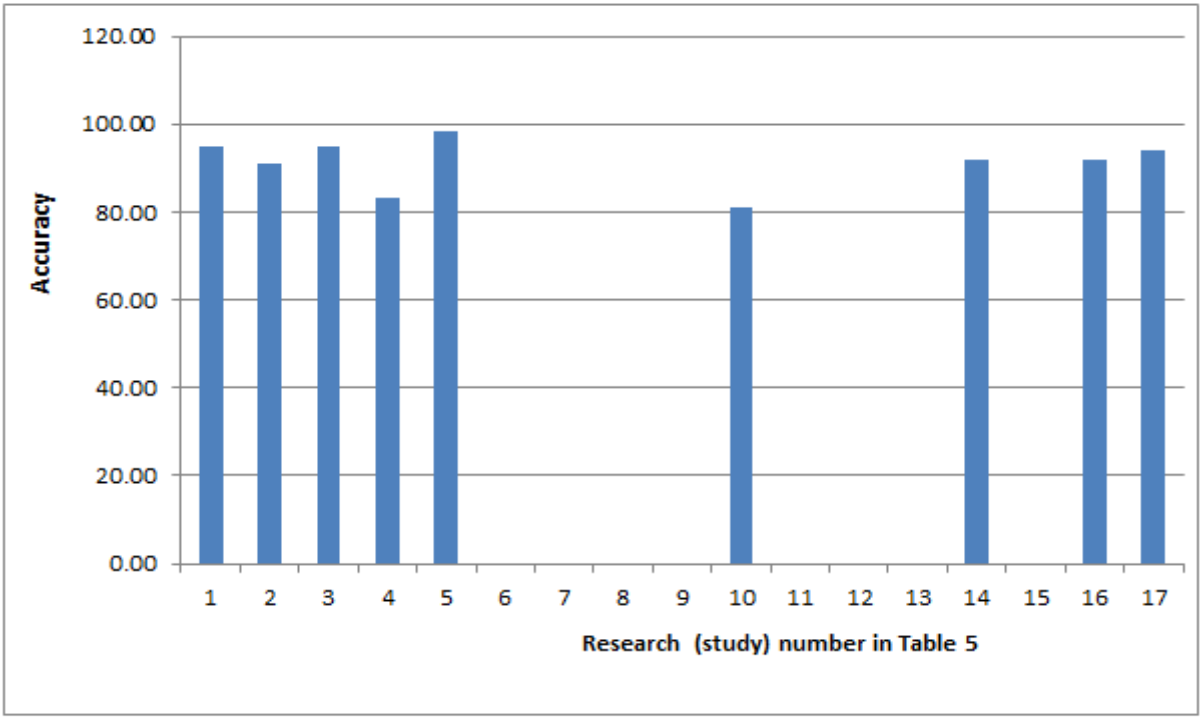

Figure 2. accuracy results gradation

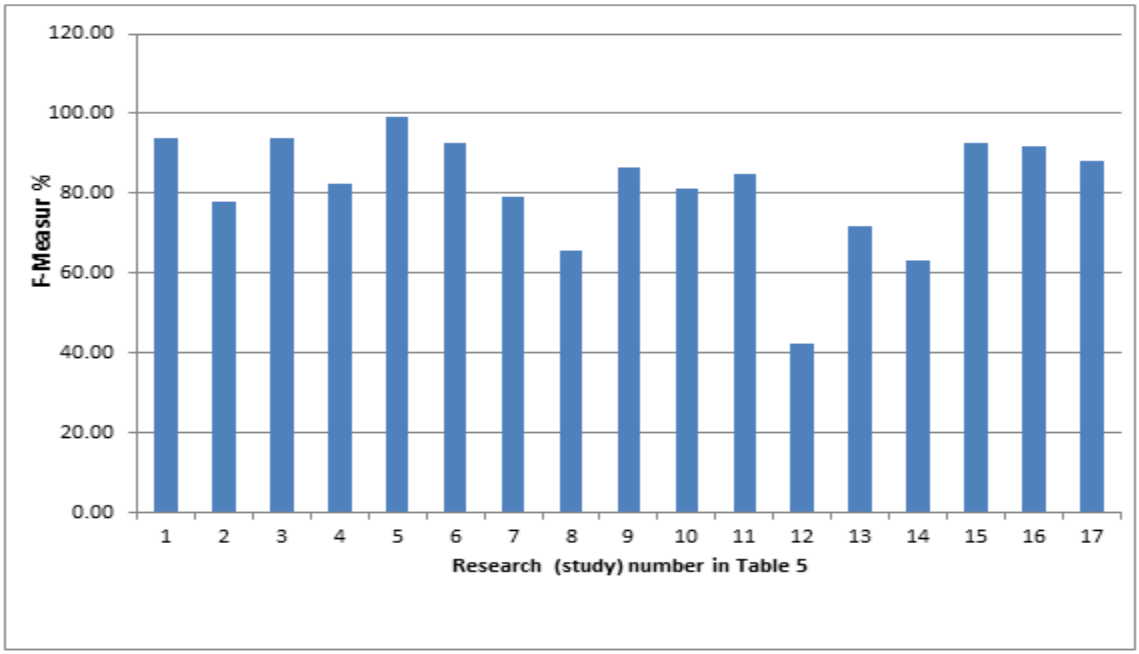

Figure 3. F-Measure results gradation

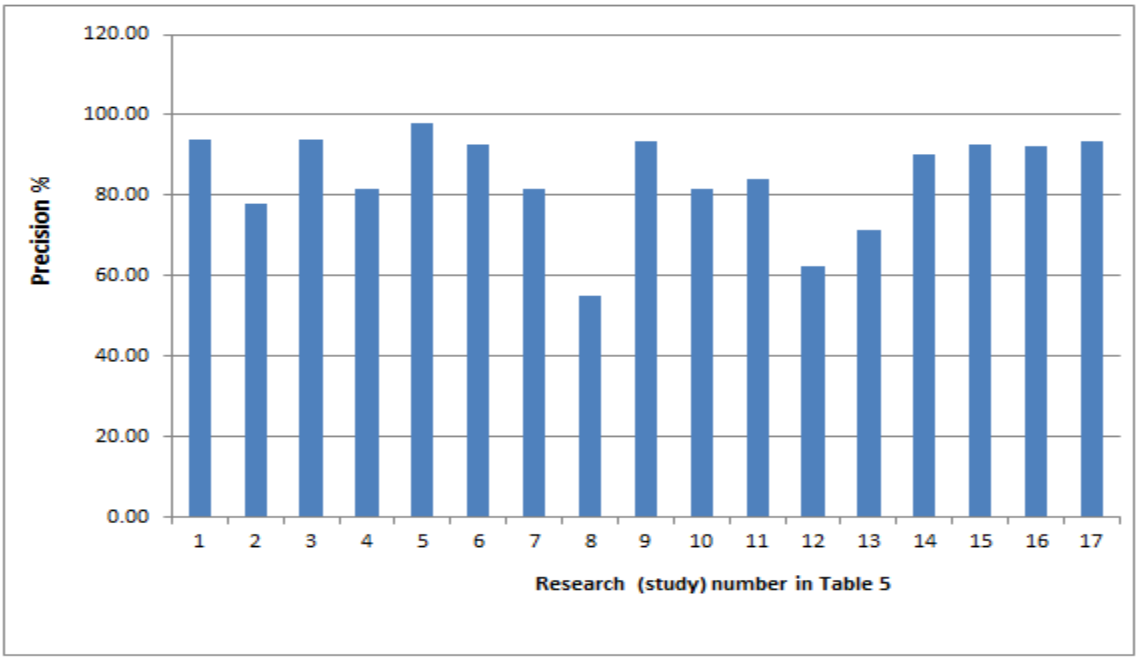

Figure4 . Precision results gradation

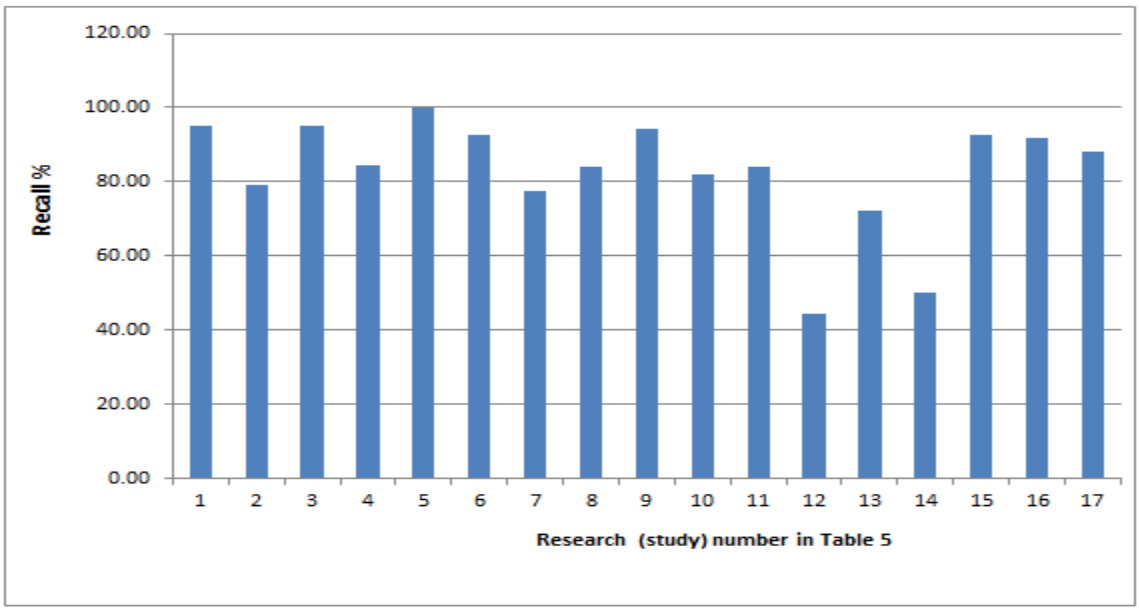

Figure 5. Recall results gradation

4. Conclusion

By noting the Metrics results of previous researches in Table 1 , it was found that in the process of binary classification of requirements (first level of classification) that the highest values of Accuracy were achieved when applying the techniques of Latent Dirichlet Allocation, Biterm Topic Modeling

and Naıve Bayes, This is when classifying a requirements document into FRs and NFRs where Accuracy reached 94.72% Precision was 94% Recall value 95% and F-Measure equal to 94% .

Accuracy was the lowest in the binary classification process when applying convolutional neural networks for classifying a requirements document, is it a requirement or information? with an Accuracy value of 83%, and the lowest value for Precision was 55% when applying Natural Language Processing techniques and TF-IDF for feature extraction. It was found that the lowest value for Recall and F-Measure when using processing techniques Natural languages and TF-IDF isotropic extraction with RNN with a Recall value of 30% and 40 % for F-Measure .

During classifying deep (requirements into more than one class), all metrics values are reached their maximum when using a text classifier equipped with a part-of-speech (POS) tagger to classify the NFRs with a value of 98.56 % for Accuracy , 98% for Precision , Recall value reached 100 % and has an F-Measure value of 99% . The minimum value of the Metrics in the multi-classification of requirements, which is when using the Naıve Bayes classifier for classification a requirements document as to whether it is a security requirement or not, and whether it is FRs or NFRs, the Accuracy reached 88.5. Also, when Word2Vec model and popular keywords for identification of NFR were used to obtain the subtypes of non-functional requirements, the minimum Metrics was reached with values of 62.25 % for Precision, Recall It has reached 44.49% and has an F-Measure value of 42.28%.

5. Acknowledgements

This paper and the research behind it would not have been possible without the exceptional support of my supervisor Ass .Prof .Dr. Nada N. Saleem. His enthusiasm, knowledge and keen attention to detail have been inspiring and kept my work on the right track from the first real beginning of this research all the way to the list of references.

6. References

1. Lauesen, S., *Task descriptions as functional requirements.* IEEE software, 2003. **20**(2): p. 58-65.
2. Winkler, J. and A. Vogelsang. *Automatic classification of requirements based on convolutional neural networks*. in *2016 IEEE 24th International Requirements Engineering Conference Workshops (REW)*. 2016. IEEE.
3. Pandey, D., U. Suman, and A.K. Ramani. *An effective requirement engineering process model for software development and requirements management*. in *2010 International Conference on Advances in Recent Technologies in Communication and Computing*. 2010. IEEE.
4. Navarro-Almanza, R., R. Juarez-Ramirez, and G. Licea. *Towards supporting software engineering using deep learning: A case of software requirements classification*. in *2017 5th International Conference in Software Engineering Research and Innovation (CONISOFT)*. 2017. IEEE.
5. Tiun, S., et al. *Classification of functional and non-functional requirement in software requirement using Word2vec and fast Text*. in *journal of Physics: conference series*. 2020. IOP Publishing.
6. Hussain, I., L. Kosseim, and O. Ormandjieva. *Using linguistic knowledge to classify non-functional requirements in SRS documents*. in *International Conference on Application of Natural Language to Information Systems*. 2008. Springer.
7. Glinz, M., *A glossary of requirements engineering terminology.* Standard Glossary of the Certified Professional for Requirements Engineering (CPRE) Studies and Exam, Version, 2011. **1**: p. 56.
8. Chung, L. and J.C.S.d. Prado Leite, *On non-functional requirements in software engineering*, in *Conceptual modeling: Foundations and applications*. 2009, Springer. p. 363-379.

9. Ernst, N.A. and J. Mylopoulos. *On the perception of software quality requirements during the project lifecycle*. in *International Working Conference on Requirements Engineering: Foundation for Software Quality*. 2010. Springer.
10. Abad, Z.S.H. and G. Ruhe. *Using real options to manage technical debt in requirements engineering*. in *2015 IEEE 23rd International Requirements Engineering Conference (RE)*. 2015. IEEE.
11. Abad, Z.S.H., et al. *What are practitioners asking about requirements engineering? an exploratory analysis of social q&a sites*. in *2016 IEEE 24th International Requirements Engineering Conference Workshops (REW)*. 2016. IEEE.
12. Sarker, I.H., M.H. Furhad, and R. Nowrozy, *Ai-driven cybersecurity: an overview, security intelligence modeling and research directions.* SN Computer Science, 2021. **2**(3): p. 1-18.
13. Sarker, I.H., et al., *Mobile data science and intelligent apps: concepts, ai-based modeling and research directions.* Mobile Networks and Applications, 2021. **26**(1): p. 285-303.
14. Sarker, I.H., et al., *Cybersecurity data science: an overview from machine learning perspective.* Journal of Big data, 2020. **7**(1): p. 1-29.
15. Reddy, R.V.K., B.S. Rao, and K.P. Raju. *Handwritten Hindi digits recognition using convolutional neural network with RMSprop optimization*. in *2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS)*. 2018. IEEE.
16. Mohammed, M., M.B. Khan, and E.B.M. Bashier, *Machine learning: algorithms and applications.* 2016: Crc Press.
17. Sarker, I.H., *Machine learning: Algorithms, real-world applications and research directions.* SN Computer Science, 2021. **2**(3): p. 1-21.
18. Yuvalı, M., B. Yaman, and Ö. Tosun, *Classification Comparison of Machine Learning Algorithms Using Two Independent CAD Datasets.* Mathematics, 2022. **10**(3): p. 1-15.
19. Shafiq, S., et al., *Machine learning for software engineering: A systematic mapping.* arXiv preprint arXiv:2005.13299, 2020.
20. Meinke, K. and A. Bennaceur. *Machine learning for software engineering: models, methods, and applications*. in *2018 IEEE/ACM 40th International Conference on Software Engineering: Companion (ICSE-Companion)*. 2018. IEEE.
21. Cerqueira, M., P. Silva, and S. Fernandes, *Systematic Literature Review on the Machine Learning Approach in Software Engineering.* American Academic Scientific Research Journal for Engineering, Technology, and Sciences, 2022. **85**(1): p. 370-396.
22. Barenkamp, M., J. Rebstadt, and O. Thomas, *Applications of AI in classical software engineering.* AI Perspectives, 2020. **2**(1): p. 1-15.
23. Laplante PA. Dictionary of computer science, E.a.T.F.C.P.
24. Parra, E., et al., *A methodology for the classification of quality of requirements using machine learning techniques.* Information and Software Technology, 2015. **67**: p. 180-195.
25. Zhao, L., et al., *Natural language processing for requirements engineering: A systematic mapping study.* ACM Computing Surveys (CSUR), 2021. **54**(3): p. 1-41.
26. Anish, P.R., et al. *Identifying architecturally significant functional requirements*. in *2015 IEEE/ACM 5th International Workshop on the Twin Peaks of Requirements and Architecture*. 2015. IEEE.
27. Nuseibeh, B., *Weaving together requirements and architectures.* Computer, 2001. **34**(3): p. 115-119.
28. Glinz, M. *On non-functional requirements*. in *15th IEEE international requirements engineering conference (RE 2007)*. 2007. IEEE.

29. Praveena, M. and V. Jaiganesh, *A literature review on supervised machine learning algorithms and boosting process.* International Journal of Computer Applications, 2017. **169**(8): p. 32-35.
30. Caron, M., et al., *Unsupervised learning of visual features by contrasting cluster assignments.* arXiv preprint arXiv:2006.09882, 2020.
31. Kusner, M., et al. *From word embeddings to document distances*. in *International conference on machine learning*. 2015. PMLR.
32. Yan, X., et al. *A biterm topic model for short texts*. in *Proceedings of the 22nd international conference on World Wide Web*. 2013.
33. Bird, S., E. Klein, and E. Loper, *Natural language processing with Python: analyzing text with the natural language toolkit*. 2009: " O'Reilly Media, Inc.".
34. Lewis, D.D. *Naive (Bayes) at forty: The independence assumption in information retrieval*. in *European conference on machine learning*. 1998. Springer.
35. VanderPlas, J., *Python data science handbook: Essential tools for working with data*. 2016: " O'Reilly Media, Inc.".
36. Ho, W.K., B.-S. Tang, and S.W. Wong, *Predicting property prices with machine learning algorithms.* Journal of Property Research, 2021. **38**(1): p. 48-70.
37. Vergara, D., S. Hernández, and F. Jorquera. *Multinomial Naive Bayes for real-time gender recognition*. in *2016 XXI Symposium on Signal Processing, Images and Artificial Vision (STSIVA)*. 2016. IEEE.
38. Learning, S.M., *A review of classification techniques, sb kotsiantis.* Informatica, 2007. **31**: p. 249-268.
39. Lubis, Z., P. Sihombing, and H. Mawengkang. *Optimization of K Value at the K-NN algorithm in clustering using the expectation maximization algorithm*. in *IOP Conference Series: Materials Science and Engineering*. 2020. IOP Publishing.
40. Géron, A., Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems. 2019: O'Reilly Media.
41. Almestekawy, A. and M. Abdulsalam, *Sentiment analysis of product reviews using bag of words and bag of concepts.* International Journal of Electronics and Information Engineering, 2019. **11**(2): p. 49-60.
42. Khayyat, M.M. and L.A. Elrefaei, *Manuscripts Image Retrieval Using Deep Learning Incorporating a Variety of Fusion Levels.* IEEE Access, 2020. **8**: p. 136460-136486.
43. Luo, L., et al., *Document triage for identifying protein–protein interactions affected by mutations: a neural network ensemble approach.* Database, 2018. **2018**.
44. Pan, M., et al., *Water level prediction model based on GRU and CNN.* Ieee Access, 2020. **8**: p. 60090-60100.
45. Li, Z., et al., *A fuzzy recurrent neural network for driver fatigue detection based on steering-wheel angle sensor data.* International Journal of Distributed Sensor Networks, 2019. **15**(9): p. 1550147719872452.
46. Siami-Namini, S., N. Tavakoli, and A.S. Namin. *A comparison of ARIMA and LSTM in forecasting time series*. in *2018 17th IEEE international conference on machine learning and applications (ICMLA)*. 2018. IEEE.
47. Zhang, S., et al., *Deep learning algorithms for bearing fault diagnostics—A comprehensive review.* IEEE Access, 2020. **8**: p. 29857-29881.
48. Shahid, F., A. Zameer, and M. Muneeb, *Predictions for COVID-19 with deep learning models of LSTM, GRU and Bi-LSTM.* Chaos, Solitons & Fractals, 2020. **140**: p. 110212.

49. Demir, N., *Ensemble methods: Elegant techniques to produce improved machine learning results.* Toptal Engineering Blog, 2016.
50. Mandal, S., et al., *Analysis of composition of microbiomes: a novel method for studying microbial composition.* Microbial ecology in health and disease, 2015. **26**(1): p. 27663.
51. Torgo, L., *Data Mining with R: Learning with Case Studies. Chapman & Hall/CRC Data Mining and Knowledge Discovery Series*. 2010, Taylor & Francis Boca raton.
52. Fong, V.L., *Software Requirements Classification Using Word Embeddings and Convolutional Neural Networks.* 2018.
53. Torgo, L., *Data mining with R: learning with case studies*. 2011: chapman and hall/CRC.
54. Saravanan, N. and V. Gayathri, *Performance and classification evaluation of J48 algorithm and Kendall's based J48 algorithm (KNJ48).* International Journal of Computer Trends and Technology, 2018. **59**(2): p. 73-80.
55. Sarkar, D., *Text Analytics with python*. 2016: Springer.
56. Shcherban, S., et al. *Automatic identification of code smell discussions on stack overflow: A preliminary investigation*. in *Proceedings of the 14th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. 2020.
57. Hakim, A.A., et al. *Automated document classification for news article in Bahasa Indonesia based on term frequency inverse document frequency (TF-IDF) approach*. in *2014 6th international conference on information technology and electrical engineering (ICITEE)*. 2014. IEEE.
58. Forman, G., *An extensive empirical study of feature selection metrics for text classification.* J. Mach. Learn. Res., 2003. **3**(Mar): p. 1289-1305.
59. Lu, M. and P. Liang. *Automatic classification of non-functional requirements from augmented app user reviews*. in *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering*. 2017.
60. Tamminen, M., *Then shall I know fully: Relative frequencies of part-of-speech n-grams in native and translated Finnish literary prose.* 2018.
61. El Mostafa, H. and F. Benabbou, *A deep learning based technique for plagiarism detection: a comparative study.* IAES International Journal of Artificial Intelligence, 2020. **9**(1): p. 81.
62. Breiman, L., *Bagging predictors.* Machine learning, 1996. **24**(2): p. 123-140.
63. Dias Canedo, E. and B. Cordeiro Mendes, *Software requirements classification using machine learning algorithms.* Entropy, 2020. **22**(9): p. 1057.
64. Sokolova, M. and G. Lapalme, *A systematic analysis of performance measures for classification tasks.* Information processing & management, 2009. **45**(4): p. 427-437.
65. Tabatabaei, S.A., J. Klein, and M. Hoogendoorn. *Estimating the $$ F_1 $$ Score for Learning from Positive and Unlabeled Examples*. in *International Conference on Machine Learning, Optimization, and Data Science*. 2020. Springer.
66. Schnack, H.G., et al., *Can structural MRI aid in clinical classification? A machine learning study in two independent samples of patients with schizophrenia, bipolar disorder and healthy subjects.* Neuroimage, 2014. **84**: p. 299-306.
67. Boutaba, R., et al., *A comprehensive survey on machine learning for networking: evolution, applications and research opportunities.* Journal of Internet Services and Applications, 2018. **9**(1): p. 1-99.
68. Allouch, A., et al., *Roadsense: Smartphone application to estimate road conditions using accelerometer and gyroscope.* IEEE Sensors Journal, 2017. **17**(13): p. 4231-4238.
69. Chicco, D. and G. Jurman, *The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation.* BMC genomics, 2020. **21**(1): p. 1-13.

70. Ghosh, S., S. Mondal, and B. Ghosh. *A comparative study of breast cancer detection based on SVM and MLP BPN classifier*. in *2014 First International Conference on Automation, Control, Energy and Systems (ACES)*. 2014. IEEE.
71. Bharati, S., M.A. Rahman, and P. Podder. *Breast cancer prediction applying different classification algorithm with comparative analysis using WEKA*. in *2018 4th International Conference on Electrical Engineering and Information & Communication Technology (iCEEiCT)*. 2018. IEEE.
72. Lipton, Z.C., C. Elkan, and B. Narayanaswamy, *Thresholding classifiers to maximize F1 score.* arXiv preprint arXiv:1402.1892, 2014.
73. Flach, P. and M. Kull, *Precision-recall-gain curves: PR analysis done right.* Advances in neural information processing systems, 2015. **28**.
74. Abad, Z.S.H., et al. *What works better? a study of classifying requirements*. in *2017 IEEE 25th International Requirements Engineering Conference (RE)*. 2017. IEEE.
75. Rahimi, N., F. Eassa, and L. Elrefaei, *One-and Two-Phase Software Requirement Classification Using Ensemble Deep Learning.* Entropy, 2021. **23**(10): p. 1264.
76. Mahmoud, M., *Software requirements classification using natural language processing and SVD.* Int. J. Comput. Appl, 2017. **164**(1): p. 7-12.
77. Kurtanovi´c, Z.;, and W. Maalej, *Automatically classifying functional and non-functional requirements using supervised machine learning.* In Proceedings of the 2017 IEEE 25th International Requirements Engineering Conference (RE), Lisbon, Portugal, 4–8 September 2017; pp. 490–495.
78. Tamai, T. and T. Anzai. *Quality Requirements Analysis with Machine Learning*. in *ENASE*. 2018.
79. Yang, H. and P. Liang. *Identification and Classification of Requirements from App User Reviews*. in *SEKE*. 2015.
80. Maalej, W., et al., *On the automatic classification of app reviews.* Requirements Engineering, 2016. **21**(3): p. 311-331.
81. Rashwan, A., O. Ormandjieva, and R. Witte. *Ontology-based classification of non-functional requirements in software specifications: a new corpus and svm-based classifier*. in *2013 IEEE 37th Annual Computer Software and Applications Conference*. 2013. IEEE.
82. Younas, M., et al., *An automated approach for identification of non-functional requirements using Word2Vec model.* Int. J. Adv. Comput. Sci. Appl, 2019. **10**(8): p. 539-547.
83. Shah, P.S.U., I. Ullah, and M. Shoaib, *Research Report (Part-II).*
84. Dekhtyar, A. and V. Fong. *Re data challenge: Requirements identification with word2vec and tensorflow*. in *2017 IEEE 25th International Requirements Engineering Conference (RE)*. 2017. IEEE.
85. Cleland-Huang, J., et al. *The detection and classification of non-functional requirements with application to early aspects*. in *14th IEEE International Requirements Engineering Conference (RE'06)*. 2006. IEEE.
86. Boetticher, G., *The PROMISE repository of empirical software engineering data.* http://promisedata. org/repository, 2007.
87. Kurtanović, Z. and W. Maalej. *Automatically classifying functional and non-functional requirements using supervised machine learning*. in *2017 IEEE 25th International Requirements Engineering Conference (RE)*. 2017. Ieee.
88. Younas, M., et al., *An automated approach for identification of non-functional requirements using word2vec model.* International Journal of Advanced Computer Science and Applications, 2019. **10**(8).