Islamic University of Technology (IUT)

Identification and Classification of Non-Functional Requirements from User Reviews

Authored by

Md. Ashif Aziz, 170042031

Nafisa Mehjabin, 170042033

Khandaker Rifah Tasnia, 170042042

Supervised by

Lutfun Nahar Lota

Assistant Professor, Department of CSE

Co-supervised by

Mohammad Anas Jawad

Lecturer, Department of CSE

Md. Mezbaur Rahman

Lecturer, Department of CSE

A thesis submitted to the Department of Computer Science and Engineering in partial fulfillment of the requirements for the degree of B.Sc Engineering in Software Engineering.

Software Engineering Program, Department of Computer Science and Engineering

Islamic University of Technology (IUT)

A Subsidiary organ of the Organization of Islamic Cooperation (OIC)

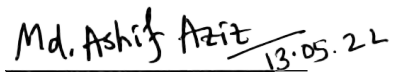
Academic Year: 2020-2021

11 of May, 2022

Declaration of Authorship

This is to certify that the work presented in this thesis is the outcome of the analysis and experiments carried out by Md. Ashif Aziz, Nafisa Mehjabin and Khandaker Rifah Tasnia under the supervision of Lutfun Nahar Lota, Assistant Professor of Department of Computer Science and Engineering (CSE), Islamic University of Technology (IUT), Gazipur, Dhaka, Bangladesh. It is also declared that neither this thesis nor any part of it has been submitted anywhere else for any degree or diploma. Information derived from the published and unpublished work of others have been acknowledged in the text and a list of references is given.

Authors:

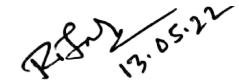


Md. Ashif Aziz

Student ID: 170042031

Nafisa Mehjabin

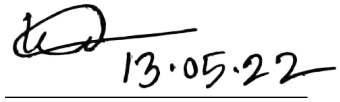
Student ID: 170042033



Khandaker Rifah Tasnia

Student ID: 170042042

Supervisor:



Lutfun Nahar Lota

Assistant Professor

Department of Computer Science and Technology

Islamic University of Technology

Acknowledgement

We would like express our gratitude towards IUT authority for granting us the fund and providing assistance required to implement our proposed system. We are indebted to our supervisor, Lutfun Nahar Lota madam for providing us with insightful knowledge and guiding us at every stage of our journey. Also we would like to thank Mohammad Anas Jawad sir and Md. Mezbaur Rahman sir for their special contributions. Finally, we would like to express our heartiest appreciation towards our family members for their continuous support, motivation, suggestions and help, without which we could not have achieved the scale of implementation that we have achieved.

Abstract

Nowadays software applications are used for diverse purposes. With the explosion of the web and mobile experiences, system design fully depends on who is using the application. For the success of the application, developers need to be aware of the users' concerns and expectations of the application. Existing research investigated that user reviews or feedback contain the quality concerns of the software that should be considered as non-functional requirements of the software to deliver a high-quality product. User reviews are usually short, unstructured, and written in an informal language, thus making it challenging to classify them based on the NFR standards and the huge quantity makes it tedious to identify requirements from review in the first place. To resolve this, we used Transformer-based language models to automate the detection of requirements from user reviews and classify Non-Functional requirements into seven sub-classes. We compared the classification results of the BERT and RoBERTa models using various evaluation metrics, and found that the fine-tuned version of RoBERTa surpassed BERT in classifying user reviews into requirement and non-requirement classes, as well as in classifying Non-functional requirements into seven subclasses.

Keywords*— *Requirements Classification, Non-functional requirements, User reviews, Transfer learning, BERT, RoBERTa

Contents

List of Figures

List of Tables

1 Introduction

Software Requirement is the key factor to maintain quality of software develoment process. These requirements represent the user expectations of the software. Usually, these requirements are listed by the software owners or stakeholders. But in order for an application to be successful, requirements should included from the concerns of end-users who uses the software.

Nowadays software applications are used for diverse purposes. All the applications are built highlighting the relationship between system and user. With the growth of the internet and mobile experiences, system design fully depends on who is using the application, where they are using it, and when they are using it. That means applications today need to be aware of the users, their location, focus, intent, and even their emotional state. So, the user is at the center now, not the system. That is why users know better what they need in the application. Basically, the users' concerns and expectations reflected in reviews or feedback comments for software applications contain valuable information regarding quality of a software [31][17] So, from the beginning of building an application the user requirements should be kept in mind, that will help the developers to deliver a quality product.[17] That is why researchers are now focusing on user reviews to automatically classify and identify requirements using different classification techniques. [28]

In this section, we have presented the the overview and explained the concepts of our research domain. In the later sections we described our motivation towards conducting this research, problem statement and the significance of solving this problem. We also mentioned the challenges we faced while conducting our thesis work. And Lastly presented the contribution that we have achieved upon completion of our research.

1.1 Overview

1.1.1 Requirements Specifications in Software Development Life Cycle

Eliciting requirements is a method for determining the demands of potential clients and users. This procedure could take some time to finish. Software Requirement Specifications (SRS) are a method of listing user or system needs that must be met in order for a product to be complete. Some requirements can describe the product's specific behaviors, features, and use cases. Some needs, such performance, scalability, and security, evaluate application properties and restrictions. In a nutshell, requirements specifications specify what a system should accomplish and how it should behave.[6]

1.1.2 Requirements Classification

The process of categorizing requirements into distinct types is known as requirements classification. Requirements are generally divided into two types- Functional requirements (FR) and Non-functional requirements (NFRs).[30] Functional requirements drive its application architecture which is what software must perform. On the other hand, Non-functional requirements drive a system's technical architecture which defines how software should perform. So, it is clear that functional requirements cover all of the project's features and functions as well as how users interact with it. While NFRs explain overall quality of the system.[6]

1.1.3 The Characteristics of Functional Requirements (FR) and Non-functional Requirements (NFRs)

Functional Requirement: A software requirement that describes a functionality that a software system must do. These criteria specify how the system's software and hardware elements behave in terms of producing or gaining outputs. Functional requirements define the desired end function of a system functioning under typical parameters, to verify that the design is adequate to produce the required output and that the final product achieves the design's capability to meet user expectations. A functional need is typically a basic feature or desired behavior that is clearly and quantitatively stated. A functional requirement for a jar can be that it retains liquid and also has a threaded top for just a cover to seal the jam with better preservation. Whenever a party fails to fulfill functional standards, it usually means it's of poor quality and may be completely unusable.[6]

Non-functional Requirements: A software requirement that explains how the software will function rather than what it will do. It basically refers to the quality attributes that a software system should have such as performance requirements, design limitations, software quality aspects, software external interface requirements, and so on. These requirements limit some degree of design freedom for building the software. Executing and evaluating qualities of software are mainly listed in Non-functional requirements.[6]

Some examples of the categories are given below:

Functional requirements

- "When an order is placed, the program must send a notification message.."
- "Visitors to the blog must be able to sign up for the application by putting their email address in the system."
- "Users must be able to utilize their phone numbers to authenticate their accounts."

- “The product shall allow a user to remove himself or herself from the list of players at any time."

Non-functional requirements

- “The system shall synchronize contacts and appointments in an acceptable time."
- “The system shall produce search results within 10 seconds"
- “Emails should be sent with a latency of no greater than 12 hours."
- “The look and feel of the system shall conform to the user interface standards of the smart device"

1.1.4 The Importance of Non-Functional Requirements(NFRs)

Non-functional requirements (NFRs) deacribes the important quality issues for software systems. These requirements can make precious role for a successful software systems. The software can be inconsistent and lack quality due to not addressing the Non-functional requirements. Also, it is time and cost consuming to fix any software. For this reason, users, clients, and developers all become unsatisfied. Some difficulties in addressing of NFRs are described below:-

1. NFRs can be descriptive. These requirements are described briefly so there can not be different perspectives for a particular requirement.[6]
2. Non-functional specifications might be subjective. Non-functional needs are interpreted and valued differently depending on the system under consideration.[6]
3. Non-functional needs can interact with one another. Because these criteria have a worldwide impact on systems, one requirement may have an impact on previously met requirements.[6]

For these reasons, non-functional needs might be challenging to manage. However, dealing with NFRs might be critical to a software system’s success. As a result, it is critical to deal with them effectively. Some of the importance of Non-functional requirements are as follows:

1. **Project completeness and quality assurance:** Non-functional requirements complete a software project. Without fulfilling these requirements the project will be inconsistent and a lot of modification will be needed further for a better user experience. Furthermore, non-functional requirements help in executing and evaluating the qualities of software. So, fulfilling these requirements assures the quality of software.
2. **Development team’s success:** The success of the development team is dependent on the success of software projects. This is contingent on user pleasure. When all non-functional requirements are considered, user happiness is achieved. As a result, the development team’s success is contingent on meeting non-functional requirements.

3. **User's satisfaction:** Non-functional requirements of a software system directly related to user satisfaction. These requirements are basically needed to improve the user experience and fulfill demands. Non-functional requirements like performance, scalability, maintainability, etc preserve user satisfaction.

1.1.5 User Reviews: Potential source of Requirements

User reviews are reviews or comments or feedback that is provided by the user of a software system or product. A person who has experienced the system gives feedback about that. A user review is a review written by anyone who has access to the internet and who shares their experience on a review site or social media platform after product testing or service evaluation.[41] It can be written by the end users who directly interact with the product/ software system or it can be written by the experts that are compensated for using the product and providing comments. The objective of user reviews is to assist stakeholders such as consumers, producers, and rivals in making decisions about the good or service that the user submitting the review has experienced.[15] User reviews of an application can be different types.[36] But we have focused here the text based review that can contain feature requests, bug experience or any quality issues. Online user reviews are becoming more popular as a source of criteria, but the focus has been on functional needs so far. They could also help with non-functional requirements elicitation to learn about the quality concerns that consumers have. Users have provided particularly meaningful information on the sub qualities they face during runtime and have shared their hands on experience in the feedback. As a result, online feedbacks or commments should be taken more seriously as a source of quality criteria.[17]

1.2 Motivation

Identifying all the non functional requirements of a specific type will allow engineers to be aware of the quality requirements that are often overlooked in its initial phases of the software process. Traditionally, task of identifying and distinguishing requirements is handled by a requirement analyst. The arduous task of manually marking which category a demand belongs to takes a lot of time and money. It might not always be accurate. Also, Manually assessing and identifying needs necessitates domain expertise, which is often scarce and costly. (i.e., Security expertise is required to properly identify security flaws in a vital software system.). So this resource constraint can discourage the analyst from properly assessing non-functional requirements and leaving the unidentified vulnerabilities that can be exploited later causing software defects to occur. Automating this task would be a beneficial practice in the software community.

Moreover, The large number of user reviews can be a valuable source of non-functional requirements, such as a software application's quality attributes. Building the application with the user's requirements from the beginning can reach the goal of meeting the user's expectations and ultimately lead to building high-quality software.

This is an emerging domain of software engineering that has been the focus of many researchers. But still, there is very little existing work on identification and automating non-functional requirements from user reviews. Also, there is a lack of a sufficiently large and labeled dataset of user reviews containing only requirements, leaving ample opportunity for working scope.

1.3 Problem Statement

Users have a direct connection to the end result of a product and desire the best quality software product possible. That's why users' concerns and expectations are mostly reflected in user reviews. Research shows, user reviews are potential source of quality attributes that are needed for the success of any software system. Basically these quality attributes should be considered as non-functional requirements. Before building a software, developers should consider these requirements to ensure that it is accepted by a large number of users. However, due to the large volume and unstructured format, manually detecting requirements and classifying them into non-functional requirement types is laborious.

Our goal is to Automate Non-functional Requirements (NFRs) classification to alleviate the need for manual classification and labeling of requirements while keeping users' concerns in mind to assist developers in delivering a solution that matches the needs of users.

1.4 Research Challenges

There were some challenges that we have faced during this research. Those are given below:

1. **Insufficient Dataset:** To perform more accurate results we need more dataset of user reviews. There are many datasets with unstructured user reviews. But most of them do not indicate any type of user requirements. For example, reviews or comments like "great app" or "awesome" do not refer to any non-functional requirements or functional requirements. Other than this there are datasets that do not have any kind of information about the software systems. There are only a few datasets that we have found that consist of different requirements. So, the lack of datasets containing user reviews addressing requirements has become a challenge for our research.

2. **Unstructured Data :** Our main target is to identify the requirements from user reviews. But users do not use any formal or structured words for writing any reviews. They use short forms of words, short sentences, emojis, complain about irrelevant things through their experience, or even give multiple requirements in the same review. No format is applicable in user reviews. So, these unstructured reviews become a challenge as the majority of them do not mention any non-functional or functional needs. and some of them refer to multiple types of requirements.

3. **Imbalanced dataset:** Non-functional requirements can take numerous forms. We retrieved needs from user evaluations and classified them as non-functional and functional requirements in the context of a document. But there was imbalance of different form non-functional requirements. Some of the classifications of non-functional requirements have a low percentage of data in the datasets. Some of them were merged with another classification as they broadly mean the same. For example, 'Fault tolerance' which is also referred to as 'Reliability' of software systems. So, we have merged 'Fault tolerance' with 'Reliability'. Some of them were totally ignored for having an extremely low percentage of reviews in the datasets. For example, very few users have mentioned security issues of software systems in the dataset which usually refers to system access and data security. So, we have excluded the data to get rid of the imbalanced dataset.

4. **Implementation Difficulties:** We have used the BERT and RoBERTa model to get a comparative result. When we fine tune a model with the annotated dataset it took plenty of time. We had to change the values of different parameters to get the best precision and recall value possible. But per evaluation took more than 4-5 hours on an average. So, it is very time consuming. To minimize the evaluation time we had to have high computational resources. When we evaluate with better computational resources the evaluation time is reduced to 40-55 minutes on an average for each evaluation. Other than that, the two models we have used have different variations. The best variation for our dataset was very difficult to choose.

1.5 Contributions

In this research we have presented a novel dataset named UR-NFR. We have collected datasets of user reviews. There are a total of 4000 user reviews in the datasets. From all of the reviews we have extracted 2623 reviews that address user requirements. We annotated the datasets into different non-functional requirements and functional requirements.

Firstly, we have identified requirements and non-requirements from user reviews. The user reviews were then annotated into total seven categories: six categories of non-functional and one functional requirements class. That means, we have used seven categories more than others. In this topic, others do not have more than five categories including functional requirements.[28]

Moreover, we have achieved better binary classification results with high accuracy value. So the predicted binary classification will be mostly accurate. We also had successful classification findings for non-functional and functional criteria. Because non-functional requirements comprise all software system improvement or quality information, our findings can be used to obtain the most accurate classification possible. Then, by assisting developers in producing successful products, it is feasible to improve software systems.

2 Literature Review

Researchers have worked extensively on the topic of requirements classification, employing a variety of methodologies and offering several novel strategies. Previous related works on software requirements classification and analysis is discussed in this section.

2.1 Machine Learning Approaches in requirements classification

Kurtanovi et. al[25] worked on automatic classification of functional and non-functional requirements using supervised machine learning. They reached a recall and precision up to 92% with Support Vector Machine (SVM) and using lexical features on the dataset Quality attribute NFRs from the RE'17 Data Challenge[10]. While with automatic extraction, they achieved up to 72% of precision and recall.

The study[12] compares the performance of three important vectorization techniques (BoW=Bag of Words, TF-IDF=Term Frequency-Inverse Document Frequency, and Chi-Squared) for classifying Software Requirements into Functional Requirements (FR) and Non-Functional Requirements (NFR), as well as the sub-classes of Non-Functional Requirements. They discovered that combining TF-IDF and Logistic Regression(LR) produces the best performance measures for binary classification, non-functional requirements classification, and requirements classifications in general, with an F-measure of 91% for binary classification, 74% for 11 granularity classification, and 78% for 12 granularity classification.

Quba et. al [33] categorized the Functional and Non-Functional Requirements and later classified the NFRs into 11 different sub-classes from "PROMISE_exp" dataset[5] using SVM and KNN. With the same approach (BoW), SVM outperformed KNN. Table 1 shows an overview of their achieved results.

Table 1: Detailed summary of the PROMISE exp dataset's Binary classification into Requirements (R) and Non-Requirements (NoR) types.

Task	Model	Precision	Recall	F1 Score
Binary Classification of FR and NFR	SVM	**0.90**	**0.90**	**0.90**
	KNN	0.82	0.82	0.82
Classification of subcategories of NFR	SVM	**0.68**	**0.67**	**0.66**
	KNN	0.56	0.48	0.49

While we found many studies on requirements classification, there are only a few who worked on classification of user requirements from **user reviews** which is a huge source of data. Lu & Liang[28] worked on non-functional requirements classification from augmented user reviews, four classification techniques BoW, TF-IDF, CHI2, and AUR-BoW with three machine learning algorithms Naive Bayes, J48, and Bagging were used. They used user reviews from whatsapp and iBook as their dataset. Figure

1 highlights the overall execution of the classification process from dataset collection throughout the evaluation of classifiers. The process consists of a total of six phases.

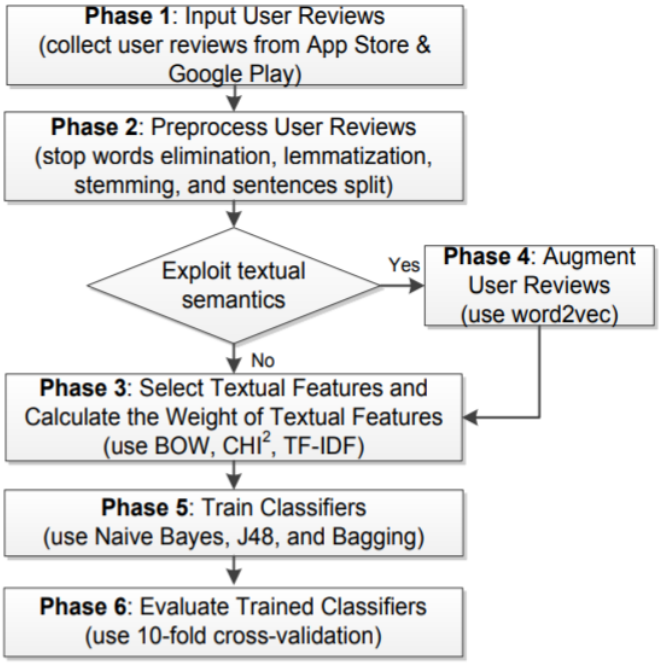


Figure 1: Stages of user reviews classification procedure followed by Lu & Liang[28]

Figure 2 presents a summary of the findings of the experiment acquired using the four classification approaches combined with three machine learning algorithms. All of the combinations achieve precision, recall, and F-measure more than 64.4 percent. With a precision of 71.4 percent and a recall of 72.3 percent, the combination of AUR-BoW and Bagging obtains the greatest F-measure of 71.8 percent.

	BoW			**TF-IDF**			**CHI²**			**AUR-BoW**		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
Naïve Bayes	0.665	0.644	0.654	0.668	0.649	0.658	0.669	0.654	0.661	**0.720**	0.653	0.685
J48	0.656	0.674	0.665	0.661	0.678	0.670	0.661	0.680	0.670	0.677	0.690	0.684
Bagging	0.685	0.694	0.690	0.689	0.699	0.694	0.688	0.697	0.692	0.714	**0.723**	**0.718**

Figure 2: Overview of the result achieved by Lu & Liang[28] combining different Classification Technique with Machine Learning Algorithm

2.1.1 Important Concepts

i) Bag-of-Words : The Bag-of-Words model is a common representation approach for item classification and is often used to represent textual materials. Each retrieved key point is quantized into one of the visual words, and each image is represented by a histogram of the visual words. A clustering method (e.g., K-means) is commonly used to generate the visual words for this purpose.[42]

ii) Term Frequency - Inverse Document Frequency : The term frequency-inverse document frequency (TF–IDF, TF*IDF, or TFIDF) is a numerical statistic in information retrieval that is

designed to describe how essential a word is to a document in a collection or corpus.[35]

In information retrieval, text mining, and user modeling, it is frequently employed as a weighting factor. The TF-IDF value increases according to the number of times a word appears in the document and is offset by the number of documents in the corpus that include the term, which helps to account for the fact that some terms appear more frequently than others.[4]

iii) Chi-squared : Chi-Squared is an algorithm for selecting features. It is a popular statistical test that is said to outperform other algorithms. [14]. It was used to determine textual aspects linked to user reviews while taking into account the user review type information. CHI2 is defined in Formula 1:

$$CHI^2(t_i, C_k) = \frac{N * (ad - bc)^2}{(a + c) * (b + d) * (a + b) * (c + d)} \tag{1}$$

iv) k-NN classifier : The k-NN classifier is a supervised technique that uses labeled data to train classifiers. It classifies a test item based on which previously categorized things are the most similar to the current test item. It locates the k closest neighbors and delivers the test item's classification based on a majority vote of those neighbors. The proximity of two things is measured using a distance metric. The Euclidean distance is frequently used as a numerical attribute metric. The distance between nominal values is binary zero if the values are the same and one if they differ. Custom distance functions unique to current challenges may be used by k-NN classifiers. The capacity to progressively train as new things are categorized, to classify several forms of data, and to handle a high number of item characteristics are all advantages of k-NN classifiers. The main disadvantage of k-NN classifiers is that classification requires O(n) time if n objects are stored.[37]

v) Naïve Bayes and Support Vector Machine : Given a single text, the Naïve Bayes classifier selects the class with the highest probability from a series of training data sets. It is assumed that each feature of a class exists independently of the other characteristics in the class. In real-world challenges, the method works well. SVM classifiers determine the best separator between two classes.[37] [38]

2.2 Deep Learning Approaches in requirements classification

In this study[29], we found out that CNN was used to classify, a natural language written, software requirements in the following 12 categories: Functional, Availability, Legal, Look and feel, Maintainability, Operational, Performance, Scalability, Security, Usability, Fault tolerance, and Portability. Figure 3 shows the architecture of their proposed CNN model. They trained their model with the PROMISE Corpus dataset and achieved an average F measure of 77% with 81% of precision and 78% of recall.

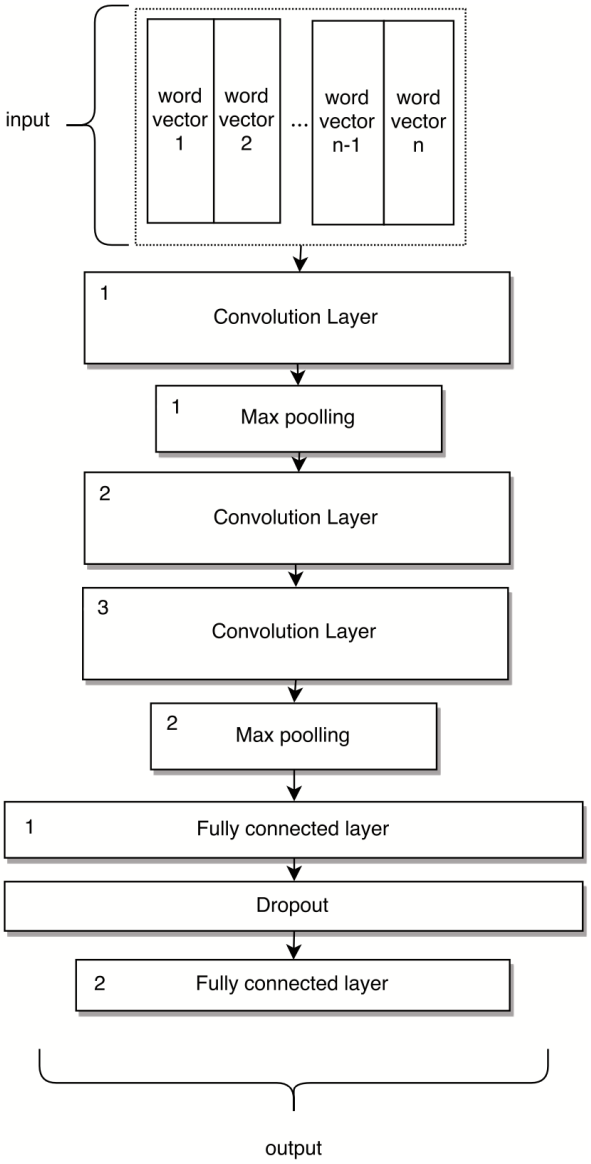


Figure 3: Convolutional Neural Network Architecture Proposal for Software Requirements Classification

In another research work[3], the authors constructed two models. One is using ANN and another using CNN which can effectively classify NFRs into multiple classes. For training their models, they used the International Requirements Engineering Conference's 2017 Data challenge dataset [10] and the

PROMISE dataset [5]. The ANN model was trained with 4 classes while the CNN was trained with a total of 5 classes of non-functional requirements. They attained precision between 82 and 94 percent, recall between 76 and 97 percent, and an F-score between 82 and 92 percent for CNN. When it came to the ANN model, they got precision between 82 and 90 percent, recall between 78 and 85 percent, and an F-score of 84 percent.

Later, Rahman et. al[34] classified non-functional requirements from OpenScience tera-PROMISE software requirement dataset using the RNN variants LSTM, and GRU algorithms and compared their performance with CNN[29]. The CNN and GRU models were shown to be less appropriate for NFR classification from textual requirements than LSTM. LSTM's classification accuracy is 6.1 percent higher than that of GRU, indicating that RNN variations can improve classification results. Figure 4 shows the overview of their achieved results.

	Precision			**Recall**			**F1-score**		
Fold	**CNN**	**GRU**	**LSTM**	**CNN**	**GRU**	**LSTM**	**CNN**	**GRU**	**LSTM**
1	0.80	0.95	0.98	0.79	0.95	0.97	0.77	0.95	0.97
2	0.75	0.97	0.95	0.76	0.97	0.97	0.73	0.97	0.96
3	0.79	0.99	0.98	0.76	0.99	0.94	0.75	0.99	0.95
4	0.81	0.97	0.98	0.81	0.97	0.97	0.80	0.97	0.97
5	0.81	0.97	0.98	0.78	0.97	0.97	0.76	0.97	0.97
6	0.83	0.97	0.95	0.78	0.97	0.97	0.76	0.97	0.96
7	0.84	0.94	0.96	0.85	0.94	0.94	0.82	0.94	0.94
8	0.85	0.88	0.98	0.77	0.88	0.97	0.76	0.88	0.97
9	0.85	1.00	1.00	0.81	1.00	1.00	0.81	1.00	1.00
10	0.80	0.97	0.97	0.75	0.97	0.97	0.74	0.97	0.97
Avg	**0.813**	**0.961**	**0.973**	**0.785**	**0.961**	**0.967**	**0.77**	**0.961**	**0.966**
Std	**0.032**	**0.033**	**0.015**	**0.027**	**0.033**	**0.017**	**0.03**	**0.033**	**0.015**

Figure 4: CNN[29], GRU and LSTM Results[34]

2.3 Transfer Learning Approaches in requirements classification

After the trend of machine learning and deep learning we noticed the use of transformer based language model in requirements classification. Two significant works related to requirements classification is described below:

In[20], to categorize requirements, a fine-tuned (BERT) model entitled "NoRBERT" was proposed and validated on the PROMISE dataset. They applied their technique to the tasks based on the datasets:

- Task1: Upon this initial NFR dataset, binary categorization of F/NFR. To illustrate the NFR classes, they consolidated all NFR subclasses.
- Task2: Binary and multiclass categorization of all NFR in the core NFR dataset using the four most common NFR subclasses (US, SE, O, PE).
- Task3: In the core NFR dataset, multiclass categorization of all NFR subclasses on NFR.

They examined precision (P), recall (R), and F1 measure for all tests. They also presented the weighted average F1-score (A) over anticipated classes for multiclass classifications. The activities were assessed in a variety of circumstances. They defined a single stratified 75 percent train and 25% test split of the dataset as .75-split. They investigated with under- and oversampling techniques for highly unbalanced binary problems, such as the NFR subclasses. They employed two pre-trained BERT models, the base and large model, both in the cased version, for fine-tuning. They tried the uncased models as well, but the cased models outperformed them. They did not preprocess the requirements and applied the BERT-tokenizer.
The output layer, the classification head, was defined on top of the pre-trained models. They utilized BERT's pooled output. The probability distribution for the various labels was obtained using the Softmax function.

They employed the cross-entropy loss function during training, which quantifies how close the predicted distribution is to the real distribution. The formula is given below:

$$H(p, q) = -\sum_{x} p(x) \log q(x) \tag{2}$$

They employed the AdamW-optimizer, which is a modified version of the popular Adamoptimizer. [24]. They employed a 0.01 weight decay and a 2e-05 maximum learning rate. The base models had a maximum sequence length of 128 while the big models had a maximum sequence length of 50.
They used the logic that larger epoch numbers allow the classifier to fit more closely to the observed data while also posing the danger of overfitting when fine-tuning NoRBERT's hyperparameters, such as

the epoch number. In their trials, systematic increases in epoch numbers revealed that 10 to 32 epochs for binary settings and 10 to 64 epochs for multiclass settings performed best on the task.

Task 1: Binary categorization of F/NFR:

They compared their findings to the state-of-the-art peformances by Kurtanovi´c and Maleej [25], Abad et al. [1] and Dekhtyar and Fong [9]. Figure 5 displays their results in contrast to the other methodologies' reported outcomes. With an F1-score of 90% for functional and 93 percent for non-functional requirements, NoRBERT obtains equivalent results. On NFR, NoRBERT surpasses everything except the highest-scoring technique by Abad et al. [1], which preprocesses the dataset using manually specified dictionaries and rules. NoRBERT, on the other hand, does not require any human preprocessing and can thus be effortlessly transferable to any other dataset.

	F (255)			NFR (370)		
Approach (Parameters)	P	R	F_1	P	R	F_1
10-fold						
K. & M. (word features w/o feat. sel.)	**.92**	.93	**.93**	.93	.92	.92
K. & M. (500 best word features)	**.92**	.79	.85	.82	.93	.87
K. & M. (500 best features)	.88	.87	.87	.87	.88	.87
A. et al. (unprocessed data)	.84	.93	.88	.95	.88	.91
A. et al. (processed data)	.90	**.97**	**.93**	**.98**	.93	**.95**
D. & F. (word2vec, ep.=100, f.=50)	—	—	—	.93	.92	.92
NoRBERT (base, ep.=10)	.91	.90	.90	.93	.94	.93
NoRBERT (base, ep.=10, ES, US)	.88	.88	.88	.92	.92	.92
NoRBERT (base, ep.=10, ES, OS)	.91	.86	.88	.91	.94	.92
NoRBERT (base, ep.=16)	.89	.88	.89	.92	.93	.92
NoRBERT (large, ep.=10, OS)	**.92**	.88	.90	.92	**.95**	.93
p-fold						
NoRBERT (base, ep.=10)	.91	.86	.88	.91	.94	.92
NoRBERT (large, ep.=10)	**.93**	**.88**	**.91**	**.92**	**.95**	**.94**
loPo						
NoRBERT (base, ep.=10, ES)	.91	.88	**.90**	**.92**	.94	.93
NoRBERT (large, ep.=16)	**.92**	**.89**	**.90**	**.92**	**.95**	**.94**

Figure 5: On the PROMISE NFR dataset, F/NFR classification was performed. The top score for each statistic per class is shown in bold.

Task 2: Multiclass categorization of the four most common NFR subclasses

They assessed NoRBERT's performance using one binary classifier for each of the dataset's four most common NFRs, namely Usability, Security, Operational, and Performance. As a result, the multiclass classifier was trained on either the four most common subclasses plus "Other" or all NFR subclasses. Figure 6 shows the findings and compares them to Kurtanovi'c and Maleej's [24]. Classifiers trained on all NFR classes are denoted by approaches with the suffix "all." The weighted average F1-score across all classes, weighted by frequency of appearance, is shown in the last column. The findings of NoRBERT are encouraging, with a 10-fold cross-validation weighted average F1-score of up to 83

percent for binary classification and 87 percent for multiclass classification.

	Approach	Parameters	Usability (US) P	R	F_1	Security (SE) P	R	F_1	Operational (O) P	R	F_1	Performance (PE) P	R	F_1	A
10-fold	K. & $M_{\cdot bin}$	(w/o feature selection)	.81	.85	.82	.91	.90	*.88	.72	.75	.73	.93	**.90**	***.90**	.83
	K. & $M_{\cdot bin}$	(50 best features)	.70	.57	.61	.81	.77	*.74	.78	.50	*.57	.87	.57	.67	.65
	K. & $M_{\cdot bin}$	(500 best features)	.80	.71	.74	.74	.81	*.74	.72	.73	*.71	.87	.81	*.82	.75
	$NoRBERT_{bin}$	(base, ep.=10)	.81	.69	.74	**.93**	.82	.87	.80	.53	.64	.88	.80	.83	.77
	$NoRBERT_{bin}$	(base, ep.=10, OS)	.78	.70	.74	.90	.86	.88	.88	.71	.79	.88	.80	.83	.81
	$NoRBERT_{bin}$	(base, ep.=16, OS, ES)	**.89**	.70	.78	.89	.89	.89	**.90**	.71	.79	.88	.81	.85	.83
	K. & $M_{\cdot multi}$	(w/o feature selection)	.65	.82	*.70	.81	.77	*.75	.81	.86	.82	.86	.81	*.80	.76
	K. & $M_{\cdot multi}$	(50 best features)	.49	.68	.55	.60	.50	*.39	.42	.47	*.33	.85	.53	*.63	.47
	K. & $M_{\cdot multi}$	(500 best features)	.70	.66	*.64	.64	.53	.56	.47	.62	*.51	.81	.74	.76	.61
	$NoRBERT_{multi}$	(base, ep.=32)	.78	.84	.81	.89	.85	.87	.79	.73	.76	.88	.78	.82	.82
	$NoRBERT_{multi}$	(large, ep.=32)	.86	.82	.84	.91	.91	**.91**	.83	.71	.77	.90	.81	.85	.84
	$NoRBERT_{multi\_all}$	(base, ep.=16, ES)	.86	**.91**	**.88**	.77	**.92**	.84	.77	.79	.78	.87	.83	.85	.84
	$NoRBERT_{multi\_all}$	(base, ep.=50)	.78	.85	.81	.78	**.92**	.85	.83	**.84**	**.83**	**.94**	.87	**.90**	.85
	$NoRBERT_{multi\_all}$	(large, ep.=50)	.83	.88	.86	.90	**.92**	**.91**	.78	**.84**	.81	.92	.87	**.90**	**.87**
p-fold	$NoRBERT_{bin}$	(base, ep.=16, OS)	.75	.70	.73	.77	.83	.80	.72	.61	.66	.84	.50	.63	.71
	$NoRBERT_{bin}$	(large, ep.=16, OS)	.83	.75	.78	**.89**	.82	.85	**.85**	.71	**.77**	.88	.74	.80	.80
	$NoRBERT_{multi}$	(base, ep.=16)	.65	.72	.68	.77	.87	.82	.69	.63	.66	.81	.58	.68	.71
	$NoRBERT_{multi}$	(large, ep.=32)	**.84**	.80	**.82**	**.89**	.89	.89	.78	.70	.74	**.91**	**.80**	**.85**	**.83**
	$NoRBERT_{multi\_all}$	(base, ep.=32)	.61	.77	.68	.81	.86	.83	.61	.84	.71	.80	.65	.72	.74
	$NoRBERT_{multi\_all}$	(large, ep.=50)	.71	**.82**	.76	.88	**.92**	**.90**	.71	**.85**	**.77**	.85	.76	.80	.81
loPo	$NoRBERT_{bin}$	(base, ep.=16, OS)	.69	.72	.70	.80	.83	.81	.79	.73	.76	**.91**	.57	.70	.74
	$NoRBERT_{bin}$	(large, ep.=16, OS)	**.87**	.70	.78	.89	.82	.85	**.88**	.79	**.83**	.89	.78	.83	.82
	$NoRBERT_{multi}$	(base, ep.=16)	.65	.78	.71	.77	.89	.83	.67	.56	.61	.86	.57	.69	.71
	$NoRBERT_{multi}$	(large, ep.=32)	.85	.79	**.82**	**.92**	.91	**.91**	.80	.77	.79	.86	**.81**	**.84**	**.84**
	$NoRBERT_{multi\_all}$	(base, ep.=32)	.61	**.81**	.70	.72	.88	.79	.68	.74	.71	.85	.72	.78	.74
	$NoRBERT_{multi\_all}$	(large, ep.=50)	.65	**.81**	.72	.82	**.92**	.87	.66	**.84**	.74	.83	.63	.72	.76

Figure 6: Result evaluation of classifying the most common NFR classes in [20]. The maximum score per metric per class is shown by bold values. F1-scores provided with asterisks do not match precision and recall.

Task 3: Multiclass categorization of all NFR subclasses

They excluded Portability for this task since it had only one representation in the sample. It cannot be in both the training and test sets at the same time, making prediction impossible.

The findings are shown in Figure 7. The multiclass classifier does a decent job. NoRBERT exceeds the findings of Navarro-Almanza et al.'s [29] convolutional neural network-based technique by at least 5% (F1-score of 82 percent vs. 77 percent), demonstrating that transfer learning beats word embedding-based deep learning on the test. To compare NoRBERT to Abad et al. [1] they used a five-fold cross validation method. Except for the Naive Bayes classifier, which requires (manually) preprocessed data, NoRBERT beats all other techniques. Our solution does not need any manual preprocessing and hence can be deployed to new applications with convenience.

The results of this work indicate that NoRBERT can detect underrepresented NFR subclasses even in setups with little training data and when applied to previously unknown projects. It outperforms all other methods that do not preprocess data. As a result, NoRBERT is a viable alternative to current techniques that involve human data preparation.

In another research work of requirements classification, Kici et. al[23] analyzed the performance of multiple pre-trained transformer models, such as BERT, DistilBERT, Roberta, AlBERT, and XLNet,

	Model	A (21)			FT (10)			L (13)			LF (38)			MN (17)			O (62)			PE (54)			SC (21)			SE (66)			US (67)			A
		P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1	
10-fold	bin_{16}	**.93**	.62	.74	.50	.20	.29	.82	.69	.75	.79	.71	.75	.54	.41	.47	**.87**	.77	.82	.88	.78	.82	.70	.67	.68	.88	.88	.88	**.87**	.72	.79	.78
	$multiB_{16}$	.73	.76	.74	**1.0**	.20	.33	.91	.77	.83	.77	.79	.78	**.86**	.35	.50	.77	.79	.78	.87	.83	.85	.68	.71	.70	.77	**.92**	.84	.86	**.91**	**.88**	.79
	$multiB_{32}$	.75	.71	.73	.38	.30	.33	.91	.77	.83	.81	.79	.80	.60	.35	.44	.78	.82	.80	.90	.83	.87	**.76**	**.76**	**.76**	.84	**.92**	.88	.78	.87	.82	.79
	$multiB_{50}$	.77	**.81**	**.79**	.60	.30	.40	.91	.77	.83	.80	.74	.77	.70	.41	.52	.83	**.84**	**.83**	**.94**	.87	**.90**	.64	.67	.65	.78	**.92**	.85	.78	.85	.81	.80
	$multiL_{32}$	.70	.76	.73	.56	.50	.53	**.92**	**.85**	**.88**	**.82**	**.87**	**.85**	.58	.41	.48	.74	.77	.76	.91	**.89**	**.90**	.70	.67	.68	.86	.89	.87	.86	.85	.86	**.81**
	$multiL_{50}$	.80	.76	.78	.60	**.60**	**.60**	.91	.77	.83	.81	.79	.80	.62	**.47**	**.53**	.78	.84	.81	.92	.87	**.90**	**.76**	**.76**	**.76**	**.90**	**.92**	**.91**	.83	.88	.86	**.82**
5x5-fold	LDA(P)	.60	.95	.74	.02	.10	.03	.20	.47	.28	.85	.60	.70	.52	.70	.60	.70	.35	.47	.95	.70	.81	.57	.81	.70	.87	.87	.87	.76	.61	.68	.62
	NB(UP)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	.45
	NB(P)	**.90**	**.90**	**.90**	**.90**	**.97**	**.93**	**1.0**	.75	**.86**	**.94**	**.94**	**.94**	**.82**	**.90**	**.86**	**.91**	.78	**.84**	**1.0**	**.90**	**.95**	**.83**	**.83**	**.83**	**1.0**	**.97**	**.98**	.77	**.97**	**.86**	**.90**
	$multiB_{32}$	.73	.77	.75	.65	.26	.37	.80	**.82**	.81	.77	.75	.76	.54	.40	.46	.79	**.82**	.81	.86	.81	.83	.68	.73	.70	.83	.89	.86	**.79**	.83	.81	.78
p-fold	bin_{16}	**.85**	.79	**.81**	**.50**	.10	.17	.50	.27	.35	.74	.42	.54	.38	.18	.24	**.72**	.61	.66	.84	.50	.63	.66	.60	.63	.77	.83	.80	**.75**	.70	.73	.64
	$multiB_{32}$	.64	.69	.67	.00	.00	.00	.67	.31	.42	.64	.49	.55	**.55**	.32	.41	.61	.84	.71	.80	.65	.72	.68	**.62**	.65	.81	.86	.83	.61	.77	.68	.66
	$multiL_{50}$	.73	**.83**	.78	**.50**	**.25**	**.33**	**.79**	**.58**	**.67**	**.82**	**.67**	**.74**	.52	**.35**	**.42**	.71	**.85**	**.77**	**.85**	**.76**	**.80**	**.72**	**.62**	**.67**	**.88**	**.92**	**.90**	.71	**.82**	**.76**	**.76**
loPo	bin_{16}	.71	.57	.63	**1.0**	**.20**	**.33**	.83	.38	.53	.81	.45	.58	.44	.24	.31	**.79**	.73	**.76**	**.91**	.57	.70	.65	.52	.58	.80	.80	.80	**.69**	.72	.70	.67
	$multiB_{32}$	**.79**	.71	.75	.00	.00	.00	.40	.15	.22	.64	.47	.55	.36	.29	.32	.68	.74	.71	.85	**.72**	**.78**	.68	**.62**	.65	.72	.88	.79	.61	**.81**	.70	.66
	$multiL_{50}$	.78	**.86**	**.82**	.50	**.20**	.27	**.89**	**.62**	**.73**	**.85**	**.58**	**.69**	**.62**	**.47**	**.53**	.66	**.84**	.74	.83	.63	.72	**.76**	**.62**	**.68**	**.82**	**.92**	**.87**	.65	**.81**	**.72**	**.73**

Figure 7: Overview of the result evaluation of multiclass categorization of all NFR subclasses in[20]

on multi-class text classification over SRS documents. They used three SRS datasets to assess the models' performance: DOORS, NFR-PROMISE, and PURE. Their numerical research demonstrates that the transformer models can classify all categories with good accuracy, with the exception of Priority of the requirements. While all models have an accuracy of 80% or above for other classification tasks, the models' accuracy for classifying the Priority does not exceed 60%. Figures 8(a) and 8(b) show the overall findings of their tests, as well as the evaluation metrics derived from the highest performing algorithm for each dataset.

	pretrained model	F1_weighted	accuracy	precision	recall
NFR	bert-base-uncased	0.896	0.896	0.872	0.864
	distilbert-base-uncased	0.910	0.910	0.886	0.880
	roberta-base	0.910	0.910	0.891	0.879
	albert-base	0.899	0.900	0.907	0.832
	xlnet-base-cased	0.914	0.914	0.902	0.879
PURE	bert-base-uncased	0.974	0.974	0.973	0.973
	distilbert-base-uncased	0.961	0.961	0.955	0.961
	roberta-base	0.977	0.977	0.968	0.986
	albert-base	0.958	0.958	0.958	0.960
	xlnet-base-cased	0.964	0.964	0.968	0.958
DOORS Type	bert-base-uncased	0.867	0.873	0.869	0.873
	distilbert-base-uncased	0.866	0.874	0.861	0.874
	roberta-base	0.864	0.870	0.863	0.870
	albert-base	0.860	0.869	0.854	0.869
	xlnet-base-cased	0.861	0.867	0.864	0.867
DOORS Severity	bert-base-uncased	0.788	0.848	0.778	0.848
	distilbert-base-uncased	0.798	0.820	0.785	0.820
	roberta-base	0.791	0.848	0.774	0.848
	albert-base	0.780	0.846	0.741	0.846
	xlnet-base-cased	0.788	0.848	0.778	0.848
DOORS Priority	bert-base-uncased	0.602	0.618	0.599	0.618
	distilbert-base-uncased	0.594	0.611	0.591	0.611
	roberta-base	0.606	0.611	0.603	0.611
	albert-base	0.592	0.597	0.590	0.597
	xlnet-base-cased	0.596	0.609	0.589	0.595

(a) Classification results with five transformer models

		Precision	Recall	F1-score	Support
NFR	Functional	0.963	0.897	0.929	58
	Non-Functional	0.917	0.971	0.943	68
PURE	Functional	0.979	0.920	0.948	50
	Non-Functional	0.920	0.979	0.948	47
DOORS Type	Enhancement	0.817	0.897	0.855	359
	Story	0.926	0.888	0.907	367
	Plan Item	0.593	0.640	0.615	75
	JUnit	0.970	0.977	0.974	133
	Test Task	0.963	0.952	0.958	166
	Maintenance Item	0.590	0.719	0.648	32
	Other	0.250	0.028	0.050	36
DOORS Priority	High	0.663	0.750	0.704	608
	Medium	0.556	0.523	0.539	453
	Low	0.352	0.178	0.236	107
DOORS Severity	Normal	0.852	0.995	0.918	984
	Major	0.421	0.073	0.125	109
	Blocker	0	0	0	26
	Critical	0	0	0	30
	Minor	0	0	0	19

(b) Detailed performance values over individual classes

Figure 8: Classification results [23]

2.3.1 Important Concepts

i) BERT

BERT [11] stands for Bidirectional Encoder Representations from Transformers and is based on Transformers, a contextualized word representation model based on a masked language model and pre-trained using bidirectional transformers[40]. Previous language models were confined to a combination of two unidirectional language models due to the nature of language modeling, where future words could not be seen (i.e. left-to-right and right-to-left). BERT employs a masked language model that predicts a sequence of randomly masked words and may thus be used to learn bidirectional representations. It also achieves cutting-edge performance on most NLP tasks with minimum task-specific architecture adjustment. Incorporating information from bidirectional representations rather than unidirectional representations, according to the creators of BERT, is critical for expressing words in natural language.[26]

The English Wikipedia and the BooksCorpus were used to train BERT initially. BERT models that have been pre-trained and can be fine-tuned have been released.

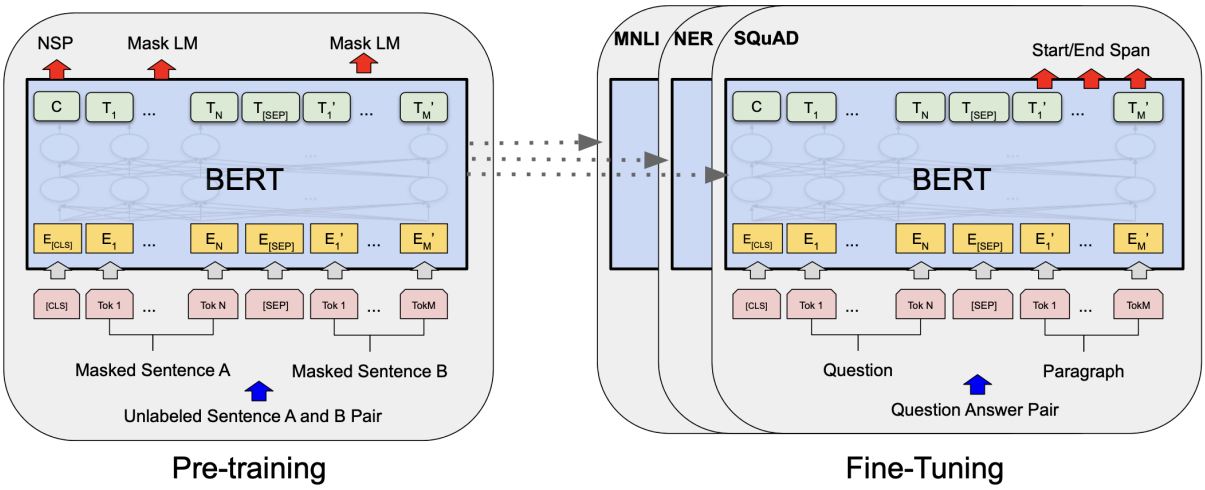


Figure 9: General architecture of Pre-training and fine-tuning BERT model [11]

Figure 9 depicts BERT's entire pre-training and fine-tuning operations, whereas Figure 10 depicts BERT's categorization capabilities. The data has been tokenized. The special token [CLS] is always the first input token in BERT. Similarly, the token [SEP] is a special separator token that can be used to separate sentences, and the token [PAD] is a padding token. The only output of BERT needed for classification and similar downstream tasks is the output BERT provides for the first token ([CLS]), that is the pooled output of all tokens. This combined output may be input into a single-layer feedforward neural network that assigns probability to distinct classes using softmax.

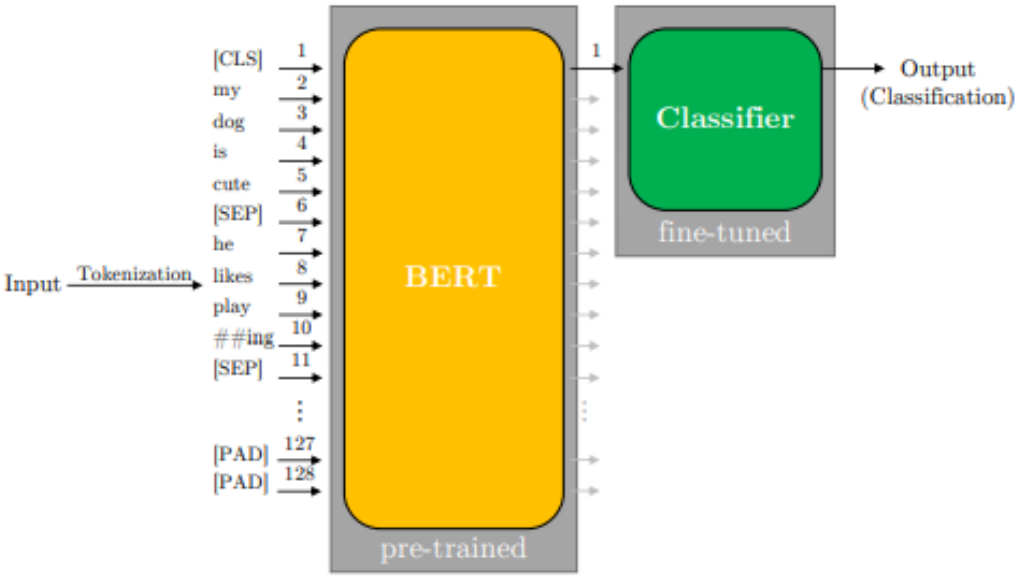


Figure 10: BERT was fine-tuned for categorization using the following architecture.

ii) RoBERTa

RoBERTa[27] is an improved version for training BERT models which can match or exceed the performance of all of the post-BERT methods. The modifications that the researchers have included are:

1. Larger batches of data and longer sequences are used to train the model.
2. Eliminating the aim of predicting the next statement
3. Changing the masking pattern on the fly

They used the following datasets to train the model. They are- English Wikipedia and the BooksCorpus, CC-NEWS, OpenWebText and Stories.

Results achieved by RoBERTa:

- The model obtained the state-of-the-art score on four GLUE tasks: Multi Natural Language Inference (MNLI), QuestionNLI, Semantic Textual Similarity Benchmark (STS-B), and Recognizing Textual Entailments (RTE), and received an 88.5 on the public standings on the GLUE benchmark NLP tasks at the time of its release.
- It can match the previous state-of-the-art results of XLNet on the SQuAD 1.1 and SQuAD 2.0 datasets.
- On RACE benchmark datasets, it also outperforms the BERT(LARGE) model and XLNet.

2.4 Limitations

Traditional approaches such as machine learning or deep learning approaches are promising and demonstrate the possibilities of several strategies for the problem of requirement classification. Many of these techniques, however, are difficult in practice because they are either overfitted to the dataset. Adding to that, they rely significantly on language and sentence structure, or require manual data pre-processing techniques. They do not perform upto the mark when they go through automatic data pre-processing. Furthermore, the methodologies either do not specify or do not sufficiently generalize from project specifics to be applicable to previously encountered projects. This is why, when it comes to requirements classification, we decided to use transfer learning algorithms that promise improved performance and generalizability with less training data.

Most of the previous work has been done on formal specifications and there are a very few works which has been done on user reviews which does not contain any formal language yet addresses lot of software requirements which should be identified and classified. The issues can be the scarcity of user review dataset for training which addresses requirements in the requirements engineering community.

3 Proposed Methodology

3.1 Pipeline of our research

Our research proposes a classifier model that accurately identifies and predicts NFRs from User Reviews So that developers can build a software that has greater acceptance by users. As the number of reviews given by users is huge, and most of the reviews contain information that is irrelevant to the developers. So It is vital to correctly identify requirements from user reviews first.

From the rigorous literature review and method analysis, we have designed our research in that way to answer these two questions-

R1. How correctly our model can identify requirements from user reviews.

R2. How correctly our model can classify Non-Functional Requirements (NFRs) into the subclasses.

For investigating the answers we have employed a robust and structured methodology for our research. The overall pipeline of the methodology is illustrated in the figure 11.

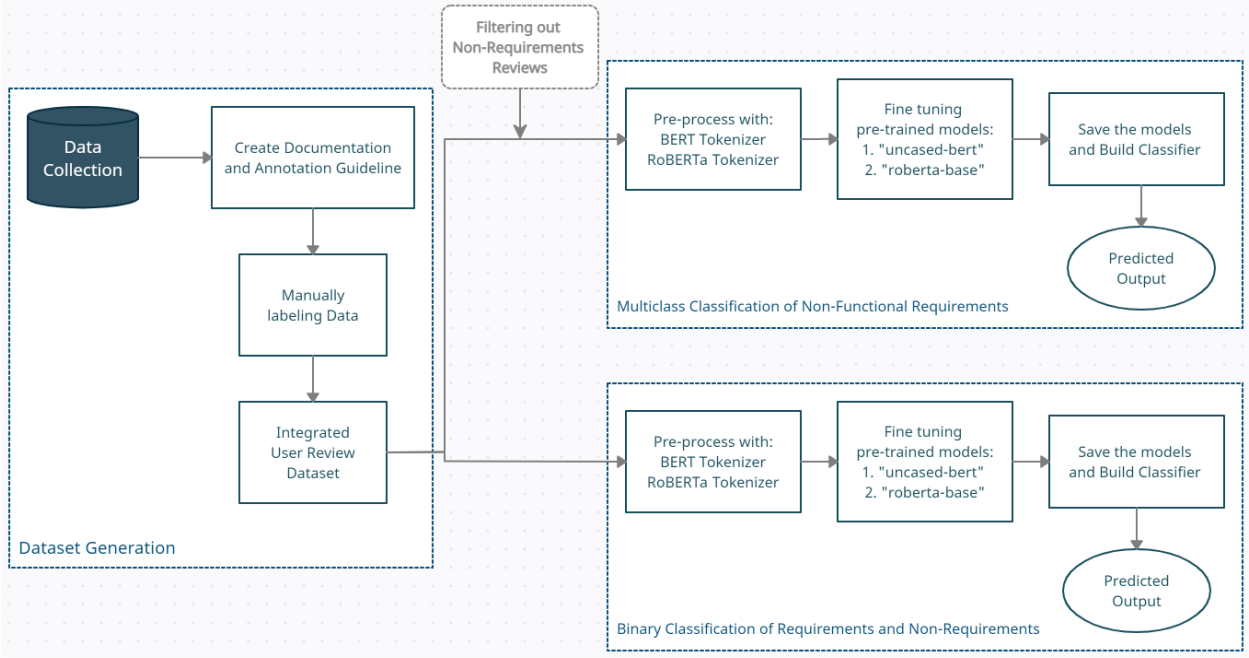


Figure 11: Proposed Methodology

Our research is segmented into two broad stages.

1. Dataset Generation:

A structured dataset consisting of useful data is vital for training any model. Through a rigorous process, we have collected data from the existing and available datasets. We analyzed the dataset to gain insights and patterns so that we can determine whether the data can be utilized to answer our research questions or not.

After that, we outlined a procedure for labeling NFRs. We have also included the factors defining each NFR classes and examples of the sub-classes in the guideline so that labeling can be done avoiding any ambiguity. Following the guidelines and standardized factors by ISO, we have manually classified the reviews into 7 subclasses of NFRs and the rest of the non-requirement reviews were labeled as "other" class. A details explanation of dataset generation is presented in section 6.2

2. User Review Classification:

To answer the research questions we have conducted two classification tasks.

1.Binary Classification of Requirement vs Non-Requirement:
To investigate the answer for R1, all the NFR subclasses are merged and defined as the "Requirement" class and "other" reviews are defined as the "Non-Requirement" class and then we performed binary classification of these two classes.

2. Multiclass Classification of NFR SubClassses :
We filtered out the non-requirement class and performed multiclass classification on all 7 subclasses of NFRs. The classes are-

1. Usability (US)
2. Look and Feel (LF)
3. Reliability (RL)
4. Portability (PO)
5. Performance (PE)
6. Maintainability (MA)
6. Functionality (F)

We investigated recent machine learning and deep learning classification approaches in depth to build our proposed classification model. We have selected the transformer-based language models BERT and

Roberta to develop our classification model. The reasoning behind this is that we expect transformer-based languages models to generalize better results on the relatively smaller training dataset, improving classification performance on unfamiliar projects.

3.2 Experiment Material

In this section, we have described the entire process from preparing annotation guidelines from literature to the composition of a novel annotated dataset. In section 6.2.1 we have ascribed why we classified the user reviews into 7 NFR classes. Then in section 6.2.2 introduces our UR-NFR dataset that is used for experiments.

3.2.1 Non Functional Requirements (NFRs) Type Documentation:

ISO 25010 represents a quality model which represents the user's requirements which can be categorized into 8 types. [22] These are the primary quality attributes of a software system, each with various sub-characteristics. From analyzing a sample of 1000 reviews we found among all other sub-characteristics "User interface aesthetics" was addressed prominently. For that reason, we decided to consider this type of review to classify as the "Look and Feel" class. Another attribute "Compatibility" is the ability of one system to coexist with another in the same environment. [13] As Compatibility can be referred to as the subconcept (co-existence) of Portability and it's quite challenging to distinguish in user reviews. So we decided to exclude the "Compatibility" class to avoid within-class imbalance [25]. Finally for performing classification tasks from user reviews, we chose seven sorts of NFRs. Those are- usability, look and feel, performance, compatibility, reliability, maintainability, portability and functionality class. Before starting annotation first we drafted annotation guidelines describing the definition and key factors of all 7 classes. The purpose of preparing an annotation guideline is to validate the annotated dataset that we produced manually. The factors and sample reviews for each class are given as follows

1. **Usability (US)**

Factors	Description
Operability	The system must be simple to use and easy to control by the users.
Ease of learning	Both beginners and users with previous familiarity with similar systems should find the system easy to learn.
Ease of remembering	For the casual user, the system must be easy to memorize.
Understandability	The user must be aware of the system's capabilities.

Sample reviews of Usability class:

1. "When it refreshes content, it disregards what you are doing (ie scrolling, viewing a tweet) and automatically brings you to the top without warning" (Twitter)

2. “There are so many ads popping up that the app is now unusable because an ad pops up literally every time I close the last” (PhotoEditor)

2. **Look and Feel (LF)**

Factors	**Description**
User experience	The user interface of the system should allow pleasant and fulfilling interaction.
User interface aesthetics	Specifies the system’s appearance and style

Sample reviews of Look and Feel class :

1. “Like the new calling option but a ridiculous place to put the button as keep pressing by mistake when trying to send a photo or video!” (Whatsapp)

2. “The new layout doesnt look nice!!! Please switch it back, it looked so much nicer” (Twitter)

3. **Reliability (RL)**

Factors	**Description**
Maturity	The system must fulfill reliability requirements during regular operation.[21]
Availability	The system must be functioning and accessible when needed.[21]
Fault tolerance	Despite the presence of hardware or software defects, the system must function as intended.[21]
Recoverability	The user must retrieve the immediately damaged data and restore the system to its desired condition.[21]
Consistency	The system must perform the given tasks consistently without failure

Sample reviews of Reliability class :

1. “I have an HTC One M8 and for some reason this app opens and then closes without out any option for me to do anything” (Whatsapp)

2. “I did have it freeze on me a couple of times and I thought it was going to crash, but it managed to recover” (Twitter)

4. **Portability (PO)**

Factors	**Description**
Adaptability	How easily a system or components of a system should be altered or updated to the changing requirements.
Replaceability	The capacity of software application to replace other software in a particular environment
Installability	Performed on software that needs to be installed in a target environment
Compatibility	Describes how a system may coexist in the same ecosystem with another system. For example, Software must be compatible with the operating system’s firewall or antivirus protection.
Interoperability	Determine if two or more components can interact with one another without any communication issue

Sample reviews of Portability class :

1. "The only downside for me (using a sony experia E1) is that my media saves to my phone not my sd card n theres no option to change it like there wae with the older version on my blackberry" (WhatsApp)

2. "kindles browser doesnt support whatever new process facebook uses now" (Facebook)

5. **Maintainability (PO)**

Factors	**Description**
Testability	Examines how easy it is to test system components or the combined product for flaws.
Modifiability	The system must be updated effectively and efficiently without introducing new bugs or degrading existing product quality.
Maintainability	The software system should have capability of being modified
Compliance	It encompasses a wide variety of requirements to verify that the solution complies with regulations established by both internal and external parties to the organization, such as a specification, policy, standard, or legislation.

Sample reviews of Maintainability class :

1. "Ive updated few days ago. There is something wrong with this update. Im not getting notifications properly. Please fix this." (WhatsApp)

2. "July 9 ,2014 update has made this app unusable" (Facebook)

6. **Performance (PE)**

Factors	**Description**
Response Time	The maximum amount of time that the system may take to response to a request.
Throughput	The system's capacity to complete a certain number of transactions in a certain amount of time.
Main memory	The memory space in which programs and data are preserved while the CPU is actively using them.
Secondary Storage	The memory in which programs and data are stored indefinitely.

Sample reviews of Performance class :

1. Sometimes takes too long to respond" (WhatsApp)

2. "Search takes an eternity, and just refreshing the time line takes longer than before" (Twitter)

7. **Functionality (F)**

Factors	**Description**
Completeness	The degree to which the set of functions of the system encompasses all of the given tasks and user objectives.
Correctness	The level of accuracy with which a product or system provides the specified outputs.
Appropriateness	The level to which the functions of a system aid in the completion of specific purpose and objectives.

Sample reviews of Functionality class:

1. "I would have rated 5 if there would have been a send option to multiple users." (WhatsApp)

2. "If it can do the push notification as well on the locked phone would be really great" (Twitter)

3.2.2 UR-NFR Dataset

We obtained the dataset from de Araújo et. al [8] for requirements extraction task which was initially constructed by Dabrowski et. al [7] The datasets contain a total of 2062 user reviews of 8 popular apps from the Amazon Store and the Play Store. This Dataset contains reviews from various popular apps of different domains, so it is relatively generalized. A statistical overview of ReBERT dataset: number of reviews per app is presented in table 2

Table 2: Statistical overview of collected data from ReBert Dataset[8]

App Name	Platform	No of Reviews
Spotify	Google Play	227
Photo Editor Pro	Google Play Store	154
Twitter	Google Play Store	183
WhatsApp	Google Play Store	169
Evernote	Amazon Store	367
Facebook	Amazon Store	327
eBay	Amazon Store	294
Netflix	Amazon Store	341
Total Number of Reviews: 2062		

Additionally, We have used the dataset collected from [2] The dataset is derived from two sub-datasets. The PAN dataset [32] is consist of 1390 reviews and the Maalez dataset [19] contains 3691 reviews classified into four categories respectively of software maintenance tasks. After analyzing the data we filtered out classes that contain non-requirement reviews and used In a total of 1938 reviews for our dataset. The composition of the integrated dataset is shown in the figure 12.

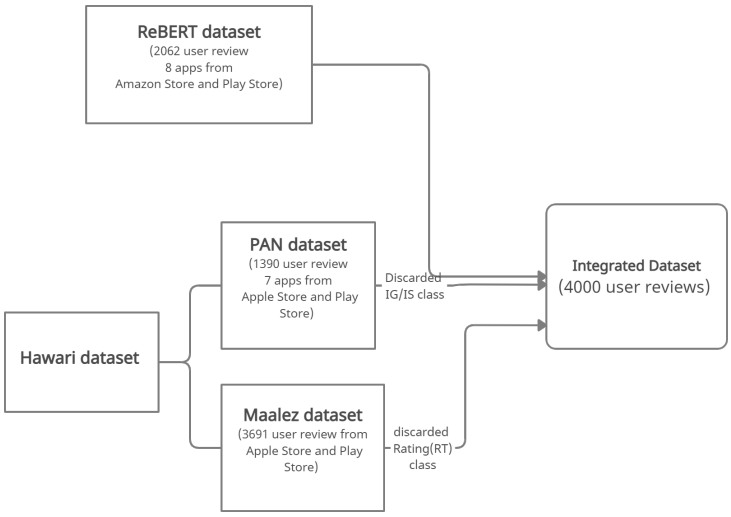


Figure 12: Composition of integrated user review dataset

We have manually labeled a total number of 4000 reviews which was used in our classifier model. We initially followed a standardized set of indicator terms retrieved by Cleland-Huang et. al [24] for each

NFR type. For example, terms like “us”, “understand” are specified as indicator terms for the “usability” class in [24]. From our sample reviews, we also found “easy”, “use” as the most frequently registered word by users in reviews indicating the usability of the app. This characteristic of an app is reported both positively and negatively by users [25]. For example, the review “The features appeal to every personality and it is easy to use” states positive feedback from a user of the Evernote app, whereas “There are so many ads popping up that the app is now unusable because an ad pops up every time I close the last” is negative feedback from a user of PhotoEditor. The latter one should be the concern of developers which can be used for better maintenance and evaluation of the app. We labeled all non requirement reviews as “others”. The figure 13 depicts the requirement vs non-requirement data distribution.

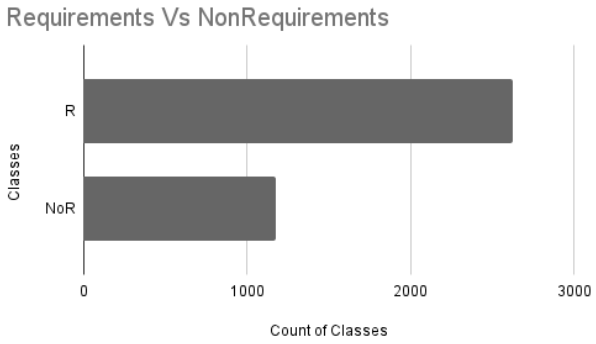


Figure 13: Requirement vs non-requirement data distribution.

In table 3, we have shown the statistical overview of our manually annotated user reviews containing functional and non functional requirements

Table 3: Statistical overview of manually annotated user reviews consist of requirements

Class Name	**No. of Reviews**	**In Percentage**
Usability (US)	372	14.2%
Look and Feel (LF)	240	9.1%
Reliability (RL)	610	23.3%
Portability (PO)	345	13.2%
Performance (PE)	209	8.0%
Maintainability (MA)	353	13.5%
Functionality (F)	494	18.8%
Total Number of NFR Reviews: 2623		

3.3 Proposed architecture of classification of NFRs

We outline the stages of both binary and multi-class classification of user reviews in this section. An illustration of the architecture is shown in the figure 14 which is explained in the following sections.

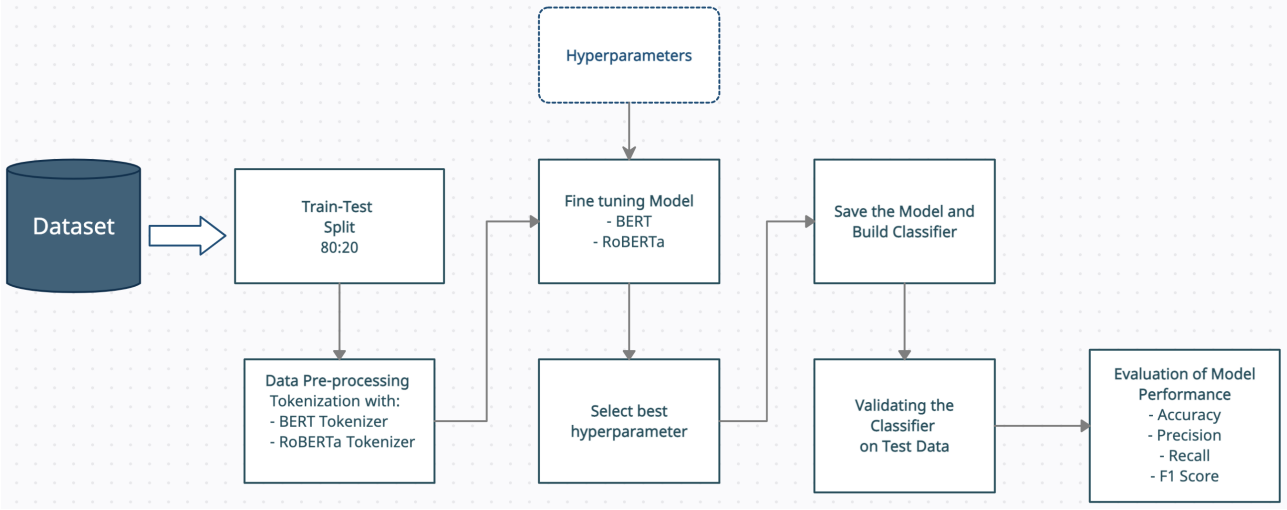


Figure 14: Proposed Architecture

3.3.1 Train-Test split

At first 4000 reviews from our UR-NFR dataset were converted into a CSV file. This is the ground truth for the binary classification of Requirement vs Non-Requirement reviews. For the multi-class classification of NFRs we filtered out the Non-Requirement reviews and converted them into another CSV file that contains 2623 reviews. For each classification model We utilized the sklearn library to split the data into the training and test sets with 80:20 ratio to train the model and evaluate the trained classifier.

3.3.2 Distribution of Review Length

The maximum length of input data is set before feeding data into the pre-trained mode as needs to be the same length. So depending on dataset, the max_length is set.[18]

The distribution of review length per class is plotted by using python library. Histogram 15 shows the distribution of two classes for binary classification and histogram 16 shows the distribution for multi-class classification.

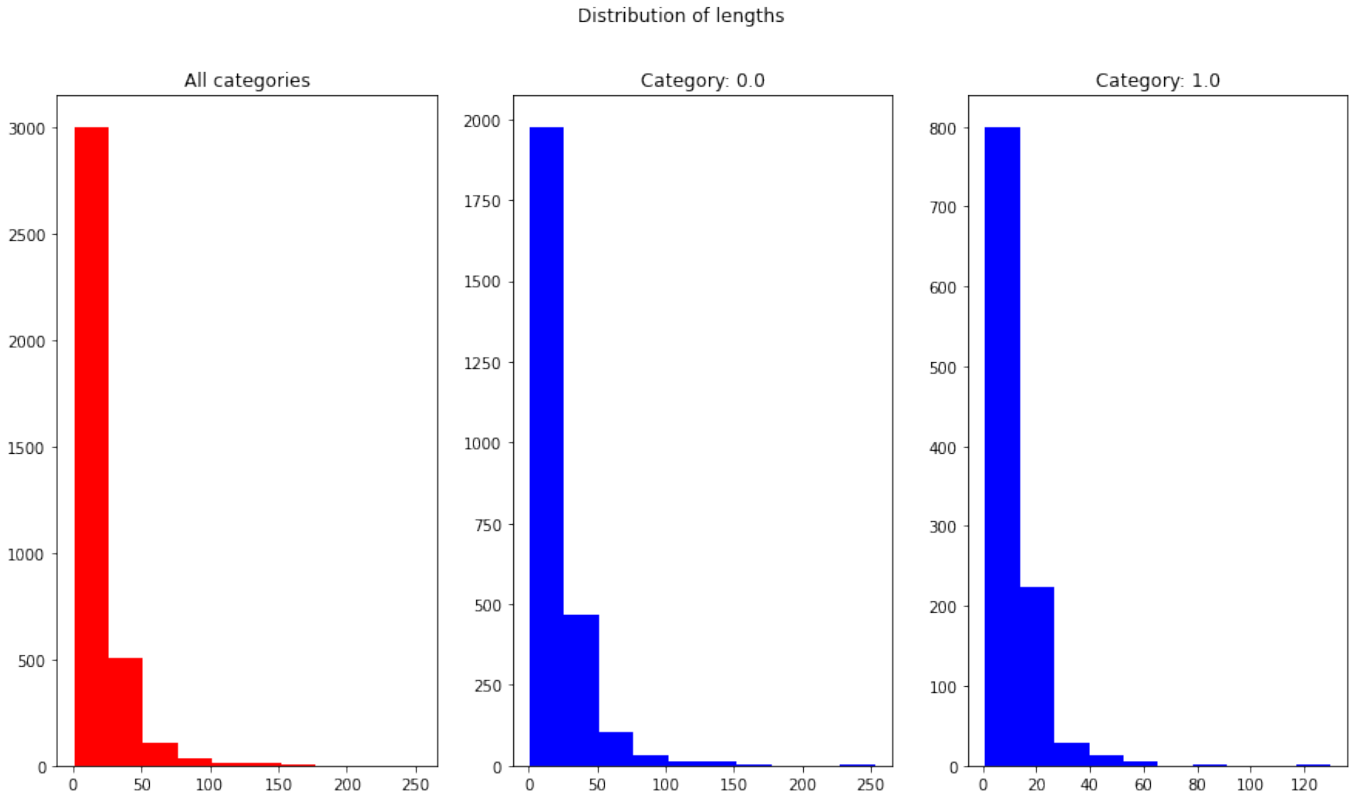


Figure 15: Review length distribution histogram for BinaryClass

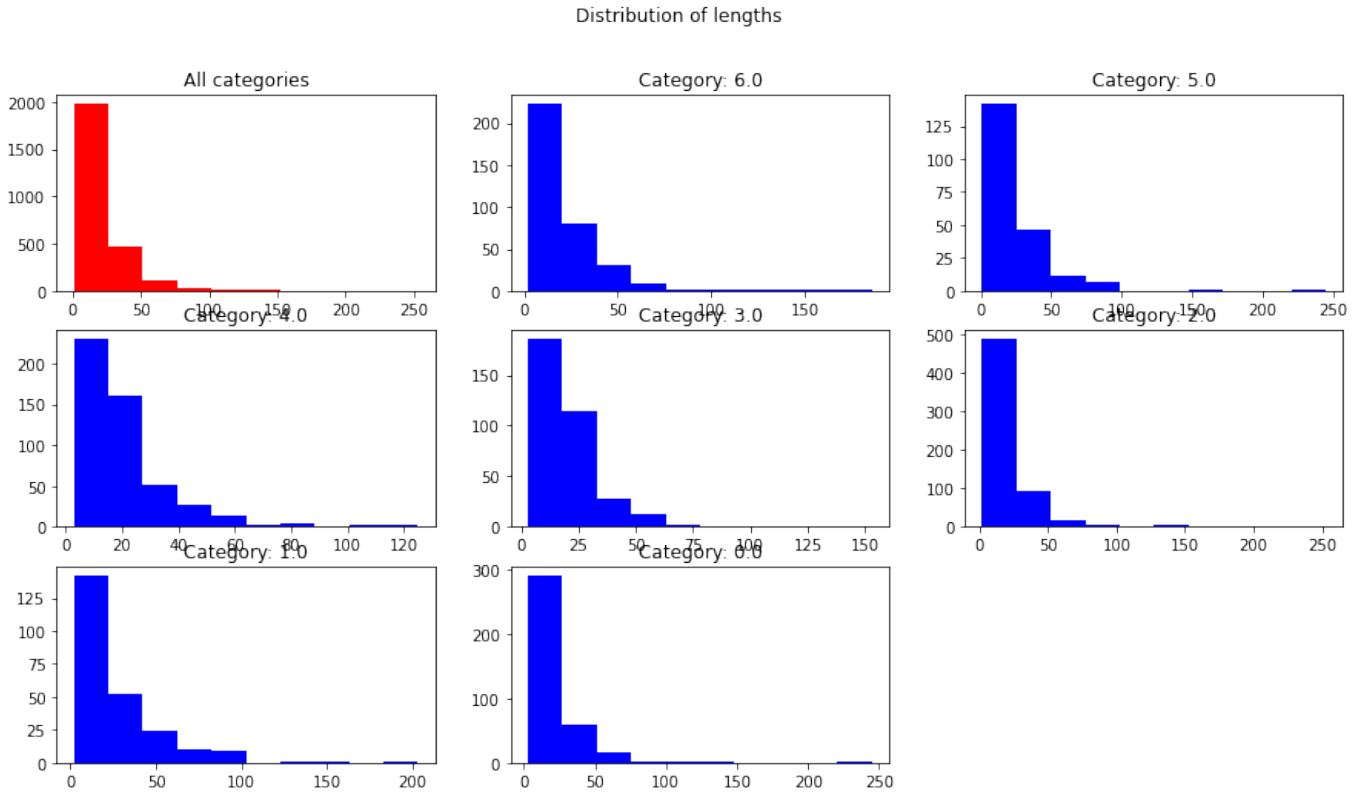


Figure 16: Review length distribution for MultiClass

3.3.3 Data Preprocessing

First the training data must be in a specific format before being fed into the Model. This format contains the following features- A unique Guid, The review we want to classify, a sentence that presents the relationship between sentences while training the model, and lastly the class to that a given review belongs.

Pretrained Transformer-based language models like BERT, RoBERTa use a self mechanism technique to learn the features itself. That is why traditional data preprocessing is not required before feeding into the model. So we didn't need to perform any preprocessing techniques like stop words removal, stemming, and lemmatization. This is because we want to keep the semantic meaning of the review text intact. In our approach, we used model-based tokenizers. BERT, RoBERTa models have Tokenizer packages available in HuggingFace. That library includes Rust-optimized code for data processing and returning all required inputs, such as masks, token ids, and so on. The tokenizer performs all the preprocessing operations - Normalizing text, Removing punctuation, Adding special tokens to distinguish the end of each sentence and splitting and grouping word pieces based on similarity, and lastly using the BERT's own vocabulary, mapping the terms in the review sentences to indexes. [11]

3.3.4 Fine-tuning Model:

Pretrained Transformer models like BERT, RoBERTa learns inner representations of sentences, so fine-tuning these models for a downstream task like text classification is straightforward and produces a state-of-art result. [11] We used the Transformers library to fine-tune these pre-trained models while utilizing GPU resources. We have built two Classifier model: 1. Binary Classifier 2. Multiclass Classifier separately for each classification task. The classifier is developed to facilitate the finetuning of models. Using strategy.scope() of the model, we have set all the hyperparameters to customize the training. We have used the gradient descent optimization algorithm provided by the library Adam with different learning rates and selected 2e-5. During each training, The Adam optimizer iteratively corrects network weights based on the training set generated using randomstate library. [24] The loss function that we used in our training is- *sparse_ categorical_ crossentropy*

First, with a training batch size of 16, the BERT classifier model is trained for three epochs. Then we gradually increased the epoch number and train the model. As small batch size increases computational time so we also increased the batch size. Batch size = 32 was recommended in [11] We train the model for 10, 16, and 32 epoch for both Binary classification and multiclass classification. We trained the RoBERTa classifier for both 16 and 32 batch size. The RoBERTa model have taken comparatively high computational resource. So we used epoch 4 for different learning rate 5e-5 which was recommended by [39]

During every iteration, we saved the state of the model, number of epochs, warmup steps, and so on in

the output directory.
Following the completion of each epoch, we evaluated the model's performance with these metrics-accuracy, precision, recall and f1-score. The details of result evaluation are described in details in section 4

4 Result Analysis

Using a fine-tuned version of BERT and RoBERTa, we investigate the impact of transfer learning here on problem of requirements classification from user reviews. We used BERT model: "uncased-bert" from transformer library and RoBERTa model: "roberta-base" and fine-tuned with our "UR-NFR" dataset. Precision (eq. 3), Recall (eq. 4) and F1-Score (eq. 5) metrics which are typically employed in performance evaluation will be utilized to evaluate our model. [16].

$$Precision = \frac{TP}{TP + FP} \tag{3}$$

$$Recall = \frac{TP}{TP + FN} \tag{4}$$

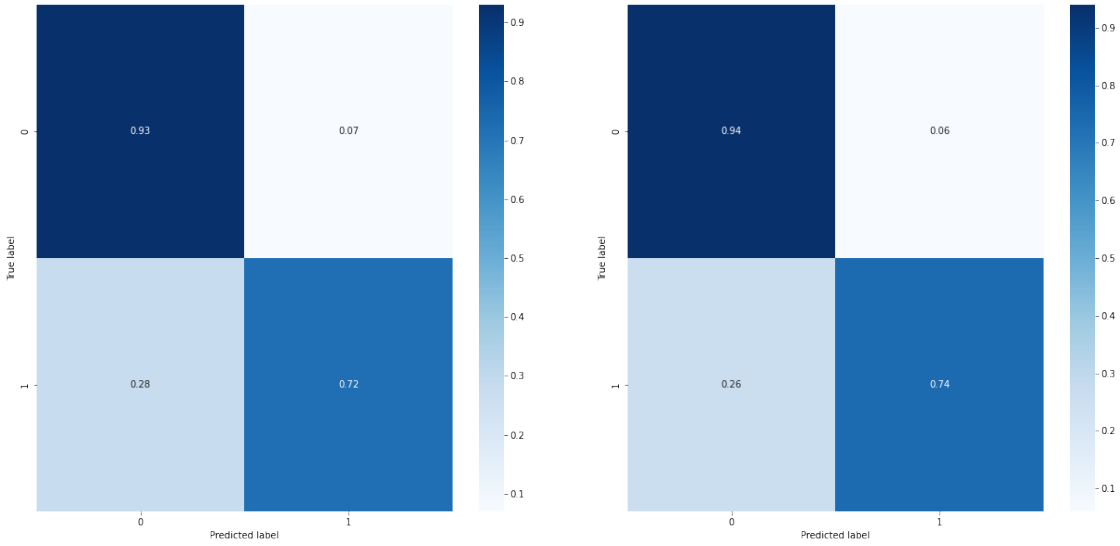
$$F1\_Score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{5}$$

4.1 Binary Classification of Req vs Non-Req

For the first classification task, we want to assess the performance of BERT and RoBERTa on the UR-NFR dataset while identifying user reviews as Requirements(R) or Non-Requirements(NoR).

Table 4 shows the results in depth for Binary classification. With 32 epochs, we discover that BERT scored best in terms of accuracy and precision. The accuracy score is 86.2% and the precision is 83%. Whereas, we got the best score of the recall 76% and F1 score 77.5% for the epochs 16. In terms of accuracy, precision, recall, and F1 Score, RoBERTa surpassed BERT. For epochs 25 and 26, it had an accuracy of 88.47 percent, a recall value of 88.5 percent, and an F1 score of 88.2 percent. The highest precision score is 89% for the number of epochs 4.
In the figure 18, you can some predicted output generated by the BERT classifier for some random input reviews. The classifier identified all the outputs correctly.

RoBERTa's findings for the binary classification of reviews into requirements and non-requirements are encouraging. Even in the confusion matrix shown in Fig 17, we can see that RoBERTa gave better predictions for both Requirement and Non-Requirement classes. The evaluation findings are



(a) For BERT (b) For RoBERTa

Figure 17: Confusion matrix for Binary classification

```
('Simple and easy to use',
 array([-2.9086643e-05, -1.0445076e+01], dtype=float32),
 0,
 'R'),
('Needs a restore purchase button.',
 array([-3.6954196e-05, -1.0206096e+01], dtype=float32),
 0,
 'R'),
('such a trash app.. good for nothing',
 array([-9.4926748e+00, -7.5456635e-05], dtype=float32),
 1,
 'NoR'),
('WOW! This app is absolutely greate',
 array([-9.8021049e+00, -5.5311582e-05], dtype=float32),
 1,
 'NoR')]
```

Figure 18: Predicted result of binary classification by BERT classifier

Table 4: On the UR-NFR dataset, binary classification of Requirements (R) and Non-Requirements (NoR) classes. The maximum score for each measure per model is shown in bold.

Model Name	Batch Size	Epochs [Learning Rate: 2e-5]	Accuracy	Precision	Recall	F1-Score
BERT	32	10	0.86	0.83	0.72	0.771
		16	0.856	0.79	**0.76**	**0.775**
		32	**0.862**	**0.83**	0.717	0.77
RoBERTa	16	4	0.871	**0.89**	0.79	0.82
		25	**0.8847**	0.883	**0.885**	**0.882**
		32	0.884	0.875	0.833	0.85

comparable to state-of-art approaches, with the added benefit of generalizability. Because this approach may be employed even for previously unfamiliar projects with minimum performance loss.

4.2 Multiclass classification of NFR sub-classes

We examined how BERT and RoBERTa perform in multiclass classification on all NFR subclasses and a functional class with the filtered UR-NFR dataset and the evaluation result is described in this section.

BERT has the highest accuracy of 67 percent, as well as 68 percent precision, 66.5 percent recall, and 67.3 percent F1 score, using 80% of data from the dataset for training for the number of epochs 16.0 with a batch size of 32. We can see in Fig 19 the confusion matrix for the BERT classifier. The best result is achieved by the Portability class which is 80% followed by the Reliability and the Look and Feel class.

In the figure 20, we have shown the predicted result of multiclass classification by the BERT classifier for some random input reviews. The classifier identified almost all the outputs correctly except for one. It could not predicted that the review, "*looks bad on iphone 5 its stretched*" belongs to **Look and Feel** class. The classifier predicted **Portability** instead. The reason is that the review contains the word ***iphone 5*** which refers to a device name. As the most of the reviews of *"Portability"* class contained device portability issue so the classifier might be biased and that's why predicted it as PO class whereas the actual issue is related to the **"user interface aesthetics"** of the app which is a factor of Look and Feel.

While RoBERTa has given the best performance for epochs 4 with a batch size of 16, The highest accuracy achieved by it is 70.8% with a precision of 71%, recall 70.5%, and F1 Score up to 70.7%. From the confusion given in Fig 21, we find that the highest number of accurate predictions of non-functional requirements is done by the model for Look and Feel followed by Reliability and Functionality.

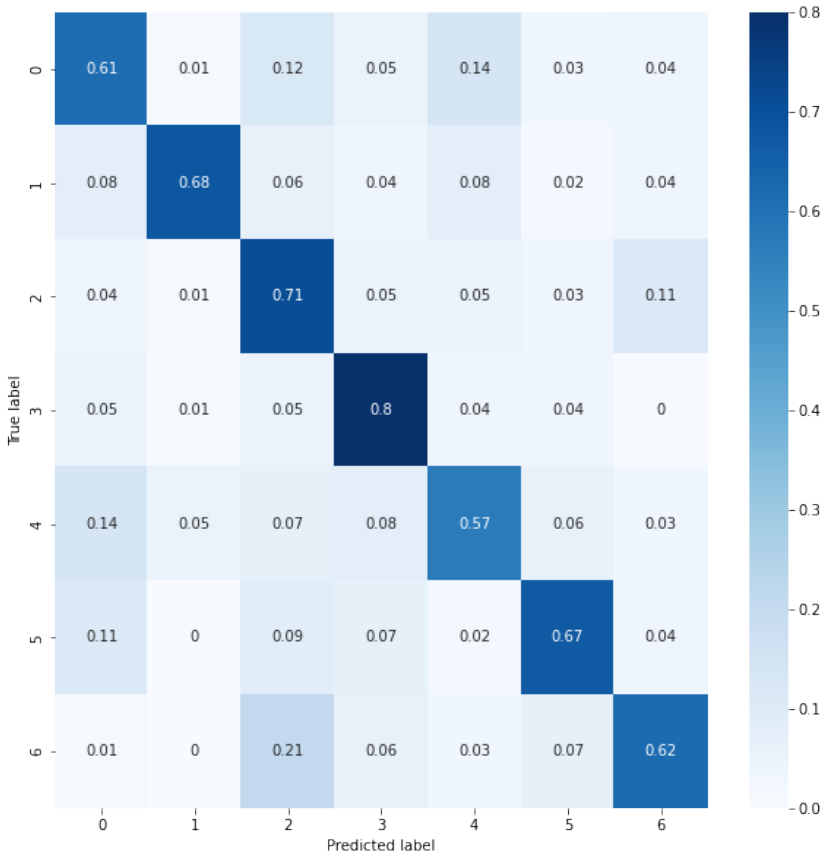


Figure 19: Confusion matrix of BERT for Multiclass classification

```
('Simple and easy to use',
 array([-0.01573997, -6.498516  , -6.1969285 , -6.119627  , -6.1592736 ,
        -5.1813264 , -6.1461334 ], dtype=float32),
 0,
 'US'),
('Needs a restore purchase button.',
 array([-5.872463  , -4.811755  , -5.7543797 , -5.778729  , -0.03724294,
        -7.0378103 , -3.9916673 ], dtype=float32),
 4,
 'F'),
('looks bad on iphone 5 its stretched',
 array([-5.0680623 , -2.705574  , -3.636759  , -0.17841972, -4.6616397 ,
        -3.5548098 , -3.6532488 ], dtype=float32),
 3,
 'PO'),
('It has great interface with Google maps, allowing you to find the restaurant/hotel after you have read the reviews.',
 array([-4.794065  , -0.03902064, -5.6914263 , -5.321366  , -4.4453473 ,
        -5.1053166 , -5.5385537 ], dtype=float32),
 1,
 'LF'),
('You can intuitively navigate through it and very user friendly',
 array([-0.01492407, -6.2170386 , -6.1455145 , -6.195702  , -5.90884   ,
        -5.481104  , -6.3442464 ], dtype=float32),
 0,
 'US'),
('Highly recommended!! The games visual is beautiful',
 array([-5.4732094 , -0.02406215, -5.94816   , -5.80865   , -5.5199223 ,
        -5.2516174 , -5.355952  ], dtype=float32),
 1,
 'LF'),
('takes almost a day to load! CAN IT BE MORE SLOW THAN THIS?',
 array([-5.0550203 , -5.238957  , -3.390922  , -5.152031  , -6.885905  ,
        -0.06165686, -4.87643   ], dtype=float32),
 5,
 'PE')]
```

Figure 20: Predicted result of multiclass classification by BERT classifier

Figure 21: Confusion matrix of RoBERTa for Multiclass classification

RoBERTa outperformed the BERT model in the context of accuracy, precision, recall, and F1 Score for multiclass classification. For each model, the difference in accuracy is somewhere between 2% to 4% for different numbers of epochs. We can see an overview of the results for different epochs numbers in Table 5.

Table 5: On the Filtered UR-NFR dataset, multi - class classification of all NFR subclasses. The maximum score for each measure per model is shown in bold.

Model Name	**Batch Size**	**Epochs** [Learning Rate: 2e-5]	**Accuracy**	**Precision**	**Recall**	**F1-Score**
BERT	32	10	0.653	0.65	0.644	0.647
		16	**0.67**	**0.68**	**0.665**	**0.673**
		32	0.648	0.65	0.64	0.645
RoBERTa	16	4	**0.708**	**0.71**	**0.705**	**0.707**
		4 [LR: 5e-5]	0.66	0.67	0.66	0.665
		25	0.695	0.70	0.68	0.689

4.3 Final Verdict

From comparative analysis, we can say that RoBERTa outperformed BERT in case of both binary and multiclass classification. It achieved higher accuracy and F1 score for both the tasks.

5 Limitations

The limitations of our research work are highlighted in this section.

We have created a novel dataset which is one of our major contributions to this research work. Though our annotation guideline's document has been validated, our annotated dataset has not been validated yet. So, there is a chance that a few reviews containing non-functional requirements might be incorrectly labeled. While annotating the reviews, we found out that some reviews belong to more than one type of requirement class. So, the selection of requirement classes for some reviews is biased in the dataset. Also, we had to exclude some non-functional requirement classes such as security, compatibility, etc due to the low percentage of data.

Overall, the accuracy of the models can be improved if it's trained with more correctly labeled and balanced data containing non-functional requirements.

6 Conclusions

In a nutshell, we aim to help developers to deliver successful software built from user-focused requirements. To deliver a successful software product, developers should keep users' concerns and requirements in mind while designing and developing the software. Because ultimately it's the users who will use the software. As a result, detecting non-functional requirements from user reviews that follows no structured format and automating the classification would be a beneficial software industry practice.

We assessed how well fine-tuned version of the BERT and RoBERTa models perform in identifying requirements and classifying NFRs from user reviews. We developed a novel dataset "UR-NFR" which was used for training the pretrained BERT and RoBERTa models. Our finding is that, RoBERTa surpassed BERT in binary requirement and non-requirement classification, as well as multiclass classification of NFRs from user reviews.

6.1 Future Directions

Extending the dataset for overall performance improvement of the fine-tuned models and increasing the number of the NFR sub-categories for the classification task are two potential future direction of this work. Another direction could be that employing other pretrained transformer-based language models (e.g. ALBERT, DistilBERT) for performance comparison.

References

[1] Z. S. H. Abad, O. Karras, P. Ghazi, M. Glinz, G. Ruhe, and K. Schneider. What works better? a study of classifying requirements, 2017.

[2] assem hawari. A dataset of mobile application reviews for classifying reviews into software engineering's maintenance tasks using data mining techniques. In *Mendeley Data*, page V2, 2019.

[3] C. Baker, L. Deng, S. Chakraborty, and J. Dehlinger. Automatic multi-class non-functional software requirements classification using neural networks. In *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*, volume 2, pages 610–615, 2019. doi: 10.1109/COMPSAC.2019.10275.

[4] J. Beel, B. Gipp, S. Langer, and C. Breitinger. paper recommender systems: A literature survey. *International Journal on Digital Libraries*, 17(4):305–338, 2016.

[5] G. Boetticher. The promise repository of empirical software engineering data. *http://promisedata.org/repository*, 2007.

[6] L. Chung, B. A. Nixon, E. Yu, and J. Mylopoulos. *Non-functional requirements in software engineering*, volume 5. Springer Science & Business Media, 2012.

[7] J. Dąbrowski, E. Letier, A. Perini, and A. Susi. Mining user opinions to support requirement engineering: an empirical study. In *International Conference on Advanced Information Systems Engineering*, pages 401–416. Springer, 2020.

[8] A. F. de Araújo and R. M. Marcacini. Re-bert: automatic extraction of software requirements from app reviews using bert language model. In *Proceedings of the 36th Annual ACM Symposium on Applied Computing*, pages 1321–1327, 2021.

[9] A. Dekhtyar and V. Fong. Re data challenge: Requirements identification with word2vec and tensorflow. In *2017 IEEE 25th International Requirements Engineering Conference (RE)*, pages 484–489, 2017. doi: 10.1109/RE.2017.26.

[10] A. Dekhtyar and V. Fong. Re data challenge: Requirements identification with word2vec and tensorflow. In *2017 IEEE 25th International Requirements Engineering Conference (RE)*, pages 484–489. IEEE, 2017.

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

[12] E. Dias Canedo and B. Cordeiro Mendes. Software requirements classification using machine learning algorithms. *Entropy*, 22(9):1057, 2020.

[13] J. Eckhardt, A. Vogelsang, and D. M. Fernández. Are" non-functional" requirements really non-functional? an investigation of non-functional requirements in practice. In *Proceedings of the 38th International Conference on Software Engineering*, pages 832–842, 2016.

[14] G. Forman et al. An extensive empirical study of feature selection metrics for text classification. *J. Mach. Learn. Res.*, 3(Mar):1289–1305, 2003.

[15] P. B. Goes, M. Lin, and C.-m. Au Yeung. "popularity effect" in user-generated content: Evidence from online product reviews. *Information Systems Research*, 25(2):222–238, 2014.

[16] M. Grandini, E. Bagli, and G. Visani. Metrics for multi-class classification: an overview. *arXiv preprint arXiv:2008.05756*, 2020.

[17] E. C. Groen, S. Kopczyńska, M. P. Hauer, T. D. Krafft, and J. Doerr. Users—the hidden software product quality experts?: A study on how app users report quality aspects in online reviews. In *2017 IEEE 25th international requirements engineering conference (RE)*, pages 80–89. IEEE, 2017.

[18] P. Gupta, S. Gandhi, and B. R. Chakravarthi. Leveraging transfer learning techniques-bert, roberta, albert and distilbert for fake review detection. In *Forum for Information Retrieval Evaluation*, pages 75–82, 2021.

[19] E. Guzman and W. Maalej. How do users like this feature? a fine grained sentiment analysis of app reviews. In *2014 IEEE 22nd international requirements engineering conference (RE)*, pages 153–162. Ieee, 2014.

[20] T. Hey, J. Keim, A. Koziolek, and W. F. Tichy. Norbert: Transfer learning for requirements classification. In *2020 IEEE 28th International Requirements Engineering Conference (RE)*, pages 169–179, 2020. doi: 10.1109/RE48521.2020.00028.

[21] M. B. Ila and H. Kitapci. Selecting an effective information and communication technology architecture for an education system based on non-functional requirements. In *2014 IEEE 8th International Conference on Application of Information and Communication Technologies (AICT)*, pages 1–3. IEEE, 2014.

[22] ISO/IEC 25010:2011. Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models. Standard, International Organization for Standardization, 2011.

[23] D. Kici, A. Bozanta, M. Cevik, D. Parikh, and A. Başar. Text classification on software requirements specifications using transformer models. In *Proceedings of the 31st Annual International Conference on Computer Science and Software Engineering*, pages 163–172, 2021.

[24] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[25] Z. Kurtanović and W. Maalej. Automatically classifying functional and non-functional requirements using supervised machine learning. In *2017 IEEE 25th International Requirements Engineering Conference (RE)*, pages 490–495. Ieee, 2017.

[26] J. Lee, W. Yoon, S. Kim, D. Kim, S. Kim, C. H. So, and J. Kang. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240, 2020.

[27] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[28] M. Lu 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*, pages 344–353, 2017.

[29] R. Navarro-Almanza, 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)*, pages 116–120. IEEE, 2017.

[30] D. Pagano and W. Maalej. Ieee standard glossary of software engineering terminology. In *IEEE Std 729-1983*, page 1–84. IEEE, 1990.

[31] D. Pagano and W. Maalej. User feedback in the appstore: An empirical study. In *2013 21st IEEE international requirements engineering conference (RE)*, pages 125–134. IEEE, 2013.

[32] S. Panichella, A. Di Sorbo, E. Guzman, C. A. Visaggio, G. Canfora, and H. C. Gall. How can i improve my app? classifying user reviews for software maintenance and evolution. In *2015 IEEE international conference on software maintenance and evolution (ICSME)*, pages 281–290. IEEE, 2015.

[33] G. Y. Quba, H. Al Qaisi, A. Althunibat, and S. AlZu'bi. Software requirements classification using machine learning algorithm's. In *2021 International Conference on Information Technology (ICIT)*, pages 685–690. IEEE, 2021.

[34] M. A. Rahman, M. A. Haque, M. N. A. Tawhid, and M. S. Siddik. Classifying non-functional requirements using rnn variants for quality software development. In *Proceedings of the 3rd ACM SIGSOFT International Workshop on Machine Learning Techniques for Software Quality Evaluation*, pages 25–30, 2019.

[35] A. Rajaraman and J. D. Ullman. *Mining of massive datasets*. Cambridge University Press, 2011.

[36] F. Rustam, A. Mehmood, M. Ahmad, S. Ullah, D. M. Khan, and G. S. Choi. Classification of shopify app user reviews using novel multi text features. *IEEE Access*, 8:30234–30244, 2020.

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

[38] J. Slankas, M. Riaz, J. T. King, and L. A. Williams. Discovering security requirements from natural language project artifacts. 2013.

[39] C. Sun, X. Qiu, Y. Xu, and X. Huang. How to fine-tune bert for text classification? In *China national conference on Chinese computational linguistics*, pages 194–206. Springer, 2019.

[40] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[41] C. Yi, Z. Jiang, X. Li, and X. Lu. Leveraging user-generated content for product promotion: the effects of firm-highlighted reviews. *Information Systems Research*, 30(3):711–725, 2019.

[42] Y. Zhang, R. Jin, and Z.-H. Zhou. Understanding bag-of-words model: a statistical framework. *International Journal of Machine Learning and Cybernetics*, 1(1-4):43–52, 2010.